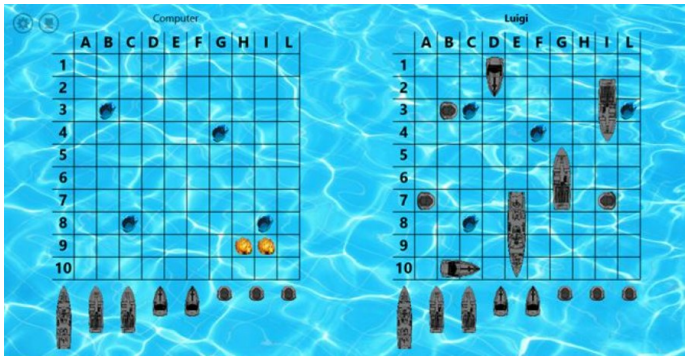
Trabalho Prático

O trabalho consiste no desenvolvimento de uma versão simplificada do jogo batalha naval. Este jogo é jogado em um tabuleiro com N linhas e M colunas. Cada posição desse tabuleiro é um quadrado que pode conter água ou parte de um navio. Um jogador informa ao outro a posição (linha, coluna) do quadrado alvo do disparo. O jogo termina quando um dos jogadores afunda todas as embarcações do seu oponente. Desenvolva um jogo de batalha naval, em que um JogadorHumano dispute contra o computador (JogadorComputador).



Exemplo do jogo Batalha Naval:

Jogo Batalha Naval Clássico no Jogos 360

1) O programa deverá gerar um tabuleiro para o JogadorComputador com 10 linhas e 10 colunas. O programa deverá ler de um arquivo (frotaComputador.txt) a posição de cada uma das seguintes embarcações:

Embarcação	**Quantidade**	**Representação no tabuleiro**
Submarino	4	S
Hidroavião	3	HH
Cruzador	2	CCC
Encouraçado	1	EEEE
Porta-aviões	1	PPPPP

Exemplo de um tabuleiro com as embarcações:

	0	1	2	3	4	5	6	7	8	9
0	A	A	A	A	A	S	A	A	A	S
1	H	H	A	A	A	A	A	H	H	A
2	A	A	A	A	A	A	A	A	A	A
3	H	H	A	A	A	A	C	C	C	A
4	A	A	A	A	A	A	A	A	A	A
5	A	C	C	C	A	A	A	A	A	A
6	A	A	A	A	A	S	A	A	S	A
7	A	A	A	A	A	A	A	A	A	A
8	E	E	E	E	A	A	A	A	A	A
9	A	A	A	A	A	P	P	P	P	P

Legenda:

- A: água

Exemplo arquivo frotaComputador.txt

```
Porta-aviões;9;5
Encouraçado;8;0
Cruzador; 5;1
Cruzador;3;6
Hidroavião;1;0
Hidroavião;1;7
Hidroavião;3;0
Submarino; 0;5
Submarino;0;9
Submarino;6;5
Submarino;6;8
```

2) O jogadorHumano deverá informar seu nome completo e o programa deverá gerar um *nickname* (apelido) para ele com o sobrenome (último nome) e as iniciais dos demais nomes (primeiro nome e nomes do meio). Esse *nickname* deve ser usado pelo programa em todas as mensagens dirigidas ao jogadorHumano.
3) O tabuleiro do JogadorHumano terá 10 linhas e 10 colunas. O JogadorHumano deverá escolher (via teclado) as posições das seguintes embarcações:

Embarcação	**Quantidade**	**Representação no tabuleiro**
Submarino	3	S
Hidroavião	2	HH
Cruzador	2	CCC
Encouraçado	1	EEEE
Porta-aviões	1	PPPPP

Caso seja informada uma posição que não caiba a embarcação inteira, o programa deverá solicitar que seja informada uma nova posição. Esse processo deve ser repetido até que seja informada uma posição válida. O programa não deve permitir que duas embarcações sejam posicionadas na mesma posição.

4) Após posicionar as embarcações, o jogo terá as seguintes etapas:

- O programa deverá mostrar o estado atual do tabuleiro do jogadorComputador (sem mostrar, é claro, as posições das embarcações). O jogadorHumano disparará um tiro, indicando (via teclado) a posição do alvo por meio dos números da linha e da coluna que definem a posição no tabuleiro. O programa deverá informar se o tiro acertou alguma embarcação e deverá mostrar o estado atual do tabuleiro, isto é, mostrar o tabuleiro indicando os tiros feitos na água e os tiros que acertaram embarcações. Caso o tiro tenha acertado a água, a vez de jogar passa para o JogadorComputador. Caso contrário, o jogadorHumano poderá dar novos tiros enquanto ele estiver acertando partes de embarcações do seu oponente (Obs: Cada disparo que um jogador faz deve ser feito em um dos quadrados do tabuleiro do outro jogador. Se um jogador informar uma posição fora dos limites do tabuleiro, deve ser solicitada uma nova posição de disparo. Cada jogador não pode atirar no mesmo lugar mais de uma vez. Se um jogador informar uma posição de disparo que já foi utilizada anteriormente, o programa deve solicitar uma nova posição de disparo).
- O JogadorComputador, na sua vez de jogar, seguirá o seguinte procedimento: O programa deverá mostrar o estado atual do tabuleiro do jogadorHumano (sem mostrar, é claro, as posições das embarcações). Na vez do JogadorComputador o programa deverá escolher **aleatoriamente** as coordenadas da posição alvo (linha e coluna). O programa deverá informar se o tiro acertou ou não uma embarcação e mostrar o estado atual do tabuleiro do jogadorHumano. Caso o tiro tenha acertado a água, a vez de jogar passa para o JogadorHumano. Caso contrário, o jogadorComputador poderá dar novos tiros enquanto ele estiver acertando partes das embarcações do seu oponente (obs: Cada disparo que um jogador faz deve ser feito em um dos quadrados do tabuleiro do outro jogador. Assim, deve ser sorteada uma posição dentro dos limites do tabuleiro. Se for informada uma posição de disparo que já foi utilizada anteriormente, o programa deve sortear uma nova posição de disparo, até que uma posição válida seja sorteada).

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
0	A	T	T	A	A	A	A	A	A	A
1	A	A	A	A	A	A	A	A	A	A
2	A	A	A	A	A	A	A	A	A	A
3	A	A	A	A	A	A	A	A	A	A
4	A	A	A	A	A	A	A	A	A	A
5	A	A	A	A	A	A	A	A	A	A
6	A	A	A	A	A	A	A	A	A	A
7	A	A	A	A	A	A	A	A	A	A
8	A	A	A	X	A	A	A	A	A	A
9	A	A	A	X	A	A	A	A	A	A

Legenda:

- A: água
- T: tiro acertou parte de uma embarcação
- X: tiro acertou na água

- Cada jogador ganhará um ponto para cada tiro que acertar parte da embarcação do seu oponente. O jogo terminará quando um jogador atingir a pontuação máxima do jogo, isto é, quando todas as embarcações de seu oponente forem afundadas. O programa deve informar qual jogador venceu o jogo. Se o jogadorHumano que tiver vencido, deve ser informado seu *nickname*.

- Por fim, o programa deve escrever em um arquivo texto (jogadas.txt) a sequência de tiros do jogador vencedor, isto é, escrever as posições (linha e coluna) dos tiros na ordem em que eles foram feitos.

    - Exemplo: Tiro 1: (0, 1)
      Tiro 2: (8, 3)

5) O programa deverá ter OBRIGATORIAMENTE as seguintes classes:

Classe Embarcacao

- Campos: nome (tipo string)

  tamanho (tipo int)

Dica: a sigla colocada no tabuleiro pode ser a posição 0 do nome da embarcação

- Construtor: recebe como parâmetro o nome da embarcação e seu tamanho
- Propriedades

Classe Posicao

- Campos: linha (tipo int)

  coluna (tipo int)

- Propriedades

Classe JogadorHumano

- Campos: tabuleiro (tipo char[,])

  pontuacao (tipo int)

  numTirosDados (tipo int)

  posTirosDados (tipo Posicao[])

  nickname (tipo string)

- Construtor: o construtor deverá receber como parâmetro o número de linhas e colunas do tabuleiro do jogador, e também o nome completo do jogador. Todos os atributos devem ser inicializados: i) O tabuleiro deve ser instanciado e todas as posições devem receber o símbolo de água ii) A pontuação deve iniciar com zero; iii) o número de tiros dados (numTirosDados) deve iniciar com zero, iv) o vetor com as posições dos tiros dados (posTirosDados) deve ser instanciado, v) o nickname deve ser gerado (utilize o método GerarNickname).
- Propriedades
- GerarNickname: este método deve receber o nome completo do jogador como parâmetro (parâmetro do método: string) e deve gerar o seu *nickname*.
- EscolherAtaque: este método retornará a Posicao de um tiro (tipo de retorno do método: Posicao). Neste método o usuário irá escolher (via teclado) as coordenadas do tiro. O método deverá também, adicionar o tiro dado no vetor posTirosDados.

  Validações: 1) Se for informada uma posição fora dos limites do tabuleiro, deverá ser solicitada uma nova posição de disparo. 2) Se for informada uma posição de disparo que já foi utilizada anteriormente (verifique no vetor posTirosDados), o programa deverá solicitar uma nova posição de disparo.

- ReceberAtaque: este método receberá a Posicao de um tiro como parâmetro (parâmetro do método: Posicao). Deve-se atualizar o tabuleiro com este tiro, caso alguma embarcação seja atingida o método retornará verdadeiro, caso contrário retornará falso.
- Método ImprimirTabuleiroJogador: imprime o tabuleiro para o jogador, mostrando inclusive o posicionamento de todas as embarcações
- Método ImprimirTabuleiroAdversario: imprime o tabuleiro para o adversário, isto é, não deve ser informado o posicionamento das embarcações.
- Método AdicionarEmbarcacao: recebe como parâmetro uma Embarcacao e sua posição inicial (tipo Posicao). A Embarcacao deve ser adicionada no tabuleiro caso seja possível adicioná-la a partir da posição informada, isto é, **a embarcação inteira deve caber no tabuleiro.** Caso seja possível adicionar a embarcação, o método deverá retornar verdadeiro, caso contrário retornará falso (Não devem ser adicionadas duas embarcações na mesma posição).

Classe JogadorComputador

- Campos: tabuleiro tabuleiro (tipo char[,])

  pontuacao (tipo int)

  numTirosDados (tipo int)

  posTirosDados (tipo Posicao[])

- Construtor: o construtor deverá receber como parâmetro o número de linhas e colunas do tabuleiro do jogador. Todos os atributos devem ser inicializados: i) O tabuleiro deve ser instanciado e todas as posições devem receber o símbolo de água; ii) A pontuação deve iniciar com zero; iii) o número de tiros dados (numTirosDados) deve iniciar com zero, iv) o vetor com as posições dos tiros dados (posTirosDados) deve ser instanciado.
- Propriedades
- EscolherAtaque: este método retornará a Posicao de um tiro (tipo de retorno do método: Posicao). O programa deverá gerar **aleatoriamente** a posição de um tiro (Utilize o método Next da classe Random). O método deverá também, adicionar o tiro dado no vetor posTirosDados. Validações: Caso a posição gerada aleatoriamente esteja fora dos limites do tabuleiro ou já tenha sido utilizada anteriormente (verifique no vetor posTirosDados), o programa deverá gerar uma nova posição de disparo.
- ReceberAtaque: este método receberá a Posicao de um tiro como parâmetro (parâmetro do método: Posicao). Deve-se atualizar o tabuleiro com este tiro, caso alguma embarcação seja atingida o método retornará verdadeiro, caso contrário retornará falso.
- Método ImprimirTabuleiroJogador: imprime o tabuleiro para o jogador, mostrando inclusive o posicionamento de todas as embarcações
- Método ImprimirTabuleiroAdversario: imprime o tabuleiro para o adversário, isto é, não deve ser informado o posicionamento das embarcações.
- Método AdicionarEmbarcacao: recebe como parâmetro uma Embarcacao e sua posição inicial (tipo Posicao). A Embarcacao deve ser adicionada no tabuleiro caso seja possível adicioná-la a partir da posição informada, isto é, **a embarcação inteira deve caber no tabuleiro.** Caso seja possível adicionar a embarcação, o método deverá retornar verdadeiro, caso contrário retornará falso (Não devem ser adicionadas duas embarcações na mesma posição).

Classe BatalhaNaval

- Classe que terá o método Main e irá simular um jogo de Batalha Naval, para tanto deve ser instanciado um objeto JogadorHumano e um objeto JogadorComputador.
- O grupo poderá criar outros métodos nesta classe para modularizar seu código.

Informações:

- **Trabalho em grupo de 3 alunos. Não serão aceitos grupos com mais ou menos integrantes.**
- **Valor: 10 pontos**
- **Novos métodos e campos podem ser adicionados nas classes, caso o trio julgue necessário. Entretanto, os métodos e campos descritos nesta especificação não devem ser modificados.**
- **No desenvolvimento do programa só podem ser utilizados comandos vistos nas aulas.**
- **O código fonte não deve ter nenhum comentário.**
- **A entrega do código (pasta do projeto) deverá ser feita por meio do Canvas na forma de um único arquivo compactado (extensão .zip). Apenas um integrante de cada grupo deverá entregar no Canvas o código.**
- **Data de entrega: 08/12 às 17h**
- **Arguição oral: nos dias 09/12, 10/12 a professora fará perguntas para os integrantes de cada grupo. Cada aluno será avaliado de acordo com seu conhecimento do trabalho. Por isso, cada aluno deverá conhecer COMPLETAMENTE o funcionamento do código que o grupo implementou. A nota será INDIVIDUAL apesar do trabalho ser em grupo. Alunos serão penalizados na entrevista de forma individual caso não consigam responder às perguntas.**
- **O grupo será penalizado por igual na fase de testes caso o programa não funcione corretamente.**
- **Trabalhos onde o plágio (cópia de colegas ou Internet) for identificado serão anulados.**