UNIVERSIDADE da MADEIRA

ESCOLA SUPERIOR DE TECNOLOGIAS E GESTÃO

Ano Letivo 2022/2023

Curso Técnico Superior Profissional em: Tecnologias e Programação de Sistemas de Informação

2 º **Ano**/ 1 º **Semestre**

Unidade Curricular: Arquitetura de Dispositivos **Data Entrega**: 12/12/2022

Docente: David Jardim **Época:** Normal

SPACE INVADERS

Considere um jogo de computador a 2 dimensões onde o objetivo é controlar uma nave espacial e disparar lasers para destruir naves inimigas sem ser atingido pelo fogo inimigo. Tenha em consideração a seguinte imagem a título de exemplo:

Para o desenvolvimento deste projeto será utilizada a framework LibGDX (https://libgdx.badlogicgames.com/). Juntamento com o enunciado será disponibilizado um template para o projeto. O jogo terá as seguintes classes:

- **Game** – classe principal onde toda a lógica do jogo será implementada
- ***Ship*** – classe abstrata para representar as naves espaciais
  - **PlayerShip** – classe concreta para representar a nave do jogador
  - **SmallShip** – classe concreta para representar a nave pequena dos inimigos
  - **MediumShip** – classe concreta para representar a nave média dos inimigos
  - **LargeShip** – classe concreta para representar a nave maior dos inimigos
  - **shoot –** método abstrato para efetuar um disparo diferente consoante o tipo de nave
- **Fleet** – classe que irá representar a frota (lista) de naves inimigas
- **Laser** – classe para representar o laser que pode ser disparado pela nave do jogador ou naves inimigas
- **Animator** – classe para animar as *spritesheets* das naves e do laser
- **BackgroundManagement** – classe para definir o fundo do ecrã do jogo

Dentro das naves inimigas há 3 tipos de unidades: **SmallShip**, **MediumShip** e **LargeShip**. Cada frota pode ter um número total de 24 elementos, com composições diferentes. Por exemplo, 8 naves pequenas, 8 naves médias e 8 naves grandes. Cada nave possui os seguintes atributos:

- animator
- posição X e Y
- valor de ataque (poder de ataque será de 5, 10, 20 pontos respetivamente para as **SmallShip, MediumShip e LargeShip**)
- boundingBox do tipo Rectangle
- collided que indica se a nave colidiu com um laser

O jogador controla a nave em 2 direções (esquerda e direita) para fugir dos tiros inimigos. O disparo é efetuado com a tecla do espaço, se o disparo do jogador atingir uma nave inimiga essa nave deve ser removida. Por cada nave atingida a pontuação do jogador deve ser atualizada tendo em conta o valor de ataque da nave abatida.

As naves inimigas ao longo do tempo deverão efetuar um disparo de forma aleatória, se o jogador for atingido por um disparo inimigo deverá ser subtraído à sua "vida" o valor correspondente. O jogo termina em 2 situações:

- Se a vida do jogador chegar a zero
- Se todas as naves forem abatidas pelo jogador

O projeto deverá ser feito no máximo por grupos de 2 elementos e submetido em .zip no formato **P1_nºaluno1_nºaluno2** no Moodle e os alunos deverão efetuar uma pequena defesa do trabalho.

Tendo em conta a notação UML, desenhe o diagrama de classes representando as classes que foram implementadas e as relações entre as mesmas (composição, herança, etc).

PARTE 1

1. Implemente todas as classes necessárias para representar as diferentes naves existentes no jogo aplicando o conceito de Herança e de Polimorfismo. Todas as classes possuem atributos e métodos idênticos, onde deverão ser declarados/implementados? (3 valores)
2. Na classe **PlayerShip** implemente os métodos necessários para:
    a. Controlar a posição da nave em 2 direções (esquerda e direita) (2 valores)
    b. Disparar o laser com a tecla espaço de forma a atingir os inimigos. (2 valores)
3. Implemente a classe **Laser** para representar um tiro. Quais deverão ser os atributos desta classe? (2 valores)
4. Utilize a classe **BitmapFont** para apresentar no ecrã a vida e a pontuação do jogador. (1 valor)

PARTE 2

1. Implemente a classe **Fleet** que irá conter todas as naves inimigas ordenadas pelo seu poder de ataque. A disposição das naves deverá ser feita de forma ordenada tal como na imagem acima. (3 valores)
2. Na classe **Fleet** implemente o código necessário para que de 2 em 2 segundos (utilize a classe **TimerTask** para temporizar uma tarefa) uma nave inimiga escolhida de forma aleatória efetue um disparo para atingir o jogador implementando o método abstrato. (2 valores)
3. Implemente o código necessário para gerir as colisões utilizando o atributo boundingBox do tipo **Rectangle**
    a. Colisões entre o tiro do jogador e as naves inimigas (2 valores)
    b. Colisões entre os tiros das naves inimigas e a nave do jogador (2 valores)
4. Utilize a classe **BitmapFont** para apresentar no ecrã uma mensagem de vitória caso o jogador destrua todas as naves inimigas ou uma mensagem de derrota caso a sua vida chegue a zero. (1 valor)