«Batalha Naval»

11019915-Allan Calixto Severo da Silva,
11035315-Ana Laura Belotto Claudio,
11014515-Gilmar Correia Jeronimo,
11110415-Nickolas Lopes Pimenta

1 de Agosto de 2018

Turma: A1 - Matutino

1 Introdução

O projeto é o jogo de estratégia chamado *BattleShip* (Batalha Naval) e foi implementada utilizando conceitos de programação orientada à objetos, na linguagem Java. No projeto foram implementados duas opções de jogo, uma *single-player*, cujo o jogandor joga conta o computador, e uma com a opção de *versus mode*, ou seja, o *player* 1 contra o *player* 2. Nesse jogo ganha quem destruir primeiro todas as embarcações do adversário, tentanto advinhar onde elas estão escondidas no campo, com uma tentativa (tiro) por vez.

Caso o jogador acerte um barco, ele tem a chance de continuar atirando para adivinhar a posição do barco, caso erre o tiro a vez passa para o oponente. Para esse projeto foi desenvolvido uma interface gráfica para comunicar os jogadores de maneira mais interativa, no qual é possível utilizar o mouse para setar posições, mudar orientações do barco e atirar. Para isso foi feito uso do JavaFX, garantindo a jogabilidade.

2 Descrição das classes

- Classe Principal: inicia o jogo, ou seja, inicia a interface gráfica e as subclassses que fazem o jogo funcionar.
- Classe Tela: define qual é a cena que vai ser mostrada no JavaFx, além do título, imagem e fonte. É uma classe abstrata. É a classe-mãe de outras classes, cada uma define uma cena a ser mostrada, temos as subclasses Regras, TelaMain, Escolher, GameScene e ButtonClass.
- Classe Regras (subclasse de Tela): é uma sub-classe da classe Tela. Responsável pelas regras do jogo.

- Classe ButtonClass (subclasse de Tela): tem como objetivo retornar um botão. É enviado uma cena e o botão retorna qual é a próxima cena, o texto e a fonte.

- Classe GameScene (subclasse de Tela): define a mecânica do jogo, qual o evento a ser realizado pelo Player, por exemplo quem começa jogando, qual a posição do barco, entre outras funções.

- Classe Escolher (subclasse de Tela): mostra na tela do computador os botões dos modos de jogo, a opção de voltar, etc.

- Classe TelaMain (subclasse de Tela): responsável pela tela inicial do jogo.

- Classe GamePlayerFunctions: é uma interface que define o que cada Player deve ter, como o método atirar, verificar o tiro que levou, validar posição para uma dada coordenada do barco. É implementada por Computer e Player.

- Classe Computer (subclasse de GamePlayerResources e implementa GamePlayerFunctions): realiza a mecânica de jogo do Player Computador. Ela cria uma matriz de probabilidade e por meio dela são definido onde serão os tiros.

- Classe Player(subclasse de GamePlayerResources e implementa GamePlayerFunctions): valida as coordenadas de tudo que o jogador clicar com o mouse ou com o teclado.

- Classe GamePlayerResources (subclasse de VBox e implementa GamePlayerFunctions): É a uma das classes fundamentais do jogo, pois é ela que vai definir toda a comunicação da interface visual com as embarcações. Essa classe extende uma "V.box"que vai definir a imagem de como o player vai aparecer, ela também cria as matrizes, verifica se tem embacação na célula ou não, verifica se eu já a célula já foi jogada, setar as embacações nas posições, identifica qual embacação foi atingida se o palpite do jogador for certo, e verifica a vida dos players.

- Classe Shot: é uma classe do tipo Enun, que enumera uma lista de tiros possíveis no jogo, que seriam: afundou, atingiu e errou.

- Classe Coordenate: Essa classe vai setar uma coordenada x,y e retorna-lá.

- Classe Célula: Essa classe vai criar um dos quadrados da matriz e verifica se existe ou não alguma embacação naquela posição, caso exista ela vai colorir para o jogador saber que ele acertou alguma coisa, caso contrário ela modifica para cor de célula já jogada.

- Classe Embarcacao: é a classe-mãe de todas as embarcações, Submarino, PortaAviao, NavioTanque e Contratorpedeiro. É responsável pelo tamanho do barco, a posição e se está na vertical ou não. É uma classe abstrata.

- Classe Contratorpedeiro (subclasse de Embarcacao): Instancia a embarcação Contratorpedeiro, com os métodos da classe-mãe.
- Classe NavioTanque (subclasse de Embarcações): Instancia a embarcação NavioTanque, com os métodos da classe-mãe.
- Classe PortaAviao (subclasse de Embarcações): Instancia a embarcação PortaAviao, com os métodos da classe-mãe.
- Classe Submarino (subclasse de Embarcações): Instancia a embarcação Submarino, com os métodos da classe-mãe.
- Classe Exception: Responsável por lançar uma exceção caso o jogador tente jogar na mesma que ele já tenha jogado.

3 Conceitos de orientação a objetos aplicados

Os seguintes conceitos foram utilizados no projeto:

- Construtores
- Herança
- Interface
- Encapsulamento
- Classe Abstrata
- Tratamento de exceções
- Padrão de Software

Todos esses conceitos foram implementados de forma adequada utilizando os conceitos de POO, seja construtores para quando uma classe for chamada, herança para ter uma super classe com atributos e métodos com sub-classes que possam ter acesso a esses métodos, encapsulamento com os atributos e métodos com as restrições de acesso adequadas, tratamento de exceções para caso se jogador tentar fazer uma jogada ilegal e também foi utilizado uma interface para padronizar as funções dos jogadores, tanto player quanto computer.

Foi utilizado um padrão de software para controlar a mecânica dos jogadores, o *Strategy*. Esse padrão é geralmente aplicado quando muitas classes relacionam-se, no caso do projeto temos várias classes que atuam de forma diferente, mas se relacionam para que o jogo funcione. No projeto o *strategy* é utilizado para chamar hora um jogador humano, hora uma máquina. Com a interface criada é possível sobreescrever os métodos e setar para cada uma das classes implementadas os métodos atira, verifica posição e retorna tiro, ou seja, método que define uma coordenada para ser atirada, método que auxilia no posicionamento dos barcos e um que retorna qual tipo de tiro foi dado.

4 Participação de cada integrante do grupo

Ana Laura: responsável pela classe das embarcações, pela inteligência do *player* computador, ajustes no códigos das classes e revisão do relatório e criaçãoo do diagrama UML.
Gilmar Correia: responsável pela interface gráfica, integração das classes como um todo para o funcionamento do projeto e revisão do relatório.
Nickolas Pimenta: responsável pela classe do jogador e pelo desenvolvimento do relatório.
Allan Calixto: Responsável no suporte do trabalho em geral.