



UNIVERSIDADE da MADEIRA

Faculdade de Ciências Exatas e Engenharia

Programação Orientada a Objetos

Projeto 1

Barbie - The Game

Verónica Medina, 2121022 Rodrigo Freitas, 2120522

2023/2024

Índice

I	Introdução	1
II	Barbie - The Game	2
II.1	World	2
II.1.1	Background	2
II.1.2	StartMenu	2
II.1.3	MainMenu	2
II.1.4	Play	2
II.1.5	Help	2
II.1.6	Credits	2
II.1.7	PlayAgain	2
II.1.8	Game	2
II.1.9	GameOver	3
II.1.10	Win	3
II.2	Actor	3
II.2.1	Button	3
II.2.2	Diamonds	3
II.2.3	Fire	3
II.2.4	LifeHearts	3
II.2.5	Makeup	4
II.2.6	BrickBlock	4
II.2.7	Players	4
II.2.8	Potions	4
III	Conclusão	5
Anexos		5

I. Introdução

O projeto 1 (Barbie - The Game) visa aplicar os conhecimentos obtidos nas aulas teóricas de Programação Orientada a Objetos, assim como nas aulas práticas, nas quais foi utilizado o programa Greenfoot.

O projeto em questão tem como objetivo o desenvolvimento de um jogo usando a plataforma Greenfoot e conceitos de POO.

O tema do jogo baseia-se no filme “Barbie - The Movie”. Ambas as personagens (Barbie e Ken) foram de férias visitar o Planeta Terra. Contudo, devido a uma anomalia, o portal que faz ligação entre o “Planeta Terra” e “Barbie Land” fechou e, consequentemente, já não conseguem voltar para casa.

O seguinte projeto tem como requisitos mínimos:

1. 2 ou mais jogadores que colaboram;
2. Score com a pontuação atualizada em tempo real;
3. Indicador de vida/energia/tempo ou semelhante para cada jogador;
4. Modificação do aspetto do mundo e dos jogadores (por exemplo, imagens, posição de objetos, ou cores) durante o jogo;
5. Indicação da pontuação obtida ao finalizar o jogo.
6. Inicialização de objetos usando os construtores;
7. Herança de métodos com um mínimo de 2 níveis além de Actor;
8. ”Overriding” de métodos;
9. ”Overloading” de métodos;
10. Encapsulamento.

II. Barbie - The Game

II.1. World

II.1.1 Background

Em **A**, está representado o fundo escolhido para o menu principal e o respetivo botão (ButtonBarbie) **A.0.2** que nos leva para o StartMenu. O mundo tem como proporções (1300,700,1).

II.1.2 StartMenu

Em **B.0.1** encontramos o StartMenu após clicar no botão Barbie (no background). O StartMenu é composto por quatro botões: PLAY, HELP, CREDITS e MAINMENU.

II.1.3 MainMenu

Em **C.0.1** temos o botão MainMenu representado por uma casinha que leva o utilizador de volta ao Background.

II.1.4 Play

Em **C.0.2** temos o botão Play que leva o utilizador ao jogo.

II.1.5 Help

Em **C.0.3** temos o botão que dá ao utilizador a explicação/instruções do jogo, caso necessite.

II.1.6 Credits

Em **C.0.4** temos o botão que mostra ao utilizador os autores do projeto.

II.1.7 PlayAgain

Em **C.0.5** temos o botão que permite o utilizador jogar de novo após ter perdido ou ganho.

II.1.8 Game

Em **D** observamos o fundo escolhido para o jogo e o seu design. No Game está localizado o código mais importante do projeto. Em **D.0.1** definimos as variáveis e a música escolhida para o decorrer do jogo. Em **D.0.2** temos o score que é inicializado a 0, o texto que aparece no jogo referente à vida dos personagens. Em **D.0.4** temos o código referente à condição GameOver, no qual se a vida da Barbie for menor ou igual a 0 ou se a vida do Ken for menor ou igual a 0, o utilizador deparar-se-á com o GameOver e a música do jogo pára. Em **D.0.5** temos a condição para ganhar o jogo,

que é se a Barbie apanhar as suas duas poções e o Ken também, sem perderem as suas vidas. Em **D.0.6** temos o código para fazer os obstáculos aparecerem de forma aleatória e código para o sapato alto e o rímel aparecerem no lado esquerdo do mundo e vice versa. Em **D.0.7** temos o código para aparecer a pontuação dos personagens e adicionar pontos à mesma, assim como para que os personagens percam vida e para mostrar a mesma no ecrã do jogo e por fim, o código para apanharem as suas poções, cujas variáveis serão usadas na função de ganhar anterior. Em **D.0.8** temos o design do nosso jogo onde foram colocados os BrickBlocks, os personagens no seu lugar inicial, os diamantes que têm de apanhar e as suas poções, o fogo, assim como os corações das suas vidas.

II.1.9 GameOver

Em **E** temos o fundo com que o utilizador se depara quando perde. (Vida dos personagens chega a 0 ou caiem no fogo).

II.1.10 Win

Em **F** temos o fundo com que o utilizador se depara quando ganha. (Pontuação incluída)

II.2. Actor

II.2.1 Button

Em **A.0.2, C.0.1, C.0.2, C.0.3, C.0.3, C.0.4** e em **C.0.5** encontramos os botões feitos e utilizados neste projeto. Foi necessário fazer alterações ás suas proporções para obter o design pretendido, assim como foi adicionado um som para quando o utilizador clica no botão.

II.2.2 Diamonds

Existem diamantes nos blocos. Diamantes rosa são os da Barbie e os cinza os do Ken, cada diamante apanhado adiciona na pontuação 10 pontos.

II.2.3 Fire

Se o utilizador deixar a Barbie ou o Ken cair no fogo rosa encontrado no centro do jogo, o mesmo acaba (GameOver).

II.2.4 LifeHearts

São a representação das vidas dos personagens, juntamente com a mensagem de texto da vida restante da Barbie e do Ken.

II.2.5 Makeup

A maquilhagem utilizada no jogo têm a função de dificultar o mesmo, dado que quando um dos personagens é atingido, ou pelo sapato ou pelo rímel, são retirados 10 pontos de vida.

II.2.6 BrickBlock

Os blocos funcionam como plataformas, nos quais os personagens têm se saltar e andar para apanhar os diamantes e as suas poções. Os blocos foram colocados no jogo de modo a termos o design pretendido, não tendo um jogo impossível nem um jogo demasiado fácil.

II.2.7 Players

Os jogadores são a Barbie e o Ken. Em **K** e **L** temos os códigos dos personagens, onde foram definidas as suas imagens espelhadas para o movimento nos dois sentidos, os seus comandos A, D e W e UP, LEFT e RIGHT, os seus movimentos, saltar andar para a esquerda e direita, cair se não estiver num bloco, se saltar por baixo de um bloco bate com a cabeça para tornar o jogo mais difícil, não continuar sempre caso tenha um bloco na esquerda ou na direita, saber se estão no chão ou não para determinar se cai ou se fica em cima do bloco (chão), e os seus pontos e vida, são adicionados 10 pontos à pontuação a cada diamante apanhado e são retirados 10 pontos de vida se são atingidos por obstáculos. Quando a Barbie é atingida por um obstáculo, o som é diferente do que quando o mesmo acontece ao Ken.

II.2.8 Potions

No jogo existem quatro poções, sendo a rosa e a amarela as poções da barbie e a verde e a azul as do Ken. Não é possível ganhar sem que os utilizadores apanhem todas as poções.

III. Conclusão

Com a realização deste projeto, concluiu-se que é possível realizar um jogo, com recurso à plataforma Greenfoot e conceitos de POO.

Os requisitos mínimos pretendidos no enunciado foram cumpridos pelo grupo.

Em geral, acreditamos que a implementação foi executada corretamente, e os códigos possuem uma linguagem clara ao olho de cada um, dado que tentamos ao máximo evitar o uso de abreviações.

Anexos

A. Background



A.0.1 Background

```
public class Background extends World {  
    public Background() {  
        super(1300, 700, 1);  
        addObject(new ButtonBarbie(), 400, 355);  
    }  
}
```

A.0.2 ButtonBarbie



```
public class ButtonBarbie extends Button {
```

```
public ButtonBarbie()
{
    GreenfootImage imagem = getImage();
    int y = imagem.getHeight();
    int x = imagem.getWidth();
    imagem.scale(x*2,y*2);
}
public void act()
{
    clickSound();
}
public void clickSound()
{
    if(Greenfoot.mouseClicked(this))
    {
        Greenfoot.setWorld(new StartMenu());
        sound();
    }
}
```

B. StartMenu



B.0.1 StartMenu

```
public class StartMenu extends World {  
    public StartMenu() {  
        super(1300, 700, 1);  
        addObject(new ButtonPlay(), 320, 155);  
        addObject(new ButtonHelp(), 320, 255);  
        addObject(new ButtonCredits(), 320, 355);  
        addObject(new ButtonMainMenu(), 100, 600); } }
```

C. Button

C.0.1 ButtonMainMenu



```
public class ButtonMainMenu extends Button
{
    public ButtonMainMenu()
    {
        GreenfootImage imagem = getImage();
        int y = imagem.getHeight();
        int x = imagem.getWidth();
        imagem.scale(x/3,y/3);
    }
    public void act()
    {
        clickSound();
    }
    public void clickSound()
    {
        if(Greenfoot.mouseClicked(this)){
            Greenfoot.setWorld(new Background());
            sound();
        }
    }
}
```

C.0.2 ButtonPlay

```
public class ButtonPlay extends Button
{
    public ButtonPlay()
    {
        GreenfootImage imagem = getImage();
        int y = imagem.getHeight();
        int x = imagem.getWidth();
        imagem.scale(x/4,y/4);
    }
}
```



PLAY

C.0.3 ButtonHelp



HELP

```
public class ButtonHelp extends Button
{
    public ButtonHelp()
    {
        GreenfootImage imagem = getImage();
        int y = imagem.getHeight();
        int x = imagem.getWidth();
        imagem.scale(x/4,y/4);
    }
    public void act()
    {
        clickSound();
    }
    public void clickSound()
    {
        if(Greenfoot.mouseClicked(this))
        {
            Greenfoot.setWorld(new Help());
            sound();
        }
    }
}
```

C.0.4 ButtonCredits

CREDITS

```
public class ButtonCredits extends Button
{
    public ButtonCredits()
    {
        GreenfootImage imagem = getImage();
        int y = imagem.getHeight();
        int x = imagem.getWidth();
        imagem.scale(x/4,y/4);
    }
    public void act()
    {
        clickSound();
    }
    public void clickSound()
    {
        if(Greenfoot.mouseClicked(this))
        {
            Greenfoot.setWorld(new Credits());
            sound();
        }
    }
}
```

C.0.5 ButtonPlayAgain

PLAY AGAIN

D. Game



D.0.1 Variáveis

```
private GreenfootSound gameMusic = new GreenfootSound("GameTheme.mp3");
private int score=0;
private int lifeBarbie=100;
private int lifeKen=100;
private boolean getPotion1Barbie=false;
private boolean getPotion2Barbie=false;
private boolean getPotion1Ken=false;
private boolean getPotion2Ken=false;
```

D.0.2 PublicGame

```
public Game() { super(1300, 700, 1);
gameDesign();
addRightHeel();
addRightMascara();
addLeftHeel();
addLeftMascara();
score=0;
playersScore();
showText("BARBIE'S HP: 100 ",getWidth()/4,50);
showText("KEN'S HP: 100 ",getWidth()/2 + 350,50);
}
```

D.0.3 Act

```
public void act(){
    gameMusic.play();
    addRightHeel();
    addRightMascara();
    addLeftHeel();
    addLeftMascara();
    barbieHP();
    kenHP();
    playersScore();
    lose();
    win();
}
```

D.0.4 Lose

```
public void lose()
{
    if (lifeBarbie <=0 || lifeKen <=0)
    {
        Greenfoot.setWorld(new GameOver());
        gameMusic.stop();
    }
}
```

D.0.5 Win

```
public void win()
{
    if(getPotion1Barbie == true && getPotion2Barbie == true &&
    getPotion1Ken == true && getPotion2Ken == true ){
        Greenfoot.setWorld(new Win(score));
        gameMusic.stop();
    }
}
```

D.0.6 Makeup

```
public void addRightHeel()
{
    if (Greenfoot.getRandomNumber(400)<1)
    {
        int x = 0;
        int y = Greenfoot.getRandomNumber(getHeight()-50);
        addObject(new Heel1(), x, y);
    }
}
```

```

public void addRightMascara()
{
if (Greenfoot.getRandomNumber(400)<1)
{ int x = 0;
int y = Greenfoot.getRandomNumber(getHeight()-50);
addObject(new Mascara1(), x, y);
}
}
public void addLeftHeel()
{
if (Greenfoot.getRandomNumber(400)<1)
{
int x = getWidth()-10;
int y = Greenfoot.getRandomNumber(getHeight()-50);
addObject(new Heel2(), x, y);
}
}
public void addLeftMascara()
{
if (Greenfoot.getRandomNumber(400)<1)
{
int x = getWidth()-10;
int y = Greenfoot.getRandomNumber(getHeight()-50);
addObject(new Mascara2(), x, y); }
}

```

D.0.7 Vida, Pontos

```

public void playersScore()
{
showText("SCORE: " + score,getWidth()/2,50);
}
public void addPointsScore(int points)
{
score= score + points; }
public void removeLifeBarbie(int healthPoints)
{
lifeBarbie=lifeBarbie + healthPoints;
}
public void removeLifeKen(int healthPoints)
{
lifeKen=lifeKen + healthPoints;
}
public void barbieHP()
{
showText("BARBIE'S HP: " + getBarbieHP(),getWidth()/4,50);
}

```

```

public int getBarbieHP()
{
    return lifeBarbie;
}
public void setBarbieHP(int life)
{
    lifeBarbie = life;
}
public void kenHP()
{
    showText("KEN'S HP: " + getKenHP(),getWidth()/2 + 350,50);
}
public int getKenHP()
{
    return lifeKen;
}
public void setKenHP(int life)
{
    lifeKen = life;
}
public void potion1Barbie()
{
    getPotion1Barbie=true;
}
public void potion2Barbie()
{
    getPotion2Barbie=true;
}
public void potion1Ken()
{
    getPotion1Ken=true;
}
public void potion2Ken()
{
    getPotion2Ken=true;
}

```

D.0.8 GameDesign

```

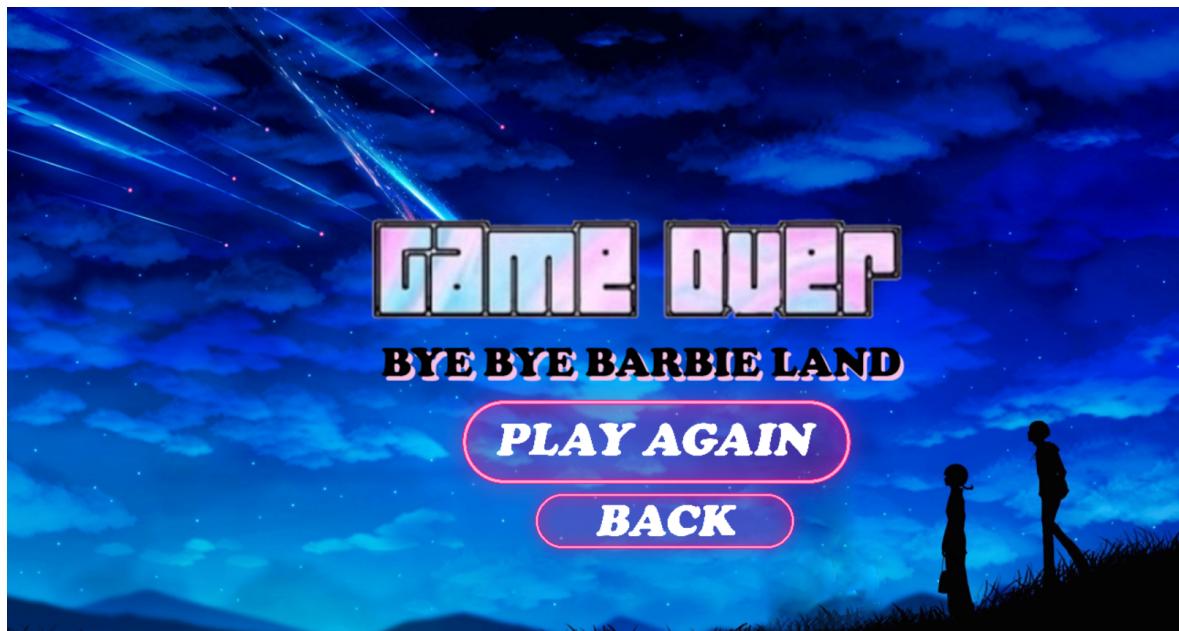
private void gameDesign()
{
    addObject(new BrickBlock(),0,685);
    addObject(new BrickBlock(),80,685);
    addObject(new BrickBlock(),160,685);
    addObject(new BrickBlock(),240,685);
    addObject(new BrickBlock(),320,685);
    addObject(new BrickBlock(),400,685);
}

```

```
addObject(new BrickBlock(),480,685);
addObject(new BrickBlock(),560,685);
addObject(new BrickBlock(),640,685);
addObject(new BrickBlock(),720,685);
addObject(new BrickBlock(),800,685);
addObject(new BrickBlock(),880,685);
addObject(new BrickBlock(),960,685);
addObject(new BrickBlock(),1040,685);
addObject(new BrickBlock(),1120,685);
addObject(new BrickBlock(),1200,685);
addObject(new BrickBlock(),1280,685);
addObject(new BrickBlock(),79,121);
addObject(new BrickBlock(),237,187);
addObject(new BrickBlock(),522,159);
addObject(new BrickBlock(),394,238);
addObject(new BrickBlock(),78,283);
addObject(new BrickBlock(),255,333);
addObject(new BrickBlock(),390,539);
addObject(new BrickBlock(),253,601);
addObject(new BrickBlock(),81,541);
addObject(new BrickBlock(),81,406);
addObject(new BrickBlock(),251,457);
addObject(new BrickBlock(),1221,121);
addObject(new BrickBlock(),1063,187);
addObject(new BrickBlock(),778,159);
addObject(new BrickBlock(),906,238);
addObject(new BrickBlock(),1222,283);
addObject(new BrickBlock(),1045,333);
addObject(new BrickBlock(),910,539);
addObject(new BrickBlock(),1047,601);
addObject(new BrickBlock(),1219,541);
addObject(new BrickBlock(),1219,406);
addObject(new BrickBlock(),1049,457);
addObject(new BrickBlock(),776,473);
addObject(new BrickBlock(),524,473);
addObject(new BrickBlock(),651,307);
addObject(new LifeHeartBarbie(),205,50);
addObject(new LifeHeartKen(),896,50);
addObject(new Potion1Barbie(),810,652);
addObject(new Potion1Ken(),71,81);
addObject(new Potion2Barbie(),1214,81);
addObject(new Potion2Ken(),490,652);
addObject(new DiamondBarbie(),238,156);
addObject(new DiamondBarbie(),520,130);
addObject(new DiamondBarbie(),252,302);
addObject(new DiamondBarbie(),1223,252);
addObject(new DiamondBarbie(),1217,508);
```

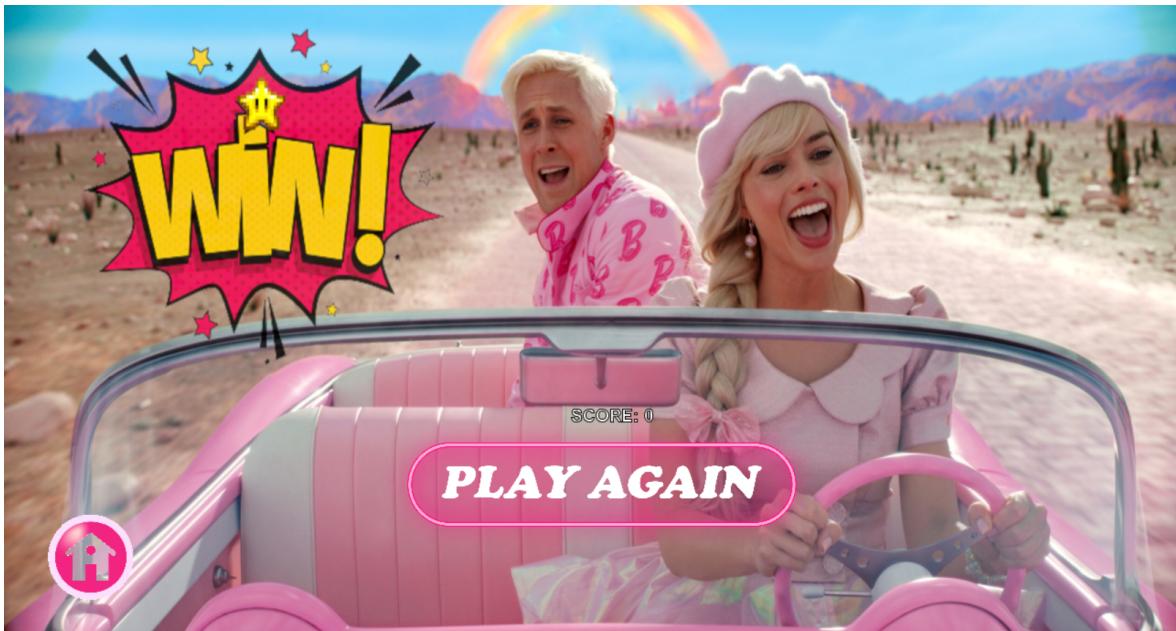
```
 addObject(new DiamondBarbie(),1046,568);
addObject(new DiamondBarbie(),908,507);
addObject(new DiamondBarbie(),775,443);
addObject(new DiamondBarbie(),907,206);
addObject(new DiamondKen(),1061,156);
addObject(new DiamondKen(),1051,427);
addObject(new DiamondKen(),779,128);
addObject(new DiamondKen(),396,207);
addObject(new DiamondKen(),524,442);
addObject(new DiamondKen(),250,424);
addObject(new DiamondKen(),78,251);
addObject(new DiamondKen(),79,510);
addObject(new DiamondKen(),253,569);
addObject(new Ken(),1223,643);
addObject(new Barbie(),78,643);
addObject(new Fire(),650,590);
}
```

E. GameOver



```
public class GameOver extends World
{
    public GameOver()
    {
        super(1300, 700, 1);
        Greenfoot.playSound("gameover.mp3");
        addObject(new ButtonBack(),720,564);
        addObject(new ButtonPlayAgain(),711,478);
    }
}
```

F. Win



```
public class Win extends World
{
    public Win(int Score)
    {
        super(1300, 700, 1);
        showText("SCORE: " + Score, 665, 450);
        GreenfootSound sound = new GreenfootSound("barbieWin.mp3");
        sound.play();
        addObject(new ButtonMainMenu(), 100, 600);
        addObject(new ButtonPlayAgain(), 654, 526);
    }
}
```

G. Actor

G.0.1 Potion1Barbie, Potion2Barbie

```
public class Potion1Barbie extends Actor
{
    public void act()
    {
        if (isTouching(Barbie.class))
        {
            getWorldOfType(Game.class).potion1Barbie();
            getWorldOfType(Game.class).removeObject(this);
        }
    }
}
```

G.0.2 Potion1Ken, Potion2Ken

```
public class Potion1Ken extends Actor
{
    public void act()
    {
        if (isTouching(Ken.class))
        {
            getWorldOfType(Game.class).potion1Ken();
            getWorldOfType(Game.class).removeObject(this);
        }
    }
}
```

G.0.3 BrickBlock

```
public class BrickBlock extends Actor
{
    public BrickBlock()
    {
        this(105,30);
    }
    public BrickBlock(int width,int height)
    {
        GreenfootImage image = getImage();
        image.scale(width,height);
        setImage(image);
    }
}
```

H. Fire

H.0.1 Variaveis

```
private GreenfootImage Fire1;  
private GreenfootImage Fire2;  
private int contador;  
private boolean FireBarbie;  
private boolean FireKen;
```

H.0.2 publicFire

```
public Fire()  
{  
    Fire1 = new GreenfootImage("portal.png");  
    Fire2 = new GreenfootImage("portalBack.png");  
    setImage(Fire2);  
}
```

H.0.3 Act

```
public void act()  
{  
    moveFire();  
    fireDeathBarbie();  
    fireDeathKen();  
}
```

H.0.4 moveFire

```
public void moveFire()  
{  
    contador++;  
    if(contador==10)  
    {  
        if(getImage() == Fire1) setImage(Fire2);  
        else setImage(Fire1);  
        contador=0;  
    }  
}
```

H.0.5 deathBarbie

```
public void fireDeathBarbie()  
{  
    if(isTouching(Barbie.class))  
    {
```

```
if(FireBarbie ==false)
{
removeTouching(Barbie.class);
getWorldOfType(Game.class).setBarbieHP(0);
}
FireBarbie = true;
}
else FireBarbie=false;
}
```

H.0.6 deathKen

```
public void fireDeathKen()
{
if(isTouching(Ken.class))
{
if(FireKen == false)
{
removeTouching(Ken.class);
getWorldOfType(Game.class).setKenHP(0);
}
FireKen=true;
}
else FireKen=false;
}
```

I. Makeup

I.0.1 Heel1, Mascara1

```
public class Heel1 extends Makeup
{
    public void act()
    {
        leftToRight();
    }
    public void leftToRight()
    {
        move(5);
        if(getX() == getWorld().getWidth() - 10) getWorld().removeObject(this);
    }
}
```

I.0.2 Heel2, Mascara2

```
public class Mascara2 extends Makeup
{
    public void act()
    {
        rightToLeft();
    }
    public void rightToLeft()
    {
        move(-5);
        if(getX() == 0) getWorld().removeObject(this);
    }
}
```

J. Players

```
public class Players extends Actor
{
    private Boolean sound;
    public void hitAudio()
    {
        if(isTouching(Makeup.class))
        {
            if (sound==false)
            {
                Greenfoot.playSound("hitSound1.mp3");
            }
            sound=true;
        }
        else sound=false;
    }
}
```

K. Barbie

K.0.1 Variaveis

```
private int speed = 6; private GreenfootImage image1, image2;  
private boolean jumping; private int jumpStrength = 20;  
private int verticalSpeed; private int gravity = 2;  
private boolean checkMakeup; private boolean sound;
```

K.0.2 publicBarbie

```
public Barbie() {  
    image1 = new GreenfootImage("barbie1back.png");  
    image2 = new GreenfootImage("barbie1.png");  
}
```

K.0.3 Act

```
public void act()  
{  
    checkFall();  
    checkKeys();  
    checkRightWall();  
    checkLeftWall();  
    checkCeiling();  
    catchDiamonds();  
    makeupHit();  
    hitAudio();  
}
```

K.0.4 HP

```
public int getHP()  
{  
    return getWorldOfType(Game.class).getBarbieHP();  
}
```

K.0.5 HitAudio

```
public void hitAudio()  
{  
    if(isTouching(Makeup.class))  
    {  
        if (sound==false)  
        {  
            Greenfoot.playSound("hitSound2.mp3");  
        }  
    }  
}
```

```

sound=true;
}
else sound=false;
}

```

K.0.6 keyControl

```

private void checkKeys()
{
if(Greenfoot.isKeyDown("w") && jumping == false)
{
jump();
Greenfoot.playSound("Jump.mp3");
}
if(Greenfoot.isKeyDown("d"))
{
setLocation(getX() + speed, getY());
setImage(image2);
}
if(Greenfoot.isKeyDown("a"))
{
setLocation(getX() - speed, getY());
setImage(image1);
}
}

```

K.0.7 Movements

```

private void checkFall()
{
if (onGround() == true)
verticalSpeed = 0;
else
fall();
}

private void fall()
{
setLocation(getX(), getY() + verticalSpeed);
if(verticalSpeed <= 12)
verticalSpeed = verticalSpeed + gravity;
jumping = true;
}

private boolean onGround()
{
int spriteHeight = getImage().getHeight();
int lookForGround = spriteHeight/2;
Actor ground = getOneObjectAtOffset(0, lookForGround, BrickBlock.class);
}

```

```

if(ground == null)
{
jumping = true;
return false;
}
else{
moveToGround(ground);
return true;
}
}
private void moveToGround(Actor ground)
{
int groundHeight = ground.getImage().getHeight();
int newY = ground.getY() - (groundHeight + getImage().getHeight())/2;
setLocation(getX(), newY);
jumping = false;
}
private boolean checkCeiling()
{
int spriteHeight = getImage().getHeight() + 1;
int lookForCeiling = spriteHeight/2;
Actor ceiling = getOneObjectAtOffset(0, -lookForCeiling, BrickBlock.class);
if(ceiling != null)
{
bopHead(ceiling);
return true;
}
else{
return false;
}
}
private void bopHead(Actor ceiling)
{
int ceilingHeight = ceiling.getImage().getHeight();
int newY = ceiling.getY() + (ceilingHeight + getImage().getHeight())/2;
setLocation(getX(), newY);
jumping = false;
verticalSpeed = 2;
}
private boolean checkRightWall()
{
int spriteWidth = getImage().getWidth() + 1;
int lookForRightWalls = spriteWidth/2;
Actor rWall = getOneObjectAtOffset(lookForRightWalls, 0, BrickBlock.class);
if(rWall == null)
{
return false;
}
}

```

```

    }
    else{
        stopByRWall(rWall);
        return true;
    }
}
private void stopByRWall(Actor rWall)
{
    int wallWidth = rWall.getImage().getWidth();
    int newX = rWall.getX() - (wallWidth + getImage().getWidth())/2;
    setLocation(newX, getY());
}
private boolean checkLeftWall()
{
    int spriteWidth = getImage().getWidth() + 1;
    int lookForLeftWalls = -spriteWidth/2;
    Actor leftWall = getOneObjectAtOffset(lookForLeftWalls, 0, BrickBlock.class);
    if(leftWall == null)
    {
        return false;
    }
    else
    {
        stopByLeftWall(leftWall);
        return true; }
    }
private void stopByLeftWall(Actor leftWall){
    int wallWidth = leftWall.getImage().getWidth();
    int newX = leftWall.getX() + (wallWidth + getImage().getWidth())/2;
    setLocation(newX, getY()); }
private void jump()
{
    verticalSpeed = verticalSpeed - jumpStrenght;
    jumping = true;
    fall(); }
}

```

K.0.8 Vida, Pontos

```

public void catchDiamonds()
{
    if(isTouching(DiamondBarbie.class))
    {
        getWorldOfType(Game.class).addPointsScore(10);
        removeTouching(DiamondBarbie.class);
    }
}

```

```
public void makeupHit()
{
if(isTouching(Makeup.class))
{
if(checkMakeup==false)
{
getWorldOfType(Game.class).removeLifeBarbie(-10);
}
checkMakeup=true;
}
else checkMakeup =false;
}
```

L. Ken

L.0.1 Variaveis

```
private int speed = 6;  
private GreenfootImage image1, image2;  
private boolean jumping;  
private int jumpStrength = 20;  
private int verticalSpeed;  
private int gravity = 2;  
private boolean checkMakeup;
```

L.0.2 PublicKen

```
public Ken()  
{  
    image1 = new GreenfootImage("ken2back.png");  
    image2 = new GreenfootImage("ken2.png");  
}
```

L.0.3 Act

```
public void act()  
{  
    checkFall();  
    checkKeys();  
    checkRightWall();  
    checkLeftWall();  
    checkCeiling();  
    catchDiamonds();  
    makeupHit();  
    super.hitAudio();  
}
```

L.0.4 HP

```
public int getHP()  
{  
    return getWorldOfType(Game.class).getKenHP();  
}
```

L.0.5 keyControl

```
private void checkKeys()  
{  
    if(Greenfoot.isKeyDown("up") && jumping == false)  
    {
```

```

jump();
Greenfoot.playSound("Jump.mp3");
}
if(Greenfoot.isKeyDown("right"))
{
setLocation(getX() + speed, getY());
setImage(image2);
}
if(Greenfoot.isKeyDown("left"))
{
setLocation(getX() - speed, getY());
setImage(image1);
}
}

```

L.0.6 Movements

```

private void checkFall()
{
if (onGround() == true)
verticalSpeed = 0;
else
fall();
}
private void fall()
{
setLocation(getX(), getY() + verticalSpeed);
if(verticalSpeed <= 12)
verticalSpeed = verticalSpeed + gravity;
jumping = true;
}
private boolean onGround()
{
int spriteHeight = getImage().getHeight() + 1;
int lookForGround = spriteHeight/2;
Actor ground = getOneObjectAtOffset(0, lookForGround, BrickBlock.class);
if(ground == null)
{
jumping = true;
return false;
}
else
{
moveToGround(ground);
return true;
}
}

```

```

private void moveToGround(Actor ground)
{
    int groundHeight = ground.getImage().getHeight();
    int newY = ground.getY() - (groundHeight + getImage().getHeight()) / 2;
    setLocation(getX(), newY);
    jumping = false;
}
private boolean checkCeiling()
{
    int spriteHeight = getImage().getHeight() + 1;
    int lookForCeiling = spriteHeight / 2;
    Actor ceiling = getOneObjectAtOffset(0, -lookForCeiling, BrickBlock.class);
    if(ceiling != null)
    {
        bopHead(ceiling);
        return true;
    }
    else
    {
        return false;
    }
}
private void bopHead(Actor ceiling)
{
    int ceilingHeight = ceiling.getImage().getHeight();
    int newY = ceiling.getY() + (ceilingHeight + getImage().getHeight()) / 2;
    setLocation(getX(), newY);
    jumping = false;
    verticalSpeed = 2;
}
private boolean checkRightWall()
{
    int spriteWidth = getImage().getWidth() + 1;
    int lookForRightWalls = spriteWidth / 2;
    Actor rWall = getOneObjectAtOffset(lookForRightWalls, 0, BrickBlock.class);
    if(rWall == null)
    {
        return false;
    }
    else{
        stopByRWall(rWall);
        return true;
    }
}
private void stopByRWall(Actor rWall)
{
    int wallWidth = rWall.getImage().getWidth();
    int newX = rWall.getX() - (wallWidth + getImage().getWidth()) / 2;

```

```

        setLocation(newX, getY());
    }
    private boolean checkLeftWall()
    {
        int spriteWidth = getImage().getWidth() + 1;
        int lookForLeftWalls = -spriteWidth/2;
        Actor leftWall = getOneObjectAtOffset(lookForLeftWalls, 0, BrickBlock.class);
        if(leftWall == null)
        {
            return false;
        }
        else
        {
            stopByLeftWall(leftWall);
            return true;
        }
    }
    private void stopByLeftWall(Actor leftWall)
    {
        int wallWidth = leftWall.getImage().getWidth();
        int newX = leftWall.getX() + (wallWidth + getImage().getWidth())/2;
        setLocation(newX, getY());
    }
    private void jump()
    {
        verticalSpeed = verticalSpeed - jumpStrenght;
        jumping = true;
        fall();
    }
}

```

L.0.7 Vida, Pontos

```

public void catchDiamonds()
{
    if(isTouching(DiamondKen.class))
    {
        getWorldOfType(Game.class).addPointsScore(10);
        removeTouching(DiamondKen.class);
    }
}
public void makeupHit()
{
    if(isTouching(Makeup.class))
    {
        if(checkMakeup==false)
        {
            getWorldOfType(Game.class).removeLifeKen(-10);
        }
    }
}

```

```
checkMakeup=true;  
}  
else checkMakeup =false;  
}  
}
```