IFFAR CAMPUS SANTO AUGUSTO

CURSO TÉCNICO DE INFORMÁTICA

DISCIPLINA DE PROGRAMAÇÃO III

RELATÓRIO SOBRE JOGO DESENVOLVIDO NO GREENFOOT – MerchandiseCatcher

Diego Rockenbach

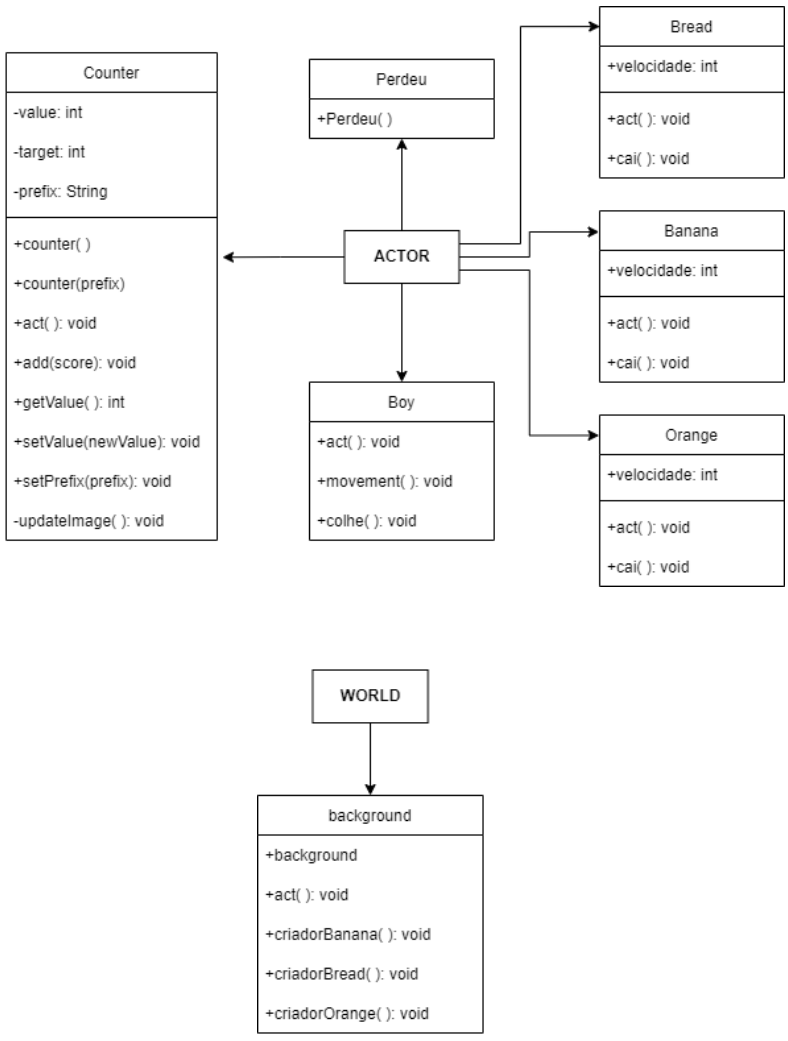
Santo Augusto, RS

Junho, 2022

Descrição do jogo

Neste jogo, o jogador controla um menino com um carrinho de compras para a esquerda e para a direita, no qual seu objetivo é pegar todas os produtos (pães, laranjas e bananas) que estão caindo das prateleiras do mercado.

Diagrama:



Classes

WORLD -> background

```
public class background extends World {
    public background()
    {
        super(600, 400, 1);
        addObject(new Boy(), 300, 300);
        Counter counter = new Counter();
        addObject(counter,77,56);
    }

    public void act(){
        criadorBanana();
        criadorBread();
        criadorOrange();
    }

    public void criadorBanana(){
        if  (Greenfoot.getRandomNumber(600) < 5) {
            int x = Greenfoot.getRandomNumber(500);
            int y = 10;
            addObject(new Banana(), x, y);
        }
    }

    public void criadorBread(){
        if  (Greenfoot.getRandomNumber(600) < 5) {
            int x = Greenfoot.getRandomNumber(500);
            int y = 10;
            addObject(new Bread(), x, y);
        }
    }

    public void criadorOrange(){
        if  (Greenfoot.getRandomNumber(600) < 5) {
            int x = Greenfoot.getRandomNumber(500);
            int y = 10;
            addObject(new Orange(), x, y);
        }
    }
}
```

A classe *background* é responsável pela criação do cenário visível para o jogador, adicionando neste também o objeto *Boy*, cujo o jogador controla, *Counter*, que mostra a pontuação do jogador, e configurando os criadores de mercadorias cujo o jogador terá que impedir que caiam.

O método *act* chama três métodos, respectivamente *criadorBanana*, *criadorBread* e *criadorOrange*, que são responsáveis pela criação de objetos da classe *Banana*, *Bread* e Orange. Em cada um dos três métodos temos uma variável que recebe um número aleatório de 0 a 500 (que será a posição horizontal que a determinada mercadoria irá aparecer aleatoriamente) e possui a variável vertical y com valor fixo de 10.

ACTOR -> Boy

```
public class Boy extends Actor
{
    public void act()
    {
       colhe();
       movement();
    }

    public void movement(){
        if(Greenfoot.isKeyDown ("right")) {
            move(6);
        }
        if(Greenfoot.isKeyDown ("left")) {
            move(-6);
        }
    }

    public void colhe(){
        Actor a = getOneIntersectingObject(Banana.class);
        if(a != null){
            getWorld().removeObject(a);
            Counter counter = (Counter)getWorld().getObjects(Counter.class).get(0);
            counter.add(1);
        }
        Actor b = getOneIntersectingObject(Bread.class);
        if(b != null){
            getWorld().removeObject(b);
            Counter counter = (Counter)getWorld().getObjects(Counter.class).get(0);
            counter.add(1);
        }
        Actor c = getOneIntersectingObject(Orange.class);
        if(c != null){
            getWorld().removeObject(c);
            Counter counter = (Counter)getWorld().getObjects(Counter.class).get(0);
            counter.add(1);
        }
    }
}
```

A classe *boy* é o personagem controlado pelo jogador. Ele possui dois métodos dentro do método *act*, sendo estes *movement* e *colhe*. O método *movement* é responsável pela movimentação do personagem, registrando se a seta esquerda (“left”) ou direita (“right”) está sendo pressionada no teclado do usuário, e, caso estiver, movimentar o personagem para a respectiva direção no plano horizontal. O método *colhe*, por outro lado, é responsável por pegar as frutas que estão caindo. Caso o *Boy* entre em contato com alguma das mercadorias (*Banana*, *Bread* ou *Orange)*, o objeto da mercadoria será removido e a classe *Counter* irá receber mais um ponto.

ACTOR -> Banana/Bread/Orange 

```
public class Banana extends Actor
{
    int velocidade = 5;

    public void act()
    {
        cai();
    }

    public void cai(){
       this.setLocation(this.getX(), this.getY()+velocidade);
       if(isAtEdge()){
         Perdeu p = new Perdeu();
         getWorld().addObject(p, 300, 200);
         Greenfoot.stop();
        }
    }
}
```

As classes *Banana, Bread* e *Orange* são exatamente iguais e servem o mesmo propósito, sua única distinção sendo a imagem do objeto (sendo este respectivamente, de uma banana, pão ou de uma laranja). O método *act* chama o método *cai,* que é responsável por encerrar o jogo criando um objeto da classe *Perdeu* pelo construtor *Perdeu( )*. Este objeto de classe faz aparecer a mensagem de “GAME OVER!” (que aparece nas coordenadas de 300x, 200y) quando alguma mercadoria encosta no final da tela, ou seja, o jogador é incapaz de pegá-la, seguido do método padrão do Greenfoot, o *stop( )* parece encerrar o jogo.

ACTOR -> Perdeu

```
public class Perdeu extends Actor
{
    public Perdeu(){
        GreenfootImage go = new GreenfootImage ("GAME OVER!",80, Color.RED, null);
        setImage(go);
    }
}
```

É a classe chamada pelo método *cai* presente nas classes *Banana, Bread* e *Orange*. Ela possui o método *Perdeu( )* que cria um objeto padrão da classe *GreenfootImage* que possui a mensagem “GAME OVER!”, em tamanho de fonte 80, cor vermelha e cor de fundo nulo.

ACTOR -> Counter 

É uma classe padrão do Greenfoot. Ela é responsável por contar o número de mercadorias pegas pelo jogador, e mostrar estas em um contador crescente no canto superior esquerdo da tela (coordenadas 77x, 56y).

Conceitos utilizados

- **Classes:** São implementações de tipo abstrato de dados, sendo basicamente moldes para criação de objetos. São elas que definem as propriedades (atributos) e os comandos (métodos) dos objetos. Neste jogo há 7 classes, sendo elas *background* (subclasse de *WORLD*), *Boy, Banana, Bread, Orange, Counter* e *Perdeu* (todas subclasses de *ACTOR*).

- **Objetos:** São criados a partir das classes e encapsulam atributos e métodos. Neste jogo há vários objetos, como *Boy, Banana, Bread, Orange, Counter, GreenfootImage*, entre outros.

- **Atributos:** No jogo há atributos nas classes *Banana, Bread* e *Orange*, em todas as ocorrências definindo o int *velocidade*.

- **Métodos:** Métodos são as ações que cada objeto pode realizar no sistema, estando presentes em diversas classes no jogo. Alguns exemplos seriam *colhe* e *movement* (ambos presentes na classe *Boy*), *cai* (presente em todas as classes de mercadoria), *act* que age como uma classe geral do Greenfoot para fazer com que os métodos das classes ajam como um todo, entre outros.

- **Construtores:** Construtores são métodos respectivos de cada classe que são responsáveis por preparar um objeto, quando este está sendo criado, em um estado desejado pronto para ser utilizado. Neste jogo há construtores nas classes *background*, *Perdeu*, *Counter, Banana, Bread* e *Orange*.

- **Herança:** No conceito de herança, classes-filho (ou subclasses) reutilizam métodos e atributos da classe-pai (ou superclasse). Neste jogo, *background* é uma subclasse de *WORLD*, enquanto *Boy, Banana, Bread, Orange, Perdeu* e *Counter* são subclasses de *ACTOR*. O conceito de herança foi firmemente destacado nos diagramas de classe.

- **Instanciação:** A instanciação de uma classe é nada mais do que a criação de um objeto dentro dela. Tais objetos irão compartilhar dos mesmos atributos, embora o conteúdo destes possa ser diferente. A cada criação de objeto, (como podemos ver ocorrer com os construtores *criadorBanana, criadorBread* e *criadorOrange*, por exemplo) ocorre uma instanciação deste objeto.

- **Interação entre objetos:** A interação entre dois objetos ou mais ocorre frequentemente neste, assim como em qualquer jogo. Um exemplo de interação entre dois objetos pode ser observado na colisão das mercadorias com a classe *Boy*, sendo o fruto dessa interação apagar a instanciação do objeto da mercadoria que entrou em contato com o personagem e adicionar um ponto a mais no *Counter*.

- **Encapsulamento:** Toda classe pode ser definida em um de 4 tipos:

    1. default (padrão) – Item visível dentro do pacote
    2. *public* – Item público a todas as classes e objetos do sistema
    3. *protected* – Visível somente dentro do pacote
    4. *private* – Visível somente dentro da própria classe

  O método utilizado para acessar classes do tipo *private* fora dela mesma é através dos *getters* e *setters*. Neste jogo, os atributos e alguns métodos da classe padrão *Counter* são do tipo *private*, portanto justificando o uso de métodos para que se altere o valor destes atributos em outras classes.