EP : Galaxy Raiders

Renato Cordeiro, Jiang Zhi e Alfredo Goldman

MAC0218 - Técnicas de Programação II

1 Introdução

Galaxy Raiders é um jogo inspirado por clássicos da Era de Ouro dos Arcades: Space Invaders (1978), Star Raiders (1979), Asteroids (1979) e Galaga (1981). Esses jogos se popularizaram graças aos fliperamas (máquinas especializadas para jogos eletrônicos) e aos primeiros videogames (dispositivos caseiros para jogos eletrônicos), introduzidos pela empresa americana Atari.

Figura 1: Fliperama com Space Invaders

No ambiente espacial onde o Galaxy Raiders é inserido, o jogador precisa controlar a nave para desviar de objetos que possam colidir na nave e causar dano na sua estrutura e também controlar os canhões de laser para lançar projéteis para destruir os asteroides.

O jogo é composto por duas partes. Um componente para visualização do jogo, chamado de front-end ou interface Web, onde o usuário consegue interagir com o jogo; e um componente de lógica do jogo, chamado de back-end ou API, onde o motor do jogo processa os comandos do usuário, atualiza o estado do jogo, e o envia para visualizar. O repositório do projeto pode ser encontrado em `https://github.com/galaxy-raiders`. Neste EP, modificaremos apenas a API.

A API foi estruturada utilizando o estilo arquitetural hexagonal. Este estilo arquitetural propõe desacoplar a lógica de negócio com dependências externas utilizando estruturas conhecidas como *portas e adaptadores*.

A estrutura de pastas do código é dividida em três pacotes:

- Pacote `core`, com a lógica do jogo. Ele é subdividido em dois pacotes para as funcionalidades de física (`core.physics`) e do jogo (`core.game`).
- Pacote `ports`, com as interfaces para dependências externas, fornecendo um ponto de testabilidade para a lógica do jogo. Ele é subdividido num pacote com portas para a interface do usuário (`ports.ui`).
- Pacote `adapters`, com a implementação das dependências externas. Ele é subdividido em dois pacotes que implementam uma interface de texto (`adapters.tui`) e uma interface web (`adapters.web`) para interagir com o jogo.

2 Tarefa

Este EP está dividido em duas partes:

1. Na parte um, vamos praticar TDD (Test-Driven Development). Os testes já foram criados. Você implementará a física do jogo, disponível no pacote `core.physics`.
2. Na parte dois, vamos praticar adicionar uma funcionalidade a um sistema existente. Você utilizará *herança de classes* da Programação Orientada a Objetos para estender o comportamento do jogo com efeitos de explosão.

Seu grupo deverá implementar as tarefas usando o código e a estrutura de pastas disponibilizada no GitHub em `https://www.github.com/galaxy-raiders`.

2.1 Parte 1 - Implementação da parte física

Na primeira parte, utilizamos TDD para implementar a física básica do jogo, os testes já foram criados para vocês, então o que será necessário é implementar as classes `Point` e `Vector` para rodar o código do jogo. Para tanto, precisamos modificar o código que se encontra em:

- `/app/src/kotlin/galaxyraiders/core/physics/Vector2D.kt`, e
- `/app/src/kotlin/galaxyraiders/core/physics/Point2D.kt`.

2.2 Parte 2 - Extensão da classe

Na segunda parte, utilizamos um conceito importante de POO de herança de classe, vamos fazer uma extensão da classe `SpaceObject`, para criar a classe `Explosion`, essa classe tem como função adicionar efeitos visuais de explosões ao acertar um asteroíde.

A classe deve se chamar `Explosion`. Você pode incluir um parâmetro booleano (`is_triggered`) para indicar que a explosão já aconteceu ou não. A escolha dos parâmetros e das funções é livre. No entanto, *a explosão deve acontecer quando o projétil acertar um asteroide e depois desaparecer*. A informação da explosão é passada pelo front-end por um JSON. Portanto, é necessário modificar partes da API no pacote `core.game`.

Para verificar a funcionalidade da classe `Explosion`, a criação de testes automatizados similares aos implementados para outras classes é recomendado. Ao criar uma classe é importante estender e compatibilizar os testes já existentes. Além disso, é recomendável adicionar novos testes novos onde for necessário. Confira o arquivo `app/build/reports/jacoco/test/html/index.html` para verificar a cobertura de testes.

3 Entrega

O EP pode ser feito em *até duas pessoas*.

A entrega deve feita por uma url do repositório de git. Cada pessoa deve fazer um *fork* do repositório `https://github.com/galaxy-raiders/galaxy-raiders-api` e colocar a url do repositório no eDisciplinas. *Faça a entrega apenas uma vez. Lembre-se de indicar quem é a sua dupla na entrega.*

No repositório, é necessário:

- a alteração do código do pacote `physics` do jogo; e
- a criação da classe `Explosion` dentro da pasta `/core/game/`. O arquivo precisará ser nomeado `Explosion.kt`. Essa inclusão exigirá alterar o código do motor do jogo e conseguir mandar o dado das explosões dentro do JSON para o front-end.

O front-end já foi modificado para exibir explosões. Portanto, *não será necessário alterar o front-end*. Se você quiser incrementar o jogo, o front-end encontra-se no repositório `https://github.com/galaxy-raiders/galaxy-raiders-web`.

4 Avaliação

O EP será avaliado a partir da implementação do código de cada parte da tarefa:

1. Implementação da física do jogo com TDD (5.0)
    - Código compila sem erros e warnigs: 1.0
    - Boas práticas de programação e clareza do código: 1.0
    - Código executa sem erros e produz os resultados desejados: 3.0
2. Implementação da herança de classe para Explosion (5.0)
    - Código compila sem erros e warnigs: 1.0
    - Boas práticas de programação e clareza do código: 1.0
    - Código executa sem erros e produz os resultados desejados: 3.0