LAMPIRAN A
LISTING PROGRAM

```
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
import java.io.IOException;
import java.util.*;
import com.nokia.mid.ui.FullCanvas;
import com.nokia.mid.sound.Sound;
import javax.microedition.io.*;
import java.lang.*;
import java.io.*;
import javax.microedition.rms.*;

class GameCanvas extends FullCanvas {

	// Game State (posisi keadaan game game)
	short gameState = 1;

	// game Level (Level dari permainan)
	int gameLevel=1;

	// ticker (gambar berjalan untuk musuh)
	int sec, tick, gameTicker;
	int enemyTick;
	int enemyCounter=0;

	// fps (frquensi per second untuk
mengurangi lack)
	byte fps = 20;


	// power (untuk peluru khusus --> bomb,
dan )
	int powerX, powerY;
	boolean power;
	boolean powerTaken = false;
	int powerOrSpecial;

	// spesial
	int specialX, specialY;

	// screenW, screenH
	int screenW, screenH;

	// standard Font
	Font standardFont;

	// initialisation
	Shooter shooter;
	Ships ships;
	Boss boss;
	Sound sound;
	Asteroid[] asteroid = new Asteroid[];
	Asteroid[] asteroidMed = new
AsteroidMed[];
	Asteroid[] asteroidBig = new
AsteroidBig[];

	// Images
	Image splash;
	Image shipSelect;
	Image UI;
	Image menu;

	// choice
	int choice=0,
		sChoice=0,
		optionChoice=0,
		inGameChoice=0;
	int score=0;


	// starfield
	int[] starfieldpos_W = new int[25];
	int[] starfieldpos_H = new int[25];

	// asteroids
	int[] asteroidCount = new int[7];
	int[] asteroidCountMed = new int[4];
	int[] asteroidCountBig = new int[3];
	int asteroidSize;
	int asteroidSmallNo=7,
		asteroidMedNo=4,
		asteroidBigNo=3;


	// ------------------------------Main Program--
---------------------------

	public GameCanvas(Shooter x) {

		shooter = x;

		screenW = getWidth();
		screenH = getHeight();

		// load splash image (splash for
3 second)
		try {
			splash =
Image.createImage("/splash.png");
		}
		catch (IOException ie){

	System.out.println("failed load splash
menu");
			return;
		}

		standardFont =
Font.getFont(Font.FACE_PROPORTIONAL,
Font.STYLE_PLAIN, Font.SIZE_MEDIUM);

		rand = new Random();

	}

	public void paint (Graphics g) {


		g.setFont(standardFont);

		//---------------------- GAME
STATE ------------------------
		// 1  - TITLE SCREEN
		// 20 - MENU SCREEN
		// 21 - OPTIONS
		// 22 - MISSION - SELECT
SHIPS
		// 24 - CREDIT
		// 25 - HIGHSCORE
		// 30 - GAMEPLAY
		// 40 - ENTER HIGH SCORE
		// 41 - INPUT NAME
```

```
                // 43 - SUBMIT HTTP
HIGHSCORE
                // 45 - IN GAME MENU

                switch (gameState) {
                        case 1 : loadTitle(g);
break;
                        case 20:
loadMenuImage(g);

menuScreen(g);

break;
                        case 21:
loadOptions(g);

options(g);

break;
                        case 22:
loadShipSelectImage(g);

shipSelect(g);

break;
                        case 24: credits(g);
break;
                        case 25:
loadHighScore(g);

break;
                        case 27:
keyConfig(g);

break;
                        case 30:
loadGamePlay(g);


gamePlay(g);

break;
                        case 40:
enterHighScore(score);

break;
                        case 41:
inputName(g);


break;
                        case 44:
httpSubmitResultMsg(g);

break;
                        case 50:
inGameMenu(g);

break;

                        default:
                }
        }

        // load splash screen
        private void loadTitle(Graphics g) {

                if (splash==null) splash =
loadImage("/splash.png");

                g.drawImage(splash, 0, 0,
g.TOP|g.LEFT);

                // hanya untuk 3 detik lalu ke
menu
                if (sec>3) {
                        gameState = 20;
                        sec = 0;

        loadMenuImage=false;
                        splash=null;
                        keybool[4]=false;
                }

        }

        // load menu image
        private void loadMenuImage(Graphics g)
{

                if (!loadMenuImage) {

                        if (menu==null) menu
= loadImage("/menu.png");


                        g.drawImage(menu, 0,
0, g.TOP|g.LEFT);

                        g.setColor(0,0,0);
                        outlinedStrText(g,
"Options", screenW/4+10,screenH/2-40);
                        outlinedStrText(g,
"Mission", screenW/4*3-10,screenH/2-20);

                        outlinedStrText(g,
"Credits", screenW/2,screenH/2);
                        outlinedStrText(g,
"High Score", screenW/4+10,screenH/2+20);
                        outlinedStrText(g,
"Quit", screenW/4*3-10,screenH/2+40);

                        loadMenuImage =
true;
                }
        }

        // load game play
        private void loadGamePlay(Graphics g) {

                if (!loadGamePlay) {

                        if (UI==null) UI =
loadImage("/UI.png");
                        if (Boss==null) UI =
loadImage("/Boss.png");

                        sound = new
Sound(523,150);
                        boss = new
Boss(screenW, screenH);
```

```
                        // ships bullet (peluru di set diawal supaya terus menerus)
                        for (int i = 0; i<MAXBULLET; i++){
                                bullet3[i] = new Bullet(screenW, screenH);
                                bullet4[i] = new Bullet(screenW, screenH);
                                bullet5[i] = new Bullet(screenW, screenH);
                                bullet6[i] = new Bullet(screenW, screenH);

                        }



                        // starfield (layar belakang)
                        RandNum ranum1 = new RandNum();
                        for(int i =0; i < 25; i++){

        ranum1.genRandXY(getWidth(),getHeight());

        starfieldpos_W[i] = ranum1.RANGE_X;

        starfieldpos_H[i] = ranum1.RANGE_Y;
                        }

                        // asteroids small
                        for(int i =0; i < asteroidSmallNo; i++){

        ranum1.genRandXY(getWidth(),getHeight());
                                asteroid[i] = new Asteroid(0);

        asteroid[i].updateAsteroid(ranum1.RANGE_X,ranum1.RANGE_Y-screenH);

        asteroidCount[i] = 0;
                        }

                        // asteroids medium
                        for(int i =0; i < asteroidMedNo; i++){

        ranum1.genRandXY(getWidth(),getHeight());

        asteroidMed[i] = new Asteroid(1);

        asteroidMed[i].updateAsteroid(ranum1.RANGE_X,ranum1.RANGE_Y-screenH);

        asteroidCountMed[i] = 0;
                        }

                        // asteroids big
                        for(int i =0; i < asteroidBigNo; i++){

        ranum1.genRandXY(getWidth(),getHeight());

        asteroidBig[i] = new Asteroid(2);

        asteroidBig[i].updateAsteroid(ranum1.RANGE_X,ranum1.RANGE_Y-screenH);

        asteroidCountBig[i] = 0;
                        }

                                loadGamePlay = true;

                        }
                }
                //-------------------------------------------------------------  set awal |^
                // load ships select screen
                private void loadShipSelectImage(Graphics g) {

                        if (!loadShipSelectImage) {


                                if (shipSelect == null) shipSelect = loadImage("/selection_bordera.png");


                                ships = new Ships(screenW, screenH);


                                loadShipSelectImage = true;

                        }
                }

                // menus screen
                private void menuScreen(Graphics g) {

                        g.setFont(standardFont);

                        choice%=5;

                        if (keybool[7]){

                                choice++;
                                keybool[7]=false;
                        }

                        if (keybool[1]){

                                choice--;
                                if (choice<0) choice=4;
                                keybool[1]=false;
                        }

                        // draw strings
                        g.setColor(255,255,255);
                        strText(g, "Options", screenW/4+10,screenH/2-40);
                        strText(g, "Mission", screenW/4*3-10,screenH/2-20);
```

```
                strText(g, "Credits",
screenW/2,screenH/2);
                strText(g, "High Score",
screenW/4+10,screenH/2+20);
                strText(g, "Quit", screenW/4*3-
10,screenH/2+40);

                // Menu Selection
                g.setColor(255,144,28);

                if (choice == 0) strText(g,
"Options", screenW/4+10,screenH/2-40);
                else if (choice == 1) strText(g,
"Mission", screenW/4*3-10,screenH/2-20);

                else if (choice == 2) strText(g,
"Credits", screenW/2,screenH/2);
                else if (choice == 3) strText(g,
"High Score", screenW/4+10,screenH/2+20);
                else strText(g, "Quit",
screenW/4*3-10,screenH/2+40);

                if (keybool[4]) {
                        // Option
                        if (choice == 0) {

        gameState=21;

        loadOptions=false;
                        }
                        // Mission
                        else if (choice == 1) {

        gameState=22;
                                menu =
null;

        loadShipSelectImage=false;
                        }
                        // Credits
                        else if (choice == 2) {

        gameState=24;
                        }
                        // High Score
                        else if (choice == 3) {

        gameState=25;
                        }
                        // Quit
                        else {
                                reset();

        shooter.destroyApp(false);
                        }
                        keybool[4]=false;

                }
        }

        // ship select screen
        private void shipSelect(Graphics g) {

                int bulX, bulY;

                sChoice%=3;

                if (keybool[5]){
                        sChoice++;
                        for (int i = 0;
i<MAXBULLET;i++) {

        bullet1[i].fired=false;

        bullet2[i].fired=false;

                        }
                        keybool[5]=false;
                }

                if (keybool[3]){
                        sChoice--;
                        if (sChoice<0)
sChoice=2;
                        for (int i = 0;
i<MAXBULLET;i++) {

        bullet1[i].fired=false;

        bullet2[i].fired=false;

                        }
                        keybool[3]=false;
                }


                // draw ShipSelection border
                g.drawImage(shipSelect, 0, 0,
g.TOP|g.LEFT);

                // draw Ships
                ships.choice(sChoice);
                ships.drawMenuShips(g);


                // draw bullet
                for (int i = 0;
i<MAXBULLET;i++) {

                        if(bullet1[i].fired){
                                if
(sChoice>=2) {

        bullet2[i].move(0,-5);

        bullet2[i].drawBullet(g);
                                }

        bullet1[i].move(0,-5);

        bullet1[i].drawBullet(g);

                        }

        bullet1[i].checkBound(true);

        bullet2[i].checkBound(true);
                }

                // choose which ship
                if (keybool[4]) {
                        if (sChoice == 0)
ships.shipChoice(0);
```

```
                    else if (sChoice == 1)
ships.shipChoice(1);
                    else if (sChoice == 2)
ships.shipChoice(2);


        ships.unloadOthers(ships.choice);
                    shipSelect=null;
                    loadGamePlay=false;
                    gameState=30;

                    for (int i = 0;
i<MAXBULLET;i++) {

        bullet1[i].fired=false;

                    }
                    keybool[4]=false;

                }

        }

        // game play
        private void gamePlay(Graphics g) {

                gameTicker++;


                // bikin starfield (background
belakang)
                g.setColor(255,255,255);
                for(int k=0;k<25;k++){

        g.drawLine(starfieldpos_W[k],starfieldpo
s_H[k],starfieldpos_W[k],starfieldpos_H[k]);

        g.drawLine(starfieldpos_W[6],starfieldpo
s_H[6],starfieldpos_W[6]+1,starfieldpos_H[6]);

        g.drawLine(starfieldpos_W[16],starfieldp
os_H[16],starfieldpos_W[16]+1,starfieldpos_H[16])
;

        g.drawLine(starfieldpos_W[22],starfieldp
os_H[22],starfieldpos_W[22]+1,starfieldpos_H[22])
;

        g.drawLine(starfieldpos_W[1],starfieldpo
s_H[1],starfieldpos_W[1]+1,starfieldpos_H[1]);

        g.drawLine(starfieldpos_W[13],starfieldp
os_H[13],starfieldpos_W[13]+1,starfieldpos_H[13])
;

        if(starfieldpos_H[k]<=screenH){

        if(k%5==0) starfieldpos_H[k]+=3;

        else if(k%2==0) starfieldpos_H[k]+=2;

        else starfieldpos_H[k]+=1;

        }else{starfieldpos_H[k]=0;}
                }

                // bikin asteroids
                if (gameLevel!=3 &&
gameLevel!=6){

                        // asteroid small
                        for(int
k=0;k<asteroidSmallNo;k++){
                                if
(gameTicker>50 && asteroidCount[k]<10) {

        asteroid[k].drawAsteroid(g);


        if(asteroid[k].temp_H<=screenH){

        asteroid[k].temp_H+=3;

                                }

        else{

        RandNum ranum1 = new RandNum();

        ranum1.genRandXY(getWidth(),getHeigh
t());

        asteroid[k].temp_W=ranum1.RANGE_X;

        asteroid[k].temp_H=-10;

        asteroidCount[k]++;
                                }
                        }


                                if
(ships.checkCollision(asteroid[k].getAsteroid_Pos
W()+1, asteroid[k].getAsteroid_PosH()+1,
7+asteroid[k].type*2, 7+asteroid[k].type*2) &&
!ships.afterBurn

        && !ships.player1Special && !ships.dead
&& !ships.hit && !ships.player2Special &&
!ships.player3Special) {

        ships.crash();

                                }

                        }

                        // asteroid med
                        for(int
k=0;k<asteroidMedNo;k++){
                                if
(gameTicker>50 && asteroidCountMed[k]<7) {

        asteroidMed[k].drawAsteroid(g);


        if(asteroidMed[k].temp_H<=screenH){

        asteroidMed[k].temp_H+=2;

                                }

        else{
```

```
            RandNum ranum1 = new RandNum();

            ranum1.genRandXY(getWidth(),getHeigh
t());

            asteroidMed[k].temp_W=ranum1.RANG
E_X;

            asteroidMed[k].temp_H=-12;

            asteroidCountMed[k]++;
                                                }
                                        }


                                        if
(ships.checkCollision(asteroidMed[k].getAsteroid_
PosW()+1, asteroidMed[k].getAsteroid_PosH()+1,
7+asteroidMed[k].type*2,
7+asteroidMed[k].type*2) && !ships.afterBurn

            && !ships.player1Special && !ships.dead
&& !ships.hit && !ships.player2Special &&
!ships.player3Special) {

            ships.crash();

                                        }

                                }

                                // asteroid big
                                for(int
k=0;k<asteroidBigNo;k++){
                                        if
(gameTicker>50 && asteroidCountBig[k]<4) {

            asteroidBig[k].drawAsteroid(g);


            if(asteroidBig[k].temp_H<=screenH){

            asteroidBig[k].temp_H+=1;

                                                }

            else{

            RandNum ranum1 = new RandNum();

            ranum1.genRandXY(getWidth(),getHeigh
t());

            asteroidBig[k].temp_W=ranum1.RANGE
_X;

            asteroidBig[k].temp_H=-14;

            asteroidCountBig[k]++;
                                                }
                                        }


                                        if
(ships.checkCollision(asteroidBig[k].getAsteroid_P
osW()+1, asteroidBig[k].getAsteroid_PosH()+1,
7+asteroidBig[k].type*2, 7+asteroidBig[k].type*2)
&& !ships.afterBurn

            && !ships.player1Special && !ships.dead
&& !ships.hit && !ships.player2Special &&
!ships.player3Special) {

            ships.crash();

                                                }

                                        }
                                }

                                // enemy (keluar satu persatu)
                                if (gameTicker>50 &&
gameTicker<1900){
                                        enemyTick++;
                                        if (enemyTick==10)
enemy[0].out();
                                        if (enemyTick==30)
enemy[1].out();
                                        if (enemyTick==50)
enemy[2].out();
                                        if (enemyTick==20)
enemy[3].out();
                                        if (enemyTick==40)
enemy[4].out();
                                        if (enemyTick==60)
enemy[5].out();
                                }


                                // bikin enemy
                                for (int y=0; y<6; y++)         {


            g.setClip(0,0,screenW, screenH);
                                        if (!enemy[y].dead
&& enemy[y].out) {
                                                if
(enemyCounter%6==3) {

                                                        if
(y<enemyNo/2){

            enemy[y].Horizontal(g,0,enemySel1);


                                                        }

            else {

            enemy[y].Horizontal(g,3,enemySel1);


                                                        }

            if(gameLevel!=1 && gameLevel!=4){


            if (enemy[y].getEnemy_PosW()==30 ||
enemy[y].getEnemy_PosW()==135 ||
enemy[y].getEnemy_PosW()==31 ||
enemy[y].getEnemy_PosW()==136){ //||
enemy[y].getEnemy_PosW()==47) {
```

```
            enemyBullet[k].updateEnemy(ships.getX(
), ships.getY(),
enemy[y].getEnemy_PosW(),enemy[y].getEnemy_
PosH());

                        k++;

                        k%=enemyBulletNo;

            }
                                                }

                                        }
                                        else if
(enemyCounter%6==2) {
                                                if
(powerTaken == true) powerTaken=false;
                                                if
(y<enemyNo/2){

            enemy[y].Horizontal(g,3,enemySel1);
                                                }

            else {

            enemy[y].Horizontal(g,0,enemySel1);
                                                }

            if(gameLevel!=1 && gameLevel!=4){

            if (enemy[y].getEnemy_PosW()==30 ||
enemy[y].getEnemy_PosW()==135 ||
enemy[y].getEnemy_PosW()==31 ||
enemy[y].getEnemy_PosW()==136) {


            enemyBullet[k].updateEnemy(ships.getX(
), ships.getY(),
enemy[y].getEnemy_PosW(),enemy[y].getEnemy_
PosH());

                        k++;

                        k%=enemyBulletNo;

            }
                                                }
                                        }
                                        else if
(enemyCounter%6==1) {

                                                if
(y<enemyNo/2) {

            enemy[y].Vertical(g,3,enemySel2);
                                                }

            else {

            if (y==4) enemy[y].Vertical(g,0,4);

            else {


            enemy[y].Vertical(g,0,enemySel2);
            }


                                                }

            if(gameLevel!=1 && gameLevel!=4){

            if (enemy[y].getEnemy_PosH()==30 ||
enemy[y].getEnemy_PosH()==150) {


            enemyBullet[k].updateEnemy(ships.getX(
), ships.getY(),
enemy[y].getEnemy_PosW(),enemy[y].getEnemy_
PosH());

                        k++;

                                k%=enemyBulletNo;

            }
                                                }
                                                if
(enemy[4].dead && !power && !powerTaken){

            powerUp(enemy[4].getEnemy_PosW(),en
emy[4].getEnemy_PosH()-20);

                                                }
                                        }
                                        else if
(enemyCounter%6==5) {
                                                if
(powerTaken == true) powerTaken=false;
                                                if
(y<enemyNo/2){

            enemy[y].Vertical(g,0,enemySel2);
                                                }

            else {

            enemy[y].Vertical(g,3,enemySel2);
                                                }

            if(gameLevel!=1 && gameLevel!=4){

            if (enemy[y].getEnemy_PosH()==30 ||
enemy[y].getEnemy_PosH()==150) {


            enemyBullet[k].updateEnemy(ships.getX(
), ships.getY(),
enemy[y].getEnemy_PosW(),enemy[y].getEnemy_
PosH());

                        k++;

                        k%=enemyBulletNo;

            }
                                                }
                                        }
```

```
                                        else if
(enemyCounter%6==4) {
                                                if
(y<enemyNo/2){

            enemy[y].Zigzag(g,3,enemySel1);
                                                }

            else {

            if (y==4) enemy[y].Zigzag(g,0,4);

            else {



            enemy[y].Zigzag(g,0,enemySel1);



            }
                                                }

            if(gameLevel!=1 && gameLevel!=4){

            if (enemy[y].getEnemy_PosH()==32 ||
enemy[y].getEnemy_PosH()==152) {


            enemyBullet[k].updateEnemy(ships.getX(
), ships.getY(),
enemy[y].getEnemy_PosW(),enemy[y].getEnemy_
PosH());

                        k++;

                        k%=enemyBulletNo;

            }
                                                }
                                                if
(enemy[4].dead && !power && !powerTaken){

            powerUp(enemy[4].getEnemy_PosW(),en
emy[4].getEnemy_PosH()-20);

                                                }
                                        }
                                        else if
(enemyCounter%6==0) {
                                                if
(y<enemyNo/2){

            enemy[y].Zigzag(g,0,enemySel2);
                                                }

            else {

            enemy[y].Zigzag(g,3,enemySel2);
                                                }

            if(gameLevel!=1 && gameLevel!=4){

            if (enemy[y].getEnemy_PosH()==32 ||
enemy[y].getEnemy_PosH()==152) {


            enemyBullet[k].updateEnemy(ships.getX(
), ships.getY(),
enemy[y].getEnemy_PosW(),enemy[y].getEnemy_
PosH());

                        k++;

                        k%=enemyBulletNo;

            }
                                                }
                                        }

                                }

                        }

                        // check  ships ketemu
enemy
                        if
(ships.checkCollision(enemy[y].getEnemy_PosW(),
enemy[y].getEnemy_PosH(),11, 11) &&
!ships.afterBurn && !ships.player1Special &&
!ships.dead && !ships.hit && !ships.gameOver &&
!ships.player2Special && !ships.player3Special &&
!enemy[y].hit && !enemy[y].dead) {

            ships.crash();

            enemy[y].crash();

            //enemy[y].hit();
                                    score+=10;

                        }

                        // check  ships3
special ketemu enemy
                        if
(enemy[y].checkCollision(ships.getX()+5,ships.lase
rY-114,5,114) && !enemy[y].dead &&
!enemy[y].hit && ships.player3Special ) {

            enemy[y].hit();
                                    score+=10;

                        }
                        // check  ships1
special ketemu enemy
                        if
(enemy[y].checkCollision(ships.getX()-
2,ships.getY()+1,19,24) && !enemy[y].dead &&
!enemy[y].hit && ships.player1Special ) {

            enemy[y].hit();
                                    score+=10;

                        }
                        // check  ships2
special ketemu enemy
                        if
(enemy[y].checkCollision(0,ships.getSpclY(),screen
W,45) && !enemy[y].dead && !enemy[y].hit &&
ships.player2Special) {

            enemy[y].hit();
                                    score+=10;
                        }
```

```
                    }

                    // enemy bullet
                    g.setClip(0,0,screenW,
screenH);
                    for (int j=0;
j<enemyBulletNo;j++) {
                              if
(enemyBullet[j].fired){

          enemyBullet[j].moveEnemy();

          enemyBullet[j].drawBulletBoss(g,(short)e
nemyBullet[j].enemyX(),(short)enemyBullet[j].ene
myY(), (byte)0);
                                        if
(enemyBullet[j].checkCollisionEnemy(ships.getX()
+1,(ships.getY()-enemyBullet[j].getY()-
ships.specialTicker*5),5,160) &&
ships.player3Special ) {

          enemyBullet[j].fired=false;



                                        }
                                        if
(enemyBullet[j].checkCollisionEnemy(0,ships.getS
pclY(),screenW,50) && ships.player2Special) {


          enemyBullet[j].fired=false;
                                        }
                                        if
(enemyBullet[j].checkCollisionEnemy(ships.getX()
+5,ships.laserY-114,5,114) &&
ships.player1Special) {


          enemyBullet[j].fired=false;
                                        }
                                        if
(ships.checkCollision(enemyBullet[j].enemyX(),
enemyBullet[j].enemyY(),1, 1) && !ships.afterBurn
&& !ships.player1Special && !ships.dead &&
!ships.hit && !ships.player2Special &&
!ships.player3Special && !ships.gameOver)
ships.hit(enemyBullet[j].enemyY());

                              }



          enemyBullet[j].checkBoundEnemy();

                    }

                    // shooting ships bullets if
getting power
                    if (!ships.dead &&
!ships.gameOver) {
                              if
(!ships.player3Special) {
                                        if
(ships.choice==1 || ships.choice==2){

          if(ships.frame()==1 || ships.frame()==2 ||
ships.frame()==4 || ships.frame()==5 ||
ships.frame()==7 || ships.frame()==8) {

               if (ships.choice==1) {

                    if (ships.power==0){


          bullet1[i].updateBullet(ships.getX()+4,
ships.getY());

                    }

                    else if (ships.power==1){


          bullet1[i].updateBullet(ships.getX()-2,
ships.getY());


          bullet2[i].updateBullet(ships.getX()+10,
ships.getY());

                    }

                    else {



          bullet1[i].updateBullet(ships.getX()-2,
ships.getY());


          bullet2[i].updateBullet(ships.getX()+10,
ships.getY());


          bullet3[i].updateBullet(ships.getX()-2,
ships.getY()+20);


          bullet4[i].updateBullet(ships.getX()+10,
ships.getY()+20);

                    }

               }

               else if (ships.choice==2) {

                    if (ships.power==0){


          bullet1[i].updateBullet(ships.getX()+4,
ships.getY());

                    }

                    else if (ships.power==1) {


          bullet1[i].updateBullet(ships.getX(),
ships.getY());


          bullet2[i].updateBullet(ships.getX()+8,
ships.getY());
```

```
                    }

                    else {


            bullet1[i].updateBullet(ships.getX(),
ships.getY());


            bullet2[i].updateBullet(ships.getX()+8,
ships.getY());


            bullet3[i].updateBullet(ships.getX()-8,
ships.getY()+3);


            bullet4[i].updateBullet(ships.getX()+16,
ships.getY()+3);

                    }

            }
                                        }
                                        else {

            if(ships.frame()==1 || ships.frame()==3 ||
ships.frame()==5) {

            if (ships.power==0){


            bullet1[i].updateBullet(ships.getX()+2,
ships.getY());


            bullet2[i].updateBullet(ships.getX()+6,
ships.getY());

            }

            else if (ships.power==0) {


            bullet1[i].updateBullet(ships.getX()+2,
ships.getY());


            bullet2[i].updateBullet(ships.getX()+6,
ships.getY());


            bullet3[i].updateBullet(ships.getX()+2,
ships.getY());


            bullet4[i].updateBullet(ships.getX()+7,
ships.getY());

            }

            else {


            bullet2[i].updateBullet(ships.getX()+6,
ships.getY());


            bullet1[i].updateBullet(ships.getX()+2,
ships.getY());


            bullet3[i].updateBullet(ships.getX()+2,
ships.getY());


            bullet5[i].updateBullet(ships.getX()+2,
ships.getY());


            bullet4[i].updateBullet(ships.getX()+7,
ships.getY());


            bullet6[i].updateBullet(ships.getX()+7,
ships.getY());


            }
                                        }
                                    }
                        }

                        // bullet checking collision

                        for (int i = 0;
i<MAXBULLET;i++) {
                                if (ships.choice==1) {
                                    if
(ships.power==0){
                                        if
(bullet1[i].fired){

            bullet1[i].move(0,-5);

            bullet1[i].drawBullet(g);

            if
(boss.checkCollision(bullet1[i].getX(),bullet1[i].get
Y(),2,3) && !boss.dead && !boss.comingDown
&& boss.getY()>=10) {

                        boss.hit();

                        bullet1[i].fired=false;

                        score+=10;

                        playSound();



            }

            for (int x = 0; x<enemyNo; x++) {

                        if
(enemy[x].checkCollision(bullet1[i].getX(),bullet1[i
```

```
].getY(),2,3) && !enemy[x].dead &&
!enemy[x].hit) {

                                enemy[x].hit();

                                bullet1[i].fired=false;

                                score+=10;

                                playSound();

                                //sound.play(1);

                        }

                }

                if(gameLevel!=3 && gameLevel!=6){

                        for (int x = 0;
x<asteroidSmallNo; x++) {

                                if
(asteroid[x].checkCollision(bullet1[i].getX(),bullet1
[i].getY(),2,3)) {


                asteroid[x].hit();


                bullet1[i].fired=false;

                                }


                        }

                        for (int x = 0;
x<asteroidMedNo; x++) {

                                if
(asteroidMed[x].checkCollision(bullet1[i].getX(),bu
llet1[i].getY(),2,3)) {


                asteroidMed[x].hit();


                bullet1[i].fired=false;

                                }


                        }

                        for (int x = 0; x<asteroidBigNo;
x++) {

                                if
(asteroidBig[x].checkCollision(bullet1[i].getX(),bull
et1[i].getY(),2,3)) {


                asteroidBig[x].hit();


                bullet1[i].fired=false;
                                        }

                                }

                        }

                }

                                        }
                                }
                                else if
(ships.power==1) {
                                        if
(bullet1[i].fired){

                bullet1[i].move(0,-5);

                bullet1[i].drawBullet(g);

                if
(boss.checkCollision(bullet1[i].getX(),bullet1[i].get
Y(),2,3) && !boss.dead && !boss.comingDown
&& boss.getY()>=10) {

                        boss.hit();

                        bullet1[i].fired=false;

                        score+=10;

                        playSound();

                        //sound.play(1);

                }

                for (int x = 0; x<enemyNo; x++) {

                        if
(enemy[x].checkCollision(bullet1[i].getX(),bullet1[i
].getY(),2,3) && !enemy[x].dead &&
!enemy[x].hit) {

                                enemy[x].hit();


                                bullet1[i].fired=false;

                                score+=10;

                                playSound();

                                //sound.play(1);

                        }

                }

                if(gameLevel!=3 && gameLevel!=6){

                        for (int x = 0;
x<asteroidSmallNo; x++) {

                                if
(asteroid[x].checkCollision(bullet1[i].getX(),bullet1
[i].getY(),2,3)) {
```

```
            asteroid[x].hit();

            bullet1[i].fired=false;
                        }

                }

                for (int x = 0;
x<asteroidMedNo; x++) {
                        if
(asteroidMed[x].checkCollision(bullet1[i].getX(),bu
llet1[i].getY(),2,3)) {

            asteroidMed[x].hit();

            bullet1[i].fired=false;
                        }

                }

                for (int x = 0; x<asteroidBigNo;
x++) {
                        if
(asteroidBig[x].checkCollision(bullet1[i].getX(),bull
et1[i].getY(),2,3)) {

            asteroidBig[x].hit();

            bullet1[i].fired=false;
                        }

                }
            }
                                                    }
                                                    if
(bullet2[i].fired){
            bullet2[i].move(0,-5);
            bullet2[i].drawBullet(g);
            if
(boss.checkCollision(bullet2[i].getX(),bullet2[i].get
Y(),2,3) && !boss.dead && !boss.comingDown
&& boss.getY()>=10) {
                boss.hit();
                bullet2[i].fired=false;
                score+=10;
                playSound();
                //sound.play(1);
            }
            for (int x = 0; x<enemyNo; x++) {
                if
(enemy[x].checkCollision(bullet2[i].getX(),bullet2[i
].getY(),2,3) && !enemy[x].dead &&
!enemy[x].hit) {
                        enemy[x].hit();
                        bullet2[i].fired=false;
                        score+=10;
                        playSound();
                        //sound.play(1);
                }
            }
            if(gameLevel!=3 && gameLevel!=6){
                for (int x = 0;
x<asteroidSmallNo; x++) {
                        if
(asteroid[x].checkCollision(bullet1[i].getX(),bullet1
[i].getY(),2,3)) {

            asteroid[x].hit();

            bullet2[i].fired=false;
                        }

                }
                for (int x = 0;
x<asteroidMedNo; x++) {
                        if
(asteroidMed[x].checkCollision(bullet2[i].getX(),bu
llet2[i].getY(),2,3)) {

            asteroidMed[x].hit();

            bullet2[i].fired=false;
                        }

                }
```

```
                    for (int x = 0; x<asteroidBigNo;
x++) {
                        if
(asteroidBig[x].checkCollision(bullet2[i].getX(),bull
et2[i].getY(),2,3)) {

            asteroidBig[x].hit();

            bullet2[i].fired=false;
                        }

                    }
            }
                                    }
                                }
                                else {
                                    if
(bullet1[i].fired){

            bullet1[i].move(0,-5);

            bullet1[i].drawBullet(g);

            if
(boss.checkCollision(bullet1[i].getX(),bullet1[i].get
Y(),2,3) && !boss.dead && !boss.comingDown
&& boss.getY()>=10) {

                    boss.hit();

                    bullet1[i].fired=false;

                    score+=10;

                    playSound();

                    //sound.play(1);

            }

            for (int x = 0; x<enemyNo; x++) {

                    if
(enemy[x].checkCollision(bullet1[i].getX(),bullet1[i
].getY(),2,3) && !enemy[x].dead &&
!enemy[x].hit) {
                        enemy[x].hit();

                        bullet1[i].fired=false;

                        score+=10;

                        playSound();

                        //sound.play(1);

                    }

            }

            if(gameLevel!=3 && gameLevel!=6){

                    for (int x = 0;
x<asteroidSmallNo; x++) {
                        if
(asteroid[x].checkCollision(bullet1[i].getX(),bullet1
[i].getY(),2,3)) {

            asteroid[x].hit();

            bullet1[i].fired=false;
                        }

                    }

                    for (int x = 0;
x<asteroidMedNo; x++) {
                        if
(asteroidMed[x].checkCollision(bullet1[i].getX(),bu
llet1[i].getY(),2,3)) {

            asteroidMed[x].hit();

            bullet1[i].fired=false;
                        }

                    }

                    for (int x = 0; x<asteroidBigNo;
x++) {
                        if
(asteroidBig[x].checkCollision(bullet1[i].getX(),bull
et1[i].getY(),2,3)) {

            asteroidBig[x].hit();

            bullet1[i].fired=false;
                        }

                    }
            }
                                    }
                                    if
(bullet2[i].fired){

            bullet2[i].move(0,-5);

            bullet2[i].drawBullet(g);
```

```
            if
(boss.checkCollision(bullet2[i].getX(),bullet2[i].get
Y(),2,3) && !boss.dead && !boss.comingDown
&& boss.getY()>=10) {

                        boss.hit();

                        bullet2[i].fired=false;

                        score+=10;

                        playSound();

                        //sound.play(1);

            }

            for (int x = 0; x<enemyNo; x++) {

                        if
(enemy[x].checkCollision(bullet2[i].getX(),bullet2[i
].getY(),2,3) && !enemy[x].dead &&
!enemy[x].hit) {

                                    enemy[x].hit();

                                    bullet2[i].fired=false;

                                    score+=10;

                                    playSound();

                                    //sound.play(1);

                        }

            }

            if(gameLevel!=3 && gameLevel!=6){

                        for (int x = 0;
x<asteroidSmallNo; x++) {

                                    if
(asteroid[x].checkCollision(bullet2[i].getX(),bullet2
[i].getY(),2,3)) {


            asteroid[x].hit();


            bullet2[i].fired=false;

                                    }


                        }

                        for (int x = 0;
x<asteroidMedNo; x++) {

                                    if
(asteroidMed[x].checkCollision(bullet2[i].getX(),bu
llet2[i].getY(),2,3)) {


            asteroidMed[x].hit();


            bullet2[i].fired=false;

                                    }


                        }

                        for (int x = 0; x<asteroidBigNo;
x++) {

                                    if
(asteroidBig[x].checkCollision(bullet2[i].getX(),bull
et2[i].getY(),2,3)) {


            asteroidBig[x].hit();


            bullet2[i].fired=false;

                                    }


                        }

            }
                                                            }
                                                            if
(bullet3[i].fired){

            bullet3[i].move(0,5);

            bullet3[i].drawBullet(g);

            if
(boss.checkCollision(bullet3[i].getX(),bullet3[i].get
Y(),2,3) && !boss.dead && !boss.comingDown
&& boss.getY()>=10) {

                        boss.hit();

                        bullet3[i].fired=false;

                        score+=10;

                        playSound();

                        //sound.play(1);

            }

            for (int x = 0; x<enemyNo; x++) {

                        if
(enemy[x].checkCollision(bullet3[i].getX(),bullet3[i
].getY(),2,3) && !enemy[x].dead &&
!enemy[x].hit) {

                                    enemy[x].hit();

                                    bullet3[i].fired=false;
```

```
                    score+=10;

                    playSound();

                    //sound.play(1);

                }

            }

            if(gameLevel!=3 && gameLevel!=6){

                for (int x = 0;
x<asteroidSmallNo; x++) {

                    if
(asteroid[x].checkCollision(bullet3[i].getX(),bullet3
[i].getY(),2,3)) {


            asteroid[x].hit();


            bullet3[i].fired=false;

                    }


                }

                for (int x = 0;
x<asteroidMedNo; x++) {

                    if
(asteroidMed[x].checkCollision(bullet3[i].getX(),bu
llet3[i].getY(),2,3)) {


            asteroidMed[x].hit();


            bullet3[i].fired=false;

                    }


                }

                for (int x = 0; x<asteroidBigNo;
x++) {

                    if
(asteroidBig[x].checkCollision(bullet3[i].getX(),bull
et3[i].getY(),2,3)) {


            asteroidBig[x].hit();


            bullet3[i].fired=false;

                    }


                }
            }
                                                    }
                                                    if
(bullet4[i].fired){

            bullet4[i].move(0,5);

            bullet4[i].drawBullet(g);

            if
(boss.checkCollision(bullet4[i].getX(),bullet4[i].get
Y(),2,3) && !boss.dead && !boss.comingDown
&& boss.getY()>=10) {

                boss.hit();

                bullet4[i].fired=false;

                score+=10;

                playSound();

                //sound.play(1);

            }

            for (int x = 0; x<enemyNo; x++) {

                if
(enemy[x].checkCollision(bullet4[i].getX(),bullet4[i
].getY(),2,3) && !enemy[x].dead &&
!enemy[x].hit) {

                    enemy[x].hit();

                    bullet4[i].fired=false;

                    score+=10;

                    playSound();

                    //sound.play(1);

                }

            }

            if(gameLevel!=3 && gameLevel!=6){

                for (int x = 0;
x<asteroidSmallNo; x++) {

                    if
(asteroid[x].checkCollision(bullet4[i].getX(),bullet4
[i].getY(),2,3)) {


            asteroid[x].hit();


            bullet4[i].fired=false;

                    }


                }
```

```
                    for (int x = 0;
x<asteroidMedNo; x++) {

                        if
(asteroidMed[x].checkCollision(bullet4[i].getX(),bu
llet4[i].getY(),2,3)) {


            asteroidMed[x].hit();


            bullet4[i].fired=false;

                        }


                    }

                    for (int x = 0; x<asteroidBigNo;
x++) {

                        if
(asteroidBig[x].checkCollision(bullet4[i].getX(),bull
et4[i].getY(),2,3)) {


            asteroidBig[x].hit();


            bullet4[i].fired=false;

                        }


                    }

            }
                                                    }
                                                }

                        }
                        else if
(ships.choice==2) {
                            if
(ships.power==0){
                                                    if
(bullet1[i].fired){

            bullet1[i].move(0,-5);

            bullet1[i].drawBullet(g);

            if
(boss.checkCollision(bullet1[i].getX(),bullet1[i].get
Y(),2,3) && !boss.dead && !boss.comingDown
&& boss.getY()>=10) {

                    boss.hit();

                    bullet1[i].fired=false;

                    score+=10;

                    playSound();

                    //sound.play(1);

            }

            for (int x = 0; x<enemyNo; x++) {

                    if
(enemy[x].checkCollision(bullet1[i].getX(),bullet1[i
].getY(),2,3) && !enemy[x].dead &&
!enemy[x].hit) {

                        enemy[x].hit();

                        bullet1[i].fired=false;

                        score+=10;

                        playSound();

                        //sound.play(1);

                    }

            }

            if(gameLevel!=3 && gameLevel!=6){

                    for (int x = 0;
x<asteroidSmallNo; x++) {

                        if
(asteroid[x].checkCollision(bullet1[i].getX(),bullet1
[i].getY(),2,3)) {


            asteroid[x].hit();


            bullet1[i].fired=false;

                        }


                    }

                    for (int x = 0;
x<asteroidMedNo; x++) {

                        if
(asteroidMed[x].checkCollision(bullet1[i].getX(),bu
llet1[i].getY(),2,3)) {


            asteroidMed[x].hit();


            bullet1[i].fired=false;

                        }


                    }

                    for (int x = 0; x<asteroidBigNo;
x++) {

                        if
```

```
(asteroidBig[x].checkCollision(bullet1[i].getX(),bull
et1[i].getY(),2,3)) {

            asteroidBig[x].hit();

            bullet1[i].fired=false;

                        }

                }

        }
                                                }
                                        }
                                        else if
(ships.power==1) {
                                                if
(bullet1[i].fired){

        bullet1[i].move(0,-5);

        bullet1[i].drawBullet(g);

        if
(boss.checkCollision(bullet1[i].getX(),bullet1[i].get
Y(),2,3) && !boss.dead && !boss.comingDown
&& boss.getY()>=10) {

                boss.hit();

                bullet1[i].fired=false;

                score+=10;

                playSound();

                //sound.play(1);

        }

        for (int x = 0; x<enemyNo; x++) {

                if
(enemy[x].checkCollision(bullet1[i].getX(),bullet1[i
].getY(),2,3) && !enemy[x].dead &&
!enemy[x].hit) {

                        enemy[x].hit();

                        bullet1[i].fired=false;

                        score+=10;

                        playSound();

                        //sound.play(1);

                }

        }

        if(gameLevel!=3 && gameLevel!=6){

                        for (int x = 0;
x<asteroidSmallNo; x++) {

                                if
(asteroid[x].checkCollision(bullet1[i].getX(),bullet1
[i].getY(),2,3)) {

            asteroid[x].hit();

            bullet1[i].fired=false;

                                }

                        }

                        for (int x = 0;
x<asteroidMedNo; x++) {

                                if
(asteroidMed[x].checkCollision(bullet1[i].getX(),bu
llet1[i].getY(),2,3)) {

            asteroidMed[x].hit();

            bullet1[i].fired=false;

                                }

                        }

                        for (int x = 0; x<asteroidBigNo;
x++) {

                                if
(asteroidBig[x].checkCollision(bullet1[i].getX(),bull
et1[i].getY(),2,3)) {

            asteroidBig[x].hit();

            bullet1[i].fired=false;

                                }

                        }

        }
                                                }
                                                if
(bullet2[i].fired){

        bullet2[i].move(0,-5);

        bullet2[i].drawBullet(g);

        if
(boss.checkCollision(bullet2[i].getX(),bullet2[i].get
```

```
Y(),2,3) && !boss.dead && !boss.comingDown
&& boss.getY()>=10) {

                    boss.hit();

                    bullet2[i].fired=false;

                    score+=10;

                    playSound();

                    //sound.play(1);

          }

          for (int x = 0; x<enemyNo; x++) {

                    if
(enemy[x].checkCollision(bullet2[i].getX(),bullet2[i
].getY(),2,3) && !enemy[x].dead &&
!enemy[x].hit) {

                              enemy[x].hit();

                              bullet2[i].fired=false;

                              score+=10;

                              playSound();

                              //sound.play(1);

                    }

          }

          if(gameLevel!=3 && gameLevel!=6){

                    for (int x = 0;
x<asteroidSmallNo; x++) {

                              if
(asteroid[x].checkCollision(bullet2[i].getX(),bullet2
[i].getY(),2,3)) {


          asteroid[x].hit();


          bullet2[i].fired=false;

                              }


                    }

                    for (int x = 0;
x<asteroidMedNo; x++) {

                              if
(asteroidMed[x].checkCollision(bullet2[i].getX(),bu
llet2[i].getY(),2,3)) {


          asteroidMed[x].hit();


          bullet2[i].fired=false;

                              }


                    }

                    for (int x = 0; x<asteroidBigNo;
x++) {

                              if
(asteroidBig[x].checkCollision(bullet2[i].getX(),bull
et2[i].getY(),2,3)) {


          asteroidBig[x].hit();


          bullet2[i].fired=false;

                              }


                    }

          }
                                                  }
                                        }
                                        else {
                                                  if
(bullet1[i].fired){

          bullet1[i].move(0,-5);

          bullet1[i].drawBullet(g);

          if
(boss.checkCollision(bullet1[i].getX(),bullet1[i].get
Y(),2,3) && !boss.dead && !boss.comingDown
&& boss.getY()>=10) {

                    boss.hit();

                    bullet1[i].fired=false;

                    score+=10;

                    playSound();

                    //sound.play(1);

          }

          for (int x = 0; x<enemyNo; x++) {

                    if
(enemy[x].checkCollision(bullet1[i].getX(),bullet1[i
].getY(),2,3) && !enemy[x].dead &&
!enemy[x].hit) {

                              enemy[x].hit();

                              bullet1[i].fired=false;

                              score+=10;
```

```
                        playSound();

                        //sound.play(1);

                    }

                }

                if(gameLevel!=3 && gameLevel!=6){

                    for (int x = 0;
x<asteroidSmallNo; x++) {

                        if
(asteroid[x].checkCollision(bullet1[i].getX(),bullet1
[i].getY(),2,3)) {


                asteroid[x].hit();


                bullet1[i].fired=false;

                        }


                    }

                    for (int x = 0;
x<asteroidMedNo; x++) {

                        if
(asteroidMed[x].checkCollision(bullet1[i].getX(),bu
llet1[i].getY(),2,3)) {


                asteroidMed[x].hit();


                bullet1[i].fired=false;

                        }


                    }

                    for (int x = 0; x<asteroidBigNo;
x++) {

                        if
(asteroidBig[x].checkCollision(bullet1[i].getX(),bull
et1[i].getY(),2,3)) {


                asteroidBig[x].hit();


                bullet1[i].fired=false;

                        }


                    }

                }
                                                    }
                                                        if
(bullet2[i].fired){

                bullet2[i].move(0,-5);

                bullet2[i].drawBullet(g);

                if
(boss.checkCollision(bullet2[i].getX(),bullet2[i].get
Y(),2,3) && !boss.dead && !boss.comingDown
&& boss.getY()>=10) {

                    boss.hit();

                    bullet2[i].fired=false;

                    score+=10;

                    playSound();

                    //sound.play(1);

                }

                for (int x = 0; x<enemyNo; x++) {

                    if
(enemy[x].checkCollision(bullet2[i].getX(),bullet2[i
].getY(),2,3) && !enemy[x].dead &&
!enemy[x].hit) {

                        enemy[x].hit();

                        bullet2[i].fired=false;

                        score+=10;

                        playSound();

                        //sound.play(1);

                    }

                }

                if(gameLevel!=3 && gameLevel!=6){

                    for (int x = 0;
x<asteroidSmallNo; x++) {

                        if
(asteroid[x].checkCollision(bullet2[i].getX(),bullet2
[i].getY(),2,3)) {


                asteroid[x].hit();


                bullet2[i].fired=false;

                        }


                    }
```

```
                    for (int x = 0;
x<asteroidMedNo; x++) {

                        if
(asteroidMed[x].checkCollision(bullet2[i].getX(),bu
llet2[i].getY(),2,3)) {


            asteroidMed[x].hit();


            bullet2[i].fired=false;

                        }


                    }

                    for (int x = 0; x<asteroidBigNo;
x++) {

                        if
(asteroidBig[x].checkCollision(bullet2[i].getX(),bull
et2[i].getY(),2,3)) {


            asteroidBig[x].hit();


            bullet2[i].fired=false;

                        }


                    }

            }
                                                }
                                                if
(bullet3[i].fired){

            bullet3[i].move(0,-5);

            bullet3[i].drawBullet(g);

            if
(boss.checkCollision(bullet3[i].getX(),bullet3[i].get
Y(),2,3) && !boss.dead && !boss.comingDown
&& boss.getY()>=10) {

                    boss.hit();

                    bullet3[i].fired=false;

                    score+=10;

                    playSound();

                    //sound.play(1);

            }

            for (int x = 0; x<enemyNo; x++) {

                    if
(enemy[x].checkCollision(bullet3[i].getX(),bullet3[i
].getY(),2,3) && !enemy[x].dead &&
!enemy[x].hit) {

                        enemy[x].hit();

                        bullet3[i].fired=false;

                        score+=10;

                        playSound();

                        //sound.play(1);

                }

            }

            if(gameLevel!=3 && gameLevel!=6){

                    for (int x = 0;
x<asteroidSmallNo; x++) {

                        if
(asteroid[x].checkCollision(bullet3[i].getX(),bullet3
[i].getY(),2,3)) {


            asteroid[x].hit();


            bullet3[i].fired=false;

                        }


                    }

                    for (int x = 0;
x<asteroidMedNo; x++) {

                        if
(asteroidMed[x].checkCollision(bullet3[i].getX(),bu
llet3[i].getY(),2,3)) {


            asteroidMed[x].hit();


            bullet3[i].fired=false;

                        }


                    }

                    for (int x = 0; x<asteroidBigNo;
x++) {

                        if
(asteroidBig[x].checkCollision(bullet3[i].getX(),bull
et3[i].getY(),2,3)) {


            asteroidBig[x].hit();


            bullet3[i].fired=false;
```

```
                        }


                    }

                }
                                                    }
                                                    if
(bullet4[i].fired){

                bullet4[i].move(0,-5);

                bullet4[i].drawBullet(g);

                if
(boss.checkCollision(bullet4[i].getX(),bullet4[i].get
Y(),2,3) && !boss.dead && !boss.comingDown
&& boss.getY()>=10) {

                    boss.hit();

                    bullet4[i].fired=false;

                    score+=10;

                    playSound();

                    //sound.play(1);

                }

                for (int x = 0; x<enemyNo; x++) {

                    if
(enemy[x].checkCollision(bullet4[i].getX(),bullet4[i
].getY(),2,3) && !enemy[x].dead &&
!enemy[x].hit) {

                        enemy[x].hit();

                        bullet4[i].fired=false;

                        score+=10;

                        playSound();

                        //sound.play(1);

                    }

                }

                if(gameLevel!=3 && gameLevel!=6){

                    for (int x = 0;
x<asteroidSmallNo; x++) {

                        if
(asteroid[x].checkCollision(bullet4[i].getX(),bullet4
[i].getY(),2,3)) {


                asteroid[x].hit();


                bullet4[i].fired=false;
                        }


                    }

                    for (int x = 0;
x<asteroidMedNo; x++) {

                        if
(asteroidMed[x].checkCollision(bullet4[i].getX(),bu
llet4[i].getY(),2,3)) {


                asteroidMed[x].hit();


                bullet4[i].fired=false;

                        }


                    }

                    for (int x = 0; x<asteroidBigNo;
x++) {

                        if
(asteroidBig[x].checkCollision(bullet4[i].getX(),bull
et4[i].getY(),2,3)) {


                asteroidBig[x].hit();


                bullet4[i].fired=false;

                        }


                    }

                }
                                                        }
                                                    }
                                                }
                                                else {
                                                    if
(ships.power==0) {
                                                        if
(bullet1[i].fired){

                bullet1[i].move(0,-5);

                bullet1[i].drawBullet(g);

                if
(boss.checkCollision(bullet1[i].getX(),bullet1[i].get
Y(),2,3) && !boss.dead && !boss.comingDown
&& boss.getY()>=10) {

                    boss.hit();

                    bullet1[i].fired=false;

                    score+=10;
```

```
                    playSound();

                    //sound.play(1);

            }

            for (int x = 0; x<enemyNo; x++) {

                    if
(enemy[x].checkCollision(bullet1[i].getX(),bullet1[i
].getY(),2,3) && !enemy[x].dead &&
!enemy[x].hit) {

                            enemy[x].hit();

                            bullet1[i].fired=false;

                            score+=10;

                            playSound();

                            //sound.play(1);

                    }

            }

            if(gameLevel!=3 && gameLevel!=6){

                    for (int x = 0;
x<asteroidSmallNo; x++) {

                            if
(asteroid[x].checkCollision(bullet1[i].getX(),bullet1
[i].getY(),2,3)) {


            asteroid[x].hit();


            bullet1[i].fired=false;

                            }


                    }

                    for (int x = 0;
x<asteroidMedNo; x++) {

                            if
(asteroidMed[x].checkCollision(bullet1[i].getX(),bu
llet1[i].getY(),2,3)) {


            asteroidMed[x].hit();


            bullet1[i].fired=false;

                            }


                    }

                    for (int x = 0; x<asteroidBigNo;
x++) {

                            if
(asteroidBig[x].checkCollision(bullet1[i].getX(),bull
et1[i].getY(),2,3)) {


            asteroidBig[x].hit();


            bullet1[i].fired=false;

                            }


                    }

            }
                                                    }
                                                    if
(bullet2[i].fired){

            bullet2[i].move(0,-5);

            bullet2[i].drawBullet(g);

            if
(boss.checkCollision(bullet2[i].getX(),bullet2[i].get
Y(),2,3) && !boss.dead && !boss.comingDown
&& boss.getY()>=10) {

                    boss.hit();

                    bullet2[i].fired=false;

                    score+=10;

                    playSound();

                    //sound.play(1);

            }

            for (int x = 0; x<enemyNo; x++) {

                    if
(enemy[x].checkCollision(bullet2[i].getX(),bullet2[i
].getY(),2,3) && !enemy[x].dead &&
!enemy[x].hit) {

                            enemy[x].hit();

                            bullet2[i].fired=false;

                            score+=10;

                            playSound();

                            //sound.play(1);

                    }

            }

            if(gameLevel!=3 && gameLevel!=6){
```

```
                    for (int x = 0;
x<asteroidSmallNo; x++) {

                        if
(asteroid[x].checkCollision(bullet2[i].getX(),bullet2
[i].getY(),2,3)) {


            asteroid[x].hit();


            bullet2[i].fired=false;

                        }


                    }

                    for (int x = 0;
x<asteroidMedNo; x++) {

                        if
(asteroidMed[x].checkCollision(bullet2[i].getX(),bu
llet2[i].getY(),2,3)) {


            asteroidMed[x].hit();


            bullet2[i].fired=false;

                        }


                    }

                    for (int x = 0; x<asteroidBigNo;
x++) {

                        if
(asteroidBig[x].checkCollision(bullet2[i].getX(),bull
et2[i].getY(),2,3)) {


            asteroidBig[x].hit();


            bullet2[i].fired=false;

                        }


                    }

            }
                                                    }
                                                }
                                                else if
(ships.power==1) {
                                                    if
(bullet1[i].fired){

            bullet1[i].move(0,-5);

            bullet1[i].drawBullet(g);

        if
(boss.checkCollision(bullet1[i].getX(),bullet1[i].get
Y(),2,3) && !boss.dead && !boss.comingDown
&& boss.getY()>=10) {

                    boss.hit();

                    bullet1[i].fired=false;

                    score+=10;

                    playSound();

                    //sound.play(1);

            }

            for (int x = 0; x<enemyNo; x++) {

                    if
(enemy[x].checkCollision(bullet1[i].getX(),bullet1[i
].getY(),2,3) && !enemy[x].dead &&
!enemy[x].hit) {

                        enemy[x].hit();

                        bullet1[i].fired=false;

                        score+=10;

                        playSound();

                        //sound.play(1);

                    }

            }

            if(gameLevel!=3 && gameLevel!=6){

                    for (int x = 0;
x<asteroidSmallNo; x++) {

                        if
(asteroid[x].checkCollision(bullet1[i].getX(),bullet1
[i].getY(),2,3)) {


            asteroid[x].hit();


            bullet1[i].fired=false;

                        }


                    }

                    for (int x = 0;
x<asteroidMedNo; x++) {

                        if
(asteroidMed[x].checkCollision(bullet1[i].getX(),bu
llet1[i].getY(),2,3)) {
```

```
                asteroidMed[x].hit();

                bullet1[i].fired=false;
                                }

                        }
                        for (int x = 0; x<asteroidBigNo;
x++) {
                                if
(asteroidBig[x].checkCollision(bullet1[i].getX(),bull
et1[i].getY(),2,3)) {

                asteroidBig[x].hit();

                bullet1[i].fired=false;
                                }

                        }
                }
                                                        }
                                                        if
(bullet2[i].fired){
                bullet2[i].move(0,-5);
                bullet2[i].drawBullet(g);
                if
(boss.checkCollision(bullet2[i].getX(),bullet2[i].get
Y(),2,3) && !boss.dead && !boss.comingDown
&& boss.getY()>=10) {
                        boss.hit();
                        bullet2[i].fired=false;
                        score+=10;
                        playSound();
                        //sound.play(1);
                }
                for (int x = 0; x<enemyNo; x++) {
                        if
(enemy[x].checkCollision(bullet2[i].getX(),bullet2[i
].getY(),2,3) && !enemy[x].dead &&
!enemy[x].hit) {
                                enemy[x].hit();
                                bullet2[i].fired=false;
                                        score+=10;
                                        playSound();
                                        //sound.play(1);
                        }
                }
                if(gameLevel!=3 && gameLevel!=6){
                        for (int x = 0;
x<asteroidSmallNo; x++) {
                                if
(asteroid[x].checkCollision(bullet2[i].getX(),bullet2
[i].getY(),2,3)) {

                asteroid[x].hit();

                bullet2[i].fired=false;
                                }

                        }
                        for (int x = 0;
x<asteroidMedNo; x++) {
                                if
(asteroidMed[x].checkCollision(bullet2[i].getX(),bu
llet2[i].getY(),2,3)) {

                asteroidMed[x].hit();

                bullet2[i].fired=false;
                                }

                        }
                        for (int x = 0; x<asteroidBigNo;
x++) {
                                if
(asteroidBig[x].checkCollision(bullet2[i].getX(),bull
et2[i].getY(),2,3)) {

                asteroidBig[x].hit();

                bullet2[i].fired=false;
                                }

                        }
```

```
        }
                                                }
                                                if
(bullet3[i].fired) {

        bullet3[i].move(-1,-4);

        bullet3[i].drawBullet(g);

        if
(boss.checkCollision(bullet3[i].getX(),bullet3[i].get
Y(),2,3) && !boss.dead && !boss.comingDown
&& boss.getY()>=10) {

                boss.hit();

                bullet3[i].fired=false;

                score+=10;

                playSound();

                //sound.play(1);

        }

        for (int x = 0; x<enemyNo; x++) {

                if
(enemy[x].checkCollision(bullet3[i].getX(),bullet3[i
].getY(),2,3) && !enemy[x].dead &&
!enemy[x].hit) {

                        enemy[x].hit();

                        bullet3[i].fired=false;

                        score+=10;

                        playSound();

                        //sound.play(1);

                }

        }

        if(gameLevel!=3 && gameLevel!=6){

                for (int x = 0;
x<asteroidSmallNo; x++) {

                        if
(asteroid[x].checkCollision(bullet3[i].getX(),bullet3
[i].getY(),2,3)) {


        asteroid[x].hit();


        bullet3[i].fired=false;

                        }


                }

                for (int x = 0;
x<asteroidMedNo; x++) {

                        if
(asteroidMed[x].checkCollision(bullet3[i].getX(),bu
llet3[i].getY(),2,3)) {


        asteroidMed[x].hit();


        bullet3[i].fired=false;

                        }


                }

                for (int x = 0; x<asteroidBigNo;
x++) {

                        if
(asteroidBig[x].checkCollision(bullet3[i].getX(),bull
et3[i].getY(),2,3)) {


        asteroidBig[x].hit();


        bullet3[i].fired=false;

                        }


                }

        }
                                                }
                                                if
(bullet4[i].fired) {

        bullet4[i].move(1,-4);

        bullet4[i].drawBullet(g);

        if
(boss.checkCollision(bullet4[i].getX(),bullet4[i].get
Y(),2,3) && !boss.dead && !boss.comingDown
&& boss.getY()>=10) {

                boss.hit();

                bullet4[i].fired=false;

                score+=10;

                playSound();

                //sound.play(1);

        }

        for (int x = 0; x<enemyNo; x++) {

                if
(enemy[x].checkCollision(bullet4[i].getX(),bullet4[i
```

```
].getY(),2,3) && !enemy[x].dead &&
!enemy[x].hit) {

                                enemy[x].hit();

                                bullet4[i].fired=false;

                                score+=10;

                                playSound();

                                //sound.play(1);

                        }

                }

                if(gameLevel!=3 && gameLevel!=6){

                        for (int x = 0;
x<asteroidSmallNo; x++) {

                                if
(asteroid[x].checkCollision(bullet4[i].getX(),bullet4
[i].getY(),2,3)) {


                asteroid[x].hit();


                bullet4[i].fired=false;

                                }


                        }

                        for (int x = 0;
x<asteroidMedNo; x++) {

                                if
(asteroidMed[x].checkCollision(bullet4[i].getX(),bu
llet4[i].getY(),2,3)) {


                asteroidMed[x].hit();


                bullet4[i].fired=false;

                                }


                        }

                        for (int x = 0; x<asteroidBigNo;
x++) {

                                if
(asteroidBig[x].checkCollision(bullet4[i].getX(),bull
et4[i].getY(),2,3)) {


                asteroidBig[x].hit();


                bullet4[i].fired=false;
                                        }


                        }

                }

        }
                                                }
                                        }
                                        else {
                                                if
(bullet1[i].fired){

                bullet1[i].move(0,-5);

                bullet1[i].drawBullet(g);

                if
(boss.checkCollision(bullet1[i].getX(),bullet1[i].get
Y(),2,3) && !boss.dead && !boss.comingDown
&& boss.getY()>=10) {

                        boss.hit();

                        bullet1[i].fired=false;

                        score+=10;

                        playSound();

                        //sound.play(1);

                }

                for (int x = 0; x<enemyNo; x++) {

                        if
(enemy[x].checkCollision(bullet1[i].getX(),bullet1[i
].getY(),2,3) && !enemy[x].dead &&
!enemy[x].hit) {

                                enemy[x].hit();

                                bullet1[i].fired=false;

                                score+=10;

                                playSound();

                                //sound.play(1);

                        }

                }

                if(gameLevel!=3 && gameLevel!=6){

                        for (int x = 0;
x<asteroidSmallNo; x++) {

                                if
(asteroid[x].checkCollision(bullet1[i].getX(),bullet1
[i].getY(),2,3)) {


                asteroid[x].hit();
```

```
                bullet1[i].fired=false;

                                }


                        }

                        for (int x = 0;
x<asteroidMedNo; x++) {

                                if
(asteroidMed[x].checkCollision(bullet1[i].getX(),bu
llet1[i].getY(),2,3)) {


                bullet1[i].fired=false;

                                }


                        }

                        for (int x = 0; x<asteroidBigNo;
x++) {

                                if
(asteroidBig[x].checkCollision(bullet1[i].getX(),bull
et1[i].getY(),2,3)) {


                asteroidBig[x].hit();


                bullet1[i].fired=false;

                                }


                        }

                }


                                                        }
                                                        if
(bullet2[i].fired){

                bullet2[i].move(0,-5);

                bullet2[i].drawBullet(g);

                if
(boss.checkCollision(bullet2[i].getX(),bullet2[i].get
Y(),2,3) && !boss.dead && !boss.comingDown
&& boss.getY()>=10) {

                        boss.hit();

                        bullet2[i].fired=false;

                        score+=10;

                        playSound();

                        //sound.play(1);

                }

                for (int x = 0; x<enemyNo; x++) {

                        if
(enemy[x].checkCollision(bullet2[i].getX(),bullet2[i
].getY(),2,3) && !enemy[x].dead &&
!enemy[x].hit) {

                                enemy[x].hit();

                                bullet2[i].fired=false;

                                score+=10;

                                playSound();

                                //sound.play(1);

                        }

                }

                if(gameLevel!=3 && gameLevel!=6){

                        for (int x = 0;
x<asteroidSmallNo; x++) {

                                if
(asteroid[x].checkCollision(bullet2[i].getX(),bullet2
[i].getY(),2,3)) {


                asteroid[x].hit();


                bullet2[i].fired=false;

                                }


                        }

                        for (int x = 0;
x<asteroidMedNo; x++) {

                                if
(asteroidMed[x].checkCollision(bullet2[i].getX(),bu
llet2[i].getY(),2,3)) {


                asteroidMed[x].hit();


                bullet2[i].fired=false;

                                }


                        }
```

```
                    for (int x = 0; x<asteroidBigNo;
x++) {

                        if
(asteroidBig[x].checkCollision(bullet2[i].getX(),bull
et2[i].getY(),2,3)) {


            asteroidBig[x].hit();


            bullet2[i].fired=false;

                        }


                    }

            }
                                                    }
                                                    if
(bullet3[i].fired) {

            bullet3[i].move(-1,-4);

            bullet3[i].drawBullet(g);

            if
(boss.checkCollision(bullet3[i].getX(),bullet3[i].get
Y(),2,3) && !boss.dead && !boss.comingDown
&& boss.getY()>=10) {

                    boss.hit();

                    bullet3[i].fired=false;

                    score+=10;

                    playSound();

                    //sound.play(1);

            }

            for (int x = 0; x<enemyNo; x++) {

                    if
(enemy[x].checkCollision(bullet3[i].getX(),bullet3[i
].getY(),2,3) && !enemy[x].dead &&
!enemy[x].hit) {

                        enemy[x].hit();

                        bullet3[i].fired=false;

                        score+=10;

                        //sound.play(1);

                    }

            }

            if(gameLevel!=3 && gameLevel!=6){

                    for (int x = 0;
x<asteroidSmallNo; x++) {

                        if
(asteroid[x].checkCollision(bullet3[i].getX(),bullet3
[i].getY(),2,3)) {


            asteroid[x].hit();


            bullet3[i].fired=false;

                        }


                    }

                    for (int x = 0;
x<asteroidMedNo; x++) {

                        if
(asteroidMed[x].checkCollision(bullet3[i].getX(),bu
llet3[i].getY(),2,3)) {


            asteroidMed[x].hit();


            bullet3[i].fired=false;

                        }


                    }

                    for (int x = 0; x<asteroidBigNo;
x++) {

                        if
(asteroidBig[x].checkCollision(bullet3[i].getX(),bull
et3[i].getY(),2,3)) {


            asteroidBig[x].hit();


            bullet3[i].fired=false;

                        }


                    }

            }
                                                    }
                                                    if
(bullet4[i].fired) {

            bullet4[i].move(1,-4);

            bullet4[i].drawBullet(g);

            if
(boss.checkCollision(bullet4[i].getX(),bullet4[i].get
```

```
Y(),2,3) && !boss.dead && !boss.comingDown
&& boss.getY()>=10) {

                    boss.hit();

                    bullet4[i].fired=false;

                    score+=10;

                    playSound();

                    //sound.play(1);

          }

          for (int x = 0; x<enemyNo; x++) {

                    if
(enemy[x].checkCollision(bullet4[i].getX(),bullet4[i
].getY(),2,3) && !enemy[x].dead &&
!enemy[x].hit) {

                              enemy[x].hit();

                              bullet4[i].fired=false;

                              score+=10;

                              playSound();

                    //sound.play(1);

                    }

          }

          if(gameLevel!=3 && gameLevel!=6){

                    for (int x = 0;
x<asteroidSmallNo; x++) {

                              if
(asteroid[x].checkCollision(bullet4[i].getX(),bullet4
[i].getY(),2,3)) {


          asteroid[x].hit();


          bullet4[i].fired=false;

                              }


                    }

                    for (int x = 0;
x<asteroidMedNo; x++) {

                              if
(asteroidMed[x].checkCollision(bullet4[i].getX(),bu
llet4[i].getY(),2,3)) {


          asteroidMed[x].hit();


          bullet4[i].fired=false;

                              }


                    }

                    for (int x = 0; x<asteroidBigNo;
x++) {

                              if
(asteroidBig[x].checkCollision(bullet4[i].getX(),bull
et4[i].getY(),2,3)) {


          asteroidBig[x].hit();


          bullet4[i].fired=false;

                              }


                    }

          }
                                                  }
                                                  if
(bullet5[i].fired) {

          bullet5[i].move(-3,-3);

          bullet5[i].drawBullet(g);

          if
(boss.checkCollision(bullet5[i].getX(),bullet5[i].get
Y(),2,3) && !boss.dead && !boss.comingDown
&& boss.getY()>=10) {

                    boss.hit();

                    bullet5[i].fired=false;

                    score+=10;

                    playSound();

                    //sound.play(1);

          }

          for (int x = 0; x<enemyNo; x++) {

                    if
(enemy[x].checkCollision(bullet5[i].getX(),bullet5[i
].getY(),2,3) && !enemy[x].dead &&
!enemy[x].hit) {

                              enemy[x].hit();

                              bullet5[i].fired=false;

                              score+=10;

                              playSound();
```

```
                        //sound.play(1);

                    }

                }

                if(gameLevel!=3 && gameLevel!=6){

                    for (int x = 0;
x<asteroidSmallNo; x++) {

                        if
(asteroid[x].checkCollision(bullet5[i].getX(),bullet1
[i].getY(),2,3)) {


                asteroid[x].hit();


                bullet5[i].fired=false;

                        }


                    }

                    for (int x = 0;
x<asteroidMedNo; x++) {

                        if
(asteroidMed[x].checkCollision(bullet5[i].getX(),bu
llet5[i].getY(),2,3)) {


                asteroidMed[x].hit();


                bullet5[i].fired=false;

                        }


                    }

                    for (int x = 0; x<asteroidBigNo;
x++) {

                        if
(asteroidBig[x].checkCollision(bullet5[i].getX(),bull
et5[i].getY(),2,3)) {


                asteroidBig[x].hit();


                bullet5[i].fired=false;

                        }


                    }

                }
                                                    }
                                                    if
(bullet6[i].fired) {

                bullet6[i].move(3,-3);

                bullet6[i].drawBullet(g);

                if
(boss.checkCollision(bullet6[i].getX(),bullet6[i].get
Y(),2,3) && !boss.dead && !boss.comingDown
&& boss.getY()>=10) {

                    boss.hit();

                    bullet6[i].fired=false;

                    score+=10;

                    playSound();

                //sound.play(1);

                }

                for (int x = 0; x<enemyNo; x++) {

                    if
(enemy[x].checkCollision(bullet6[i].getX(),bullet6[i
].getY(),2,3) && !enemy[x].dead &&
!enemy[x].hit) {

                        enemy[x].hit();

                        bullet6[i].fired=false;

                        score+=10;

                        playSound();

                        //sound.play(1);

                    }

                }

                if (gameLevel!=3 && gameLevel!=6){

                    for (int x = 0;
x<asteroidSmallNo; x++) {

                        if
(asteroid[x].checkCollision(bullet6[i].getX(),bullet6
[i].getY(),2,3)) {


                asteroid[x].hit();


                bullet6[i].fired=false;

                        }


                    }

                    for (int x = 0;
x<asteroidMedNo; x++) {

                        if
```

```
(asteroidMed[x].checkCollision(bullet6[i].getX(),bullet6[i].getY(),2,3)) {

            asteroidMed[x].hit();

            bullet6[i].fired=false;
                        }

                }

                for (int x = 0; x<asteroidBigNo; x++) {
                        if (asteroidBig[x].checkCollision(bullet6[i].getX(),bullet6[i].getY(),2,3)) {

            asteroidBig[x].hit();

            bullet6[i].fired=false;
                        }

                }

        }
                                                }

                                        }

                }

                // player special
                if (keybool[4]){
                        if (ships.choice==1 ) ships.player1Special();
                        else if (ships.choice==2) ships.player2Special();
                        else ships.player3Special();
                }

                // draw ships
                ships.getSpeed();
                ships.updatePos(keybool);
                ships.drawShips(g);
                if (ships.player1Special) ships.drawPlayerSpecial(g);

                // life bar kanan atas
                g.setClip(0,0,screenW, screenH);
                ships.paintGradient(g, screenW-30, 5);

                // getReady untuk awal permainan
                        if(gameTicker>10 && gameTicker<50 && gameLevel==1){

            if(gameTicker%5!=0){

            g.setColor(255,255,255);

            g.setFont(Font.getFont(Font.FACE_PROPORTIONAL, Font.STYLE_BOLD, Font.SIZE_LARGE));

            g.drawString("GET READY", screenW/2, screenH/2-15-1, g.TOP|g.HCENTER);

            g.setColor(255,144,28);

            g.drawRoundRect(screenW/2-50,screenH/2-17,100,18,20,20);
                                }
                        }

                        // WARNING untuk boss datang
                        if (gameTicker>1950 && gameTicker<2000)  {
                                g.setColor(255,0,0);
                                if (gameTicker%5!=0) {

            g.setFont(Font.getFont(Font.FACE_PROPORTIONAL, Font.STYLE_BOLD, Font.SIZE_LARGE));

            g.drawString("WARNING",screenW/2 ,screenH/2-15-1 , g.TOP|g.HCENTER);

            g.drawRoundRect(screenW/2-50,screenH/2-17,100,18,20,20);
                                }
                        }

                        // load boss
                        if (gameLevel==3 || gameLevel==6) {
                                if (gameTicker==1950) {
                                        boss.load();
                                        for (int i = 0; i<4; i++){


            bossBulletLeft1[i] = new Bullet (screenW, screenH);

            bossBulletRight1[i] = new Bullet (screenW, screenH);

            bossBulletLeft2[i] = new Bullet (screenW, screenH);

            bossBulletRight2[i] = new Bullet (screenW, screenH);

                                        }
                                }
                        }
```

```
                    // next level or boss
                    if (gameLevel!=3 &&
gameLevel!=6){
                                        if
(gameTicker==1950){
                                                    if
(gameLevel==2 || gameLevel==5) unloadAsteroid();
                                                    else
loadAsteroid();

            gameTicker=0;

            gameLevel++;

                                        }
                    }
                    else if (gameTicker>2000 &&
gameLevel==3 || gameTicker>2000 &&
gameLevel==6){

                                        if (boss.getY()<10
&& !boss.withdraw) boss.comingDown();
                                        else if
(boss.bossLife<1000 && !bossAgain) {

            boss.withdraw();

            bossAgain=true;
                                        }


                                        // boss dead

                                        else if
(boss.newGame) {

            boss.unload();

                                                    for (int i =
0; i<4; i++){

            bossBulletLeft1[i] = null;

            bossBulletRight1[i] = null;

            bossBulletLeft2[i] = null;

            bossBulletRight2[i] = null;
                                                    }

            loadAsteroid();

            boss.difficulty();

            enemyCounter=0;


            difficulty++;

            gameTicker=0;

            gameLevel=1;

            boss.startGame();
                                        }
```

```
                                        else {


                                                    // draw Boss

            boss.AI1(ships.getX(),ships.getY());

            boss.drawBoss(g);


                                                    // check
Collision with the boss
                                                    if
(ships.checkCollision(boss.getX()+7,
boss.getY(),82, 45) && !ships.afterBurn &&
!boss.dead && !ships.dead && !ships.hit &&
!ships.player1Special && !ships.player2Special &&
!ships.player3Special) ships.crash();

                                                    if
(boss.special) {// && boss.specialTicker-10 >
((ships.getY()-boss.getY()-45)/5) &&
boss.specialTicker-10 < (57+(ships.getY()-
boss.getY()-45)/5)) {
                                                                if
(ships.checkCollision(boss.getX()+10,
boss.getY()+51+boss.specialTicker*5-184,15, 184)
&& !ships.afterBurn && !ships.player1Special &&
!ships.dead && !ships.hit && !ships.player2Special
&& !ships.player3Special)
ships.hit(ships.getY()+5);
                                                                if
(ships.checkCollision(boss.getX()+44,
boss.getY()+51+boss.specialTicker*5-184,3, 184)
&& !ships.afterBurn && !ships.player1Special &&
!ships.dead && !ships.hit && !ships.player2Special
&& !ships.player3Special)
ships.hit(ships.getY()+5);
                                                                if
(ships.checkCollision(boss.getX()+49,
boss.getY()+51+boss.specialTicker*5-184,3, 184)
&& !ships.afterBurn && !ships.player1Special &&
!ships.dead && !ships.hit && !ships.player2Special
&& !ships.player3Special)
ships.hit(ships.getY()+5);
                                                                if
(ships.checkCollision(boss.getX()+71,
boss.getY()+51+boss.specialTicker*5-184,15, 184)
&& !ships.afterBurn && !ships.player1Special &&
!ships.dead && !ships.hit && !ships.player2Special
&& !ships.player3Special)
ships.hit(ships.getY()+5);
                                                    }

                                                    // check
Collision with the special bullet
                                                    if
(ships.checkCollision(boss.getX()+26,
boss.clawLeft(),5, 40) && !ships.afterBurn &&
!ships.player1Special && !boss.dead  &&
!ships.dead && !ships.hit ) ships.crash();
                                                    if
(ships.checkCollision(boss.getX()+65,
boss.clawRight(),5, 40) && !ships.afterBurn &&
!ships.player1Special && !boss.dead &&
!ships.dead && !ships.hit ) ships.crash();
```

```
                                        // check
collision player special & boss
                                        if
(boss.checkCollision(ships.getX()+5,ships.laserY-
114,5,114) && !boss.dead && !boss.comingDown
&& !boss.withdraw && boss.getY()>=10 &&
ships.player3Special ) {

            boss.hit();

            boss.hit();

            boss.hit();

            score+=30;

                                        }
                                        if
(boss.checkCollision(0,ships.getSpclY(),128,50)
&& !boss.dead && !boss.comingDown &&
!boss.withdraw && boss.getY()>=10 &&
ships.player2Special) {

            boss.hit();

            boss.hit();

            score+=20;
                                        }
                                        if
(boss.checkCollision(ships.getX()-
2,ships.getY()+1,19,24) &&  !boss.dead &&
!boss.comingDown && !boss.withdraw &&
boss.getY()>=10  && ships.player1Special ) {

            boss.hit();

            score+=10;

                                        }

                                        if
(!boss.dead) {
                                                if
(boss.ticker%15==0 && boss.yPos==10){


            bossBulletLeft1[j].updateBossBullet(boss.
getX()+35, boss.getY()+45);

            bossBulletRight1[j].updateBossBullet(bos
s.getX()+35, boss.getY()+45);

            bossBulletLeft2[j].updateBossBullet(boss.
getX()+59, boss.getY()+45);

            bossBulletRight2[j].updateBossBullet(bos
s.getX()+59, boss.getY()+45);

            j++;

            j%=4;

                                                }


            g.setClip(0,0,screenW, screenH);

            for (int j=0; j<4; j++) {

            if (bossBulletLeft1[j].fired){

                        bossBulletLeft1[j].bossMove(-
2,2);


            bossBulletLeft1[j].drawBulletBoss(g,(sho
rt)bossBulletLeft1[j].bossX(),(short)bossBulletLeft1
[j].bossY(),(byte)1);

                        if
(ships.checkCollision(bossBulletLeft1[j].bossX(),bo
ssBulletLeft1[j].bossY(),5,5) &&
!ships.player1Special && !ships.afterBurn &&
!ships.player2Special && !ships.player3Special &&
!ships.dead && !ships.hit) {


            ships.hit(ships.getY()+5);


                                    }

                        if
(bossBulletLeft1[j].checkCollision(0,ships.getSpclY
(),screenW,50) && ships.player2Special) {


            bossBulletLeft1[j].fired=false;

                        }

                        if
(bossBulletLeft1[j].checkCollision(ships.getX()+5,s
hips.laserY-114,5,114) && ships.player3Special) {


            bossBulletLeft1[j].fired=false;

                        }

                        if
(bossBulletLeft1[j].checkCollision(ships.getX()-
2,ships.getY()+1,19,24) && ships.player1Special) {


            bossBulletLeft1[j].fired=false;

                        }

            }

            if (bossBulletLeft2[j].fired){

                        bossBulletLeft2[j].bossMove(-
1,3);


            bossBulletLeft2[j].drawBulletBoss(g,(sho
rt)bossBulletLeft2[j].bossX(),(short)bossBulletLeft2
[j].bossY(),(byte)1);

                        if
```

```
(ships.checkCollision(bossBulletLeft2[j].bossX(),bo
ssBulletLeft2[j].bossY(),5,5) &&
!ships.player1Special && !ships.afterBurn &&
!ships.player2Special && !ships.player3Special &&
!ships.dead && !ships.hit) {

            ships.hit(ships.getY()+5);

                        }

                if
(bossBulletLeft2[j].checkCollision(0,ships.getSpclY
(),screenW,50) && ships.player2Special) {

            bossBulletLeft2[j].fired=false;

                }

                if
(bossBulletLeft2[j].checkCollision(ships.getX()+5,s
hips.laserY-114,5,114) && ships.player3Special) {

            bossBulletLeft2[j].fired=false;

                }

                if
(bossBulletLeft2[j].checkCollision(ships.getX()-
2,ships.getY()+1,19,24) && ships.player1Special) {

            bossBulletLeft2[j].fired=false;

                }

            }

            if (bossBulletRight1[j].fired){

            bossBulletRight1[j].bossMove(1,3);

            bossBulletRight1[j].drawBulletBoss(g,(sh
ort)bossBulletRight1[j].bossX(),(short)bossBulletRi
ght1[j].bossY(),(byte)1);

                if
(ships.checkCollision(bossBulletRight1[j].bossX(),b
ossBulletRight1[j].bossY(),5,5) &&
!ships.player1Special && !ships.afterBurn &&
!ships.player2Special && !ships.player3Special &&
!ships.dead && !ships.hit) {

            ships.hit(ships.getY()+5);

                        }

                if
(bossBulletRight1[j].checkCollision(0,ships.getSpcl
Y(),screenW,50) && ships.player2Special) {

            bossBulletRight1[j].fired=false;

                }

                if
(bossBulletRight1[j].checkCollision(ships.getX()+5,
ships.laserY-114,5,114) && ships.player3Special) {

            bossBulletRight1[j].fired=false;

                }

                if
(bossBulletRight1[j].checkCollision(ships.getX()-
2,ships.getY()+1,19,24) && ships.player1Special) {

            bossBulletRight1[j].fired=false;

                }

            }

            if (bossBulletRight2[j].fired){

            bossBulletRight2[j].bossMove(2,2);

            bossBulletRight2[j].drawBulletBoss(g,(sh
ort)bossBulletRight2[j].bossX(),(short)bossBulletRi
ght2[j].bossY(),(byte)1);

                if
(ships.checkCollision(bossBulletRight2[j].bossX(),b
ossBulletRight2[j].bossY(),5,5) &&
!ships.player1Special && !ships.afterBurn &&
!ships.player2Special && !ships.player3Special &&
!ships.dead && !ships.hit) {

            ships.hit(ships.getY()+5);

                        }

                if
(bossBulletRight2[j].checkCollision(0,ships.getSpcl
Y(),screenW,50) && ships.player2Special) {

            bossBulletRight2[j].fired=false;

                }

                if
(bossBulletRight2[j].checkCollision(ships.getX()+5,
ships.laserY-114,5,114) && ships.player3Special) {

            bossBulletRight2[j].fired=false;

                }
```

```
                    if
(bossBulletRight2[j].checkCollision(ships.getX()-
2,ships.getY()+1,19,24) && ships.player1Special) {


            bossBulletRight2[j].fired=false;

                    }

            }

            bossBulletLeft1[j].checkBoundBoss();

            bossBulletRight1[j].checkBoundBoss();

            bossBulletLeft2[j].checkBoundBoss();

            bossBulletRight2[j].checkBoundBoss();
                                                }
                                        }
                                }

                    }

                    // draw Special
                    if (ships.player2Special ||
ships.player3Special) ships.drawPlayerSpecial(g);

                    // draw UI
                    g.setFont(standardFont);

                    g.setClip(0,0,screenW,
screenH);
                    g.setColor(255,255,255);

                    g.drawString(""+score+"", 40,
3, g.TOP|g.RIGHT);

            g.drawString("x"+ships.shipslife,
screenW-25+8+2, screenH-10-1-2, g.TOP|g.LEFT);

            g.drawString("x"+ships.specialPower, 10,
screenH-10-1-2, g.TOP|g.LEFT);

                    g.setClip(screenW-25+2,
screenH-10, 6,7);
                    g.drawImage(UI, screenW-25-
22+2-(ships.choice-1)*6, screenH-10,
g.TOP|g.LEFT);

                    g.setClip(2, screenH-10, 6,7);
                    g.drawImage(UI, 2-40, screenH-
10, g.TOP|g.LEFT);

                    // get power
                    if (power) {

                            g.setClip(powerX,
powerY, 11,12);
                            powerY+=2;


            if(powerOrSpecial==2) g.drawImage(UI,
powerX-11, powerY, g.TOP|g.LEFT);
                            else g.drawImage(UI,
powerX, powerY, g.TOP|g.LEFT);
                                    if
(ships.checkCollision(powerX, powerY, 11,12) &&
power && !ships.hit && !ships.dead  &&
!ships.gameOver) {
                                            if
(powerOrSpecial==2) ships.specialUp();
                                            else
ships.powerUp();

            power=false;

            powerTaken=true;
                            }
                            if
(powerY>screenH+10){

            power=false;

            powerTaken=true;
                            }
                    }


            }



            // credits
            private void credits(Graphics g){

                    if (keybool[4]) {
                            keybool[4] = false;
                            loadMenuImage =
false;
                            gameState = 20;
                    }

                    if (!loadCredits) {

                            int textHeight=10;

                            g.drawImage(menu,
0,0, g.TOP|g.LEFT);


            g.setFont(standardFont);
                            g.setColor(0,0,0);

            outlinedStrText(g,"CREDITS",screenW/2
, screenH/2-54-8-20);

            outlinedStrText(g,"Game
Design",screenW/2, screenH/2-39-6-15);

            outlinedStrText(g,"Martinus
Herwin",screenW/2, screenH/2-29-4-10);

            outlinedStrText(g,"Artwork",screenW/2,
screenH/2-14-2-5);

            outlinedStrText(g,"Martinus Herwin
",screenW/2, screenH/2-4);

            outlinedStrText(g,"Programmers",screen
W/2, screenH/2+11+2+5);
```

```
            outlinedStrText(g,"Martinus
Herwin",screenW/2, screenH/2+21+4+10);
                                    outlinedStrText(g,"---
-$$$$$-----",screenW/2, screenH/2+31+6+150);

            outlinedStrText(g,"Press 5 to
Exit",screenW/2, screenH-20);



            g.setColor(255,255,255);
                                    strText(g,"Martinus
Herwin",screenW/2, screenH/2-29-4-10);
                                    strText(g,"Martinus
Herwin",screenW/2, screenH/2-4);
                                    strText(g,"Martinus
Herwin",screenW/2, screenH/2+21+4+10);
                                    strText(g,"----$$$$$--
---",screenW/2, screenH/2+31+6+15);


            g.setColor(255,144,28);
                                    strText(g,"Game
Design",screenW/2, screenH/2-39-6-15);

            strText(g,"CREDITS",screenW/2,
screenH/2-54-8-20);

            strText(g,"Artwork",screenW/2,
screenH/2-14-2-5);

            strText(g,"Programmers",screenW/2,
screenH/2+11+2+5);
                                    strText(g,"Press 5 to
Exit",screenW/2, screenH-20);

                        }
            }

            // key config
            private void keyConfig(Graphics g) {

                        if (!loadKeyConfig) {

                                    int xOffset=20,
yOffset=40;

                                    String[] labels = {"1",
"2", "3", "4", "5", "6", "7", "8", "9"};//, "*", "0",
"#"};



                                    int d=19;
                                    int x=2;
                                    int y=2;


            g.setFont(standardFont);
                                    g.setColor(0,0,0);
                                    g.fillRect(0,0,
screenW, screenH);
                                    for (int i=0; i<3; i++)
{
                                                for (int j=0;
j<3; j++) {
                                                            if
(i==1 && j==1){

                        g.setColor(255, 255, 255);

                        g.fillRoundRect(x + xOffset + 19, y +
yOffset + 60, 18, 18, 8, 8);

                        g.setColor(153, 51, 51); // button border

                        g.fillRoundRect(x + 1 + xOffset + 19, y +
1 + yOffset + 60, 16, 16, 8, 8); // button fill

                        g.setColor(255, 255, 255);

                        g.drawString("5", x + 6 + xOffset + 19, y
+ 5 + yOffset + 60-1, g.TOP|g.LEFT); // button
number

                        g.drawString("= Special Power", x + 6 +
xOffset + 19 + 17 , y + 5 + yOffset + 60,
g.TOP|g.LEFT);
                                                            }

                        else {

                        g.setColor(255, 255, 255);


                        g.fillRoundRect(x+d*j +xOffset, y+d*i
+yOffset, 18, 18, 8, 8); // button border

                        g.setColor(153, 51, 51);


                        g.fillRoundRect((x+1)+d*j +xOffset,
(y+1)+d*i +yOffset, 16, 16, 8, 8); // button fill

                        g.setColor(255, 255, 255);

                        g.drawString(labels[i*3+j], (x+6)+d*j
+xOffset, (y+5)+d*i +yOffset-1, g.TOP|g.LEFT); //
button number
                                                            }
                                                }
                                    }

                                    g.setColor(255, 255,
255);
                                    g.fillRoundRect(x +
xOffset + 19, y + yOffset + 82, 18, 18, 8, 8);
                                    g.setColor(153, 51,
51); // button border
                                    g.fillRoundRect(x + 1
+ xOffset + 19, y + 1 + yOffset + 82, 16, 16, 8, 8); //
button fill
                                    g.setColor(255, 255,
255);
                                    g.drawString("*", x +
6 + xOffset + 19, y + 5 + yOffset + 82-1,
g.TOP|g.LEFT); // button number
                                    g.drawString("= in
Game Menu", x + 6 + xOffset + 19 + 17, y + 5 +
yOffset + 82, g.TOP|g.LEFT); // button number
```

```
                        g.drawString("=
Movements", x + xOffset + 60, y + 5 + yOffset +
19, g.TOP|g.LEFT);

                        strText(g,"Press 5 to
Exit",screenW/2, screenH-20);

                        loadKeyConfig = true;

                }

                // To option menu
                if (keybool[4]){
                        gameState=21;
                        keybool[4]=false;
                        loadOptions=false;
                }
        }

        // load options image
        private void loadOptions(Graphics g){

                if (!loadOptions){


        g.setFont(standardFont);
                        g.drawImage(menu,
0,0, g.TOP|g.LEFT);

                        g.setColor(0,0,0);

                        outlinedStrText(g,
"Key Config", screenW/2, screenH/2-10);
                        outlinedStrText(g,
"Back", screenW/2, screenH/2+10);

                        loadOptions = true;
                }

        }

        // options
        private void options(Graphics g) {

                if (keybool[7]){
                        optionChoice++;
                        keybool[7]=false;
                }

                if (keybool[1]){
                        optionChoice--;
                        if (optionChoice<0)
optionChoice=1;
                        keybool[1]=false;
                }

                optionChoice%=2;

                g.setFont(standardFont);

                g.setColor(255,255,255);
                strText(g, "Key Config",
screenW/2, screenH/2-10);
                strText(g, "Back", screenW/2,
screenH/2+10);

                g.setColor(255,144,28);

                        if (optionChoice==0) strText(g,
"Key Config", screenW/2, screenH/2-10);
                        else strText(g, "Back",
screenW/2, screenH/2+10);

                        if (keybool[4]) {
                                if (optionChoice==0)
{

        gameState=27;

        loadKeyConfig = false;

                                }
                                else {

        gameState=20;

        loadMenuImage=false;
                                }
                                keybool[4]=false;
                        }
        }


        public void ticker() {
                if(tick == 10) {
                        sec++;
                        tick = 0;
                }
                tick++;
        }


        private Image loadImage ( String filename
) {
                Image img;
                try {
                        img =
Image.createImage(filename);
                } catch(IOException ie){
                        img = null;

        System.out.println("error loading image");
                }
                return img;
        }



        //method to output string
        public void strText(Graphics g, String
msg, int W, int H){

        g.drawString(msg,W,H,g.TOP|g.HCENT
ER);
        }

        public void keyPressed(int keycode) {
                key=keycode;

                if(keycode == KEY_NUM2 ||
keycode == -1) {
                        keybool[1] = true;
```

```
                }
                if(keycode == KEY_NUM4 ||
keycode == -3) {
                        keybool[3] = true;
                }
                if(keycode == KEY_NUM8 ||
keycode == -2) {
                        keybool[7] = true;
                }
                if(keycode == KEY_NUM6  ||
keycode == -4) {
                        keybool[5] = true;
                }
                if(keycode == KEY_NUM1) {
                        keybool[0] = true;
                }
                if(keycode == KEY_NUM7) {
                        keybool[6] = true;
                }
                if(keycode == KEY_NUM9) {
                        keybool[8] = true;
                }
                if(keycode == KEY_NUM3) {
                        keybool[2] = true;
                }
                if(keycode == KEY_NUM5 ||
keycode == -5 || keycode == -6 ) {
                        keybool[4] = true;
                }
                if(keycode == KEY_POUND) {
                        keybool[11] = true;
                }
                if(keycode == KEY_STAR ||
keycode == 42) {
                        keybool[13] = true;
                }
        }


        // input name
        private void inputName(Graphics g) {


                char data1[] =
{'A','B','C','D','E','F','G','H','I','J','K','L','M','N','O','P','
Q','R','S','T','U','V','W','X','Y','Z','<',' '};
                char a[]={' '};

                int name1=0;
                int     charno = 0;
                int x=0,y=0;
                int xx=0,yy=0;

                String list1="";

                choice2%=28;

                g.setColor(0,0,0);
                g.fillRect(0,0,screenW,
screenH);


        g.setFont(Font.getDefaultFont());

                // selecting the char
                if (keybool[1] || key == -1){
                        key = 0; keybool[1] =
false;
                        choice2=choice2-7;
                        if (choice2<0){

        choice2=choice2+28;
                        }

                }
                else if (keybool[7] || key == -2){
                        key = 0; keybool[7] =
false;
                        choice2=choice2+7;
                }
                else if(keybool[3] || key == -3) {
                        key = 0; keybool[3] =
false;
                        choice2--;
                        if (choice2<0){

        choice2=choice2+28;
                        }
                }
                else if (keybool[5] || key == -4){
                        key = 0; keybool[5] =
false;
                        choice2++;
                }

                g.setColor(255, 255, 255);
                g.setFont(standardFont);
                g.drawString("Input Name",
(screenW/2), 20, g.TOP|g.HCENTER);

                g.drawRoundRect((screenW/2)-
56, screenH/3+50-1, 112, 17,15,15);

                g.setColor(255, 144, 28); //
color for box

                for(x=0;x<4;x++){
                        for(y=0;y<7;y++){
                                charno = x *
7 + y;
                                if
(charno==27){

        g.drawString("End", (screenW/2)+51-11,
screenH/3+33-1, g.TOP|g.LEFT);
                                }
                                a[0]=
data1[charno];
                                String list =
new String(a);

        g.drawString(list, (screenW/2)-51-
3+y*17, screenH/3-3+x*14-6-1, g.TOP|g.LEFT);
                        }
                }

                // karakter ubah warna
                for(x=0;x<4;x++){
                        for(y=0;y<7;y++){
                                name1 = x *
7 + y;
                                if
(choice2==name1){
```

```
            a[0]= data1[name1];

            String list = new String(a);

            color++;

            g.setColor(255, 255, 255);
                                                if
(choice2==27){

            g.drawRect((screenW/2)+51-11-2,
(screenH/3+33)-1,20,11);
                                                }

            else {

            g.drawRect(screenW/2-51-3+y*17-2,
screenH/3-2+x*14-6-2,9,11);
                                                }
                                        }
                                }
                        }

                        if(keybool[4]){
                                key=0;

                                // if < is pressed then
delete one char
                                if (choice2==26){

            if(namecount==0){

            name[namecount]=' ';

            namecount=0;
                                        } else {

            name[namecount]=' ';

            namecount--;
                                        }
                                }

                                // if 5 is pressed then
put the char
                                else if(choice2<26
&& first){
                                        if
((name[namecount]==' ') || namecount==4) {

            name[namecount]=data1[choice2];

                                        } else {

            name[namecount+1]=data1[choice2];
//previous action was a delete

                                        }
                                        if
(namecount>=4){

            namecount=4;
                                        } else {


            namecount++;
                                                }

                                        }

                                else if (choice2<26
&& !first) {
                                                nameRect =
0;
                                                for (int i =
0; i < 5; i++) {


            name[i]=' ';
                                                }
                                                namecount
= 0;

            name[namecount]=data1[choice2];

                                                first = true;
                                        }

                                        // if End is pressed
then go to highscore menu
                                        else if(choice2==27){
                                                // Reset();
                                                for (int i =
0; i < 5; i++) {

                nameRect +=
standardFont.charWidth(name[i]);
                                                }
                                                nameRect
+= 1;
                                                keybool[4]
= false;
                                                list1 = new
String(name);
                                                list1 =
list1.trim();

            addHighScore(list1, score, sChoice);

            gameState=25;

                                                first = false;


                                        }
                                        keybool[4]=false;
                                }


            }

            private void highScore(Graphics g){

            }

            public void addHighScore(String
playerName, int hscore, int selection) {
                        try{
                                byte[] data, dataScore;
String string, rec;
                                string = playerName +
"/" + Integer.toString(hscore) + "|" +
Integer.toString(selection)+ " ";
```

```
                openHighScores();
                if
(myStore.getNumRecords()==0) {
                        data =
string.getBytes();

        myStore.addRecord(data, 0, data.length);

        createHiScoreArray();
                } else {
                        if
(hiScoreRec.equals("_")) data = string.getBytes();
                        else {

        createHiScoreArray();

        rec = sortHiScore(playerName, hscore,
selection);

        data = rec.getBytes();
                        }

        myStore.setRecord(1, data, 0,
data.length);

        createHiScoreArray();
                }
                closeHighScores();
            } catch(RecordStoreException
rse) {
            }
        }

        private void createHiScoreArray() {
            int previousIndex = 0, j=0,
playerScore=0, playerShips=0;
            String playerName = "";
            if (hiScoreRec.equals("_") ||
hiScoreRec==null) j=0;
            else {
                for (int i=0;
i<hiScoreRec.length(); i++) {
                    if
(hiScoreRec.charAt(i)=='/') {

        playerName =
hiScoreRec.substring(previousIndex, i);

        previousIndex = i+1;
                    } else if
(hiScoreRec.charAt(i)=='|') {

        playerScore =
Integer.parseInt(hiScoreRec.substring(previousInde
x, i));

        previousIndex = i+1;

        //scoreArray[j] = new
ScoreRecord(playerName, playerScore);

        //j++;
                    }
                    else if
(hiScoreRec.charAt(i)==' ') {

        playerShips =
Integer.parseInt(hiScoreRec.substring(previousInde
x, i));

        previousIndex = i+1;

        scoreArray[j] = new
ScoreRecord(playerName, playerScore,
playerShips);

        j++;
                        }
                    }
                }
                numberScores = j;
        }

        private String sortHiScore(String
playerName, int hscore, int selection) {
                ScoreRecord[] tmp = new
ScoreRecord[5];
                ScoreRecord newRec = new
ScoreRecord(playerName, hscore, selection);
                String rec = "";
                int i=0;
                while (scoreArray[i]!= null &&
scoreArray[i].score > hscore && i <= 4) {
                    tmp[i] =
scoreArray[i];
                    i++;
                }
                if (i<5) {
                    tmp[i] = newRec;
                    while (i<4) {
                        if
(scoreArray[i]!=null)

        tmp[i+1] = scoreArray[i];
                        i++;
                    }
                }
                for (i=0; i<5; i++) {
                    if (tmp[i]!=null) {

        scoreArray[i] = tmp[i];
                        rec =
rec + scoreArray[i].write();

                    }
                }
                hiScoreRec = rec.trim() + " ";
                return rec;
        }



        private void loadHighScore(Graphics g) {

                if (menu==null)
menu=loadImage("/menu.png");
                if (UI == null) UI =
loadImage("/UI.png");
                String scores[];
                int result = getHighScore();
                score = 0;
                if (keybool[4]) {
                    gameState=42;
```

```
                    keybool[4] = false;


                }

                //g.setColor(0,0,0);

        g.drawImage(menu,0,0,g.TOP|g.LEFT);

                outlinedStrText(g,"HIGH
SCORES", screenW/2, screenH/5-12);
                g.setColor(255, 144, 28);

                g.drawString("HIGH
SCORES", screenW/2, screenH/5-12,
g.BASELINE|g.HCENTER);
                outlinedStrText(g,"Press 5 to
Exit", screenW/2, screenH-20);
                g.setColor(255, 144,28);
                g.drawString("Press 5 to Exit",
screenW/2, screenH-20, g.TOP|g.HCENTER);
                if (result == 1) {
                    outlinedStrText(g,"No
high scores", screenW/2, screenH/5+4);
                    g.setColor(255, 144,
28);
                    g.drawString("No
high scores", screenW/2, screenH/5+4,
g.TOP|g.HCENTER);
                }
                else {
                    for (int i=0; i<5; i++)
{

        g.setColor(241,175,108);

        g.fillRoundRect((screenW/2)-57,
screenH/5+i*18, 112, 15, 12, 12);

        g.setColor(84,28,0);

        g.drawRoundRect((screenW/2)-57,
screenH/5+i*18, 112, 15, 12, 12);
                    }
                    StringBuffer scoreInfo
= new StringBuffer();
                    String infoToDraw;
                    int counterRec=0;
                    g.setColor(84,28,0);
                    for (int i=0;
i<numberScores; i++) {

        g.setClip(0,0,screenW, screenH);

        g.drawString(scoreArray[i].name,
(screenW/2)-50+9, screenH/5+4+i*18-2,
g.TOP|g.LEFT);
                        String
scoreStr = Integer.toString(scoreArray[i].score);

        g.drawString(scoreStr, (screenW/2)+50,
screenH/5+4+i*18-2, g.TOP|g.RIGHT);
                        g.setClip
(screenW/2-50, screenH/5+4+i*18-1+2, 6,7);

        g.drawImage(UI,screenW/2-50-22-
scoreArray[i].selection*6, screenH/5+4+i*18-1+2,
g.TOP|g.LEFT);
                    }
                }
                outlinedStrText(g,"Press 5 to
Exit", screenW/2, screenH-20);
                g.setColor(255, 144,28);
                g.drawString("Press 5 to Exit",
screenW/2, screenH-20, g.TOP|g.HCENTER);
        }

        public int openHighScores() {
            try{
                myStore =
RecordStore.openRecordStore("ScoreDB", true);
                if (
myStore.getNumRecords() == 0 ) {
                    String
nullRec = new String("_");
                    byte[] data
= nullRec.getBytes();

        myStore.addRecord(data, 0, data.length);

                }
                if (
myStore.getNumRecords() > 0 ) {

        //System.out.println("number of Records
= " + myStore.getNumRecords());
                    hiScoreRec
= new String(myStore.getRecord(1));

        //System.out.println("hiScoreRec - " +
hiScoreRec);
                    if
(hiScoreRec.equals("_")) return 1; //no scores
                    else
{createHiScoreArray(); return 0; } //scores exist
                }
            }
            catch(RecordStoreException
rse){
            }
            return 1;
        }

        private void closeHighScores() {
            if(myStore != null){
                try {

        myStore.closeRecordStore();
                }

        catch(RecordStoreException frse){}
            }
        }




        private void askUploadScore(Graphics g)
{
```

```
                int[][] coords = { {screenW/2-
43, screenH*7/10}, {screenW/2-10, screenH*7/10}
};
                g.setColor(0,0,0);
                g.fillRect(0,0,screenW,
screenH);
                //fillGradientBG(g, 0, 0, 103,
102, 204, 255);
                g.setColor(255, 255, 255);
                g.drawString("Would you like
to", screenW/2, screenH*1/10,
g.TOP|g.HCENTER);
                g.drawString("upload your high
scores", screenW/2, screenH*2/10,
g.TOP|g.HCENTER);
                g.drawString("to the server?",
screenW/2, screenH*3/10, g.TOP|g.HCENTER);
                g.drawString("(Layanan summit
high score", screenW/2, screenH*4/10,
g.TOP|g.HCENTER);
                g.drawString("masih under
reconstruction", screenW/2, screenH*5/10,
g.TOP|g.HCENTER);
                g.drawString("masih error)",
screenW/2, screenH*6/10, g.TOP|g.HCENTER);
                g.drawString("YES",
screenW/2-25, screenH*7/10+10, g.TOP|g.LEFT);
                g.drawString("NO",
screenW/2+8, screenH*7/10+10, g.TOP|g.LEFT);



                if ( keybool[4] ) {
                        keybool[4] = false;
                        //Yes
                        if ( menuSelection ==
0 ) {
                                menu=null;
                                UI=null;

        //unloadGamePlay = false;

        loadMenuImage = false;

                                reset();

        unloadGamePlay();
                                gameState =
43;
                                //return;
                        }
                        //No
                        else if (
menuSelection == 1 ) {
                                menu=null;
                                UI=null;

        //unloadGamePlay = false;

        loadMenuImage = false;

                                reset();

        unloadGamePlay();
                                gameState =
20;
                                //return;
                        }
                }


                g.setColor(255,144,28);
                if (menuSelection==0)
g.drawString("YES", screenW/2-25,
screenH*7/10+10, g.TOP|g.LEFT);
                else g.drawString("NO",
screenW/2+8, screenH*7/10+10, g.TOP|g.LEFT);

        }

        private void inGameMenu(Graphics g) {

                if (keybool[7]){

                        inGameChoice++;
                        keybool[7]=false;
                }

                if (keybool[1]){

                        inGameChoice--;
                        if (inGameChoice<0)
inGameChoice=2;
                        keybool[1]=false;
                }

                inGameChoice%=2;

                g.setColor(255,255,255);

        strText(g,"Paused",screenW/2,screenH/2-
20);

        strText(g,"Resume",screenW/2,screenH/2
);
                strText(g,"Quit to Main
Menu",screenW/2,screenH/2+20);

                g.setColor(255,144,28);
                if(inGameChoice==0)
        strText(g,"Resume",screenW/2,screenH/2
);
                else strText(g,"Quit to Main
Menu",screenW/2,screenH/2+20);


                if(keybool[4]){
                        if(inGameChoice==0)
        gameState=30;
                        else {


        loadMenuImage=false;

        //unloadGamePlay=false;
                                reset();

        unloadGamePlay();

        gameState=20;
                        }
                        keybool[4]=false;
                }
        }
```

LAMPIRAN B
QUESIONER GAME