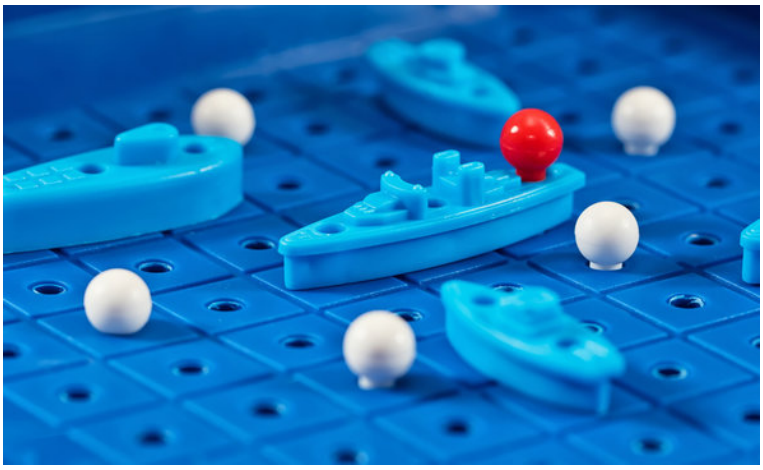
NOVA SCHOOL OF
SCIENCE & TECHNOLOGY

Departamento de Informática

Introdução à Programação
Batalha Naval

Enunciado do Primeiro Trabalho



Ano letivo 2023/24

Versão 1 (2023/10/12)

Preâmbulo

O trabalho é **individual** e é entregue no Mooshak, que aceita submissões até às **20h00** do dia **27 de outubro de 2023**. Leia este enunciado com a máxima atenção, para perceber muito bem o problema e todos os detalhes sobre o concurso do Mooshak e os critérios de avaliação do trabalho.

1 Conceitos e Objetivo do Trabalho

Quem é que nunca jogou à Batalha Naval? Cada um dos dois jogadores começa por dispor os seus barcos numa grelha, às escondidas do adversário. Quando é a sua vez de jogar, dá tiros para atingir os barcos do adversário. O objetivo é afundar todos os barcos do adversário antes dele afundar os seus.

Nesta versão do jogo, algumas regras são diferentes das habituais. A batalha desenrola-se em *grelhas* lineares, que se encontram divididas em *células*, como se ilustra na Figura 1. As células são identificadas pelo seu número de ordem, que é atribuído da esquerda para a direita (a primeira célula ou a célula na primeira posição, a segunda célula ou a célula na segunda posição, etc.), O *tamanho* de uma grelha é o número de células da grelha.

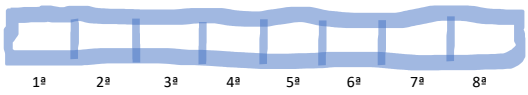


Figura 1: Grelha com 8 células

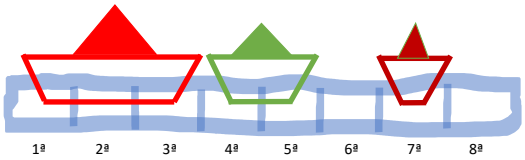


Figura 2: Disposição da frota

Antes do jogo começar, os jogadores combinam o tamanho das grelhas que vão usar e a constituição das frotas (quais os tipos de barcos e quantos barcos há de cada tipo). Depois, cada jogador dispõe os seus barcos na sua grelha, sem sobreposições. Note que os barcos podem estar lado a lado, como na disposição esquematizada na Figura 2. Nesse exemplo, a frota tem três barcos: um barco de três canos nas primeira, segunda e terceira células; um barco de dois canos nas quarta e quinta células; um barco de um cano na sétima célula. O *número de canos* de um barco é o número de células que ocupa.

Os jogadores jogam alternadamente. Cada jogada consiste em dar um tiro numa posição da grelha do adversário. Se o tiro atinge um barco, esse barco *afunda-se* (deixa de existir), independentemente do número de canos que tem. Se o tiro acerta numa posição onde já existiu um barco, o jogador sofre uma penalização. A *pontuação* de um jogador é determinada pelos barcos que afundou e pelos tiros que deu em células com "barcos afundados". O jogo termina assim que a frota de um jogador seja toda afundada.

A disposição da frota na grelha vai ser representada por uma sequência de caracteres cujo comprimento é o tamanho da grelha. Quando uma célula não tem barco, o carácter da sequência é '.' (ponto). As células onde existe um barco têm uma mesma letra nas posições correspondentes da sequência e, se houver barcos lado a lado, as letras que os representam são diferentes. Repare que qualquer disposição pode ser representada por muitas sequências diferentes. Por exemplo, "`RRRGG.R.`" e "`AAABB.C.`" são duas representações da disposição da Figura 2. Se, num determinado momento do jogo, houver barcos afundados, a representação do *estado da frota* tem o carácter '*' nas posições onde esses barcos se encontravam. Vejamos três exemplos, assumindo que a disposição da frota tinha sido descrita por "`RRRGG.R.`".

- Se o único barco afundado fosse o de dois canos, o estado da frota seria (descrito por) "`RRR**.R.`".
- Se o único barco não afundado fosse o de um cano, o estado da frota seria "`*****.R.`".
- Se a frota já tivesse sido toda afundada, o seu estado seria "`*****.*.`".

O objetivo deste trabalho é programar em Java uma versão (bastante alterada) do jogo Batalha Naval. Com a informação sobre os nomes dos jogadores, a disposição das suas frotas e a sequência dos tiros dados, o programa deve ser capaz de indicar, em qualquer momento do jogo, qual é a pontuação e o estado da frota de cada jogador e, caso o jogo ainda não tenha terminado, quem é o próximo jogador a dar um tiro.

2 Regras do Jogo

Antes do jogo começar, os dois jogadores combinam o tamanho das grelhas a usar, a constituição das frotas e quem é o primeiro a jogar. Depois, cada um dispõe os seus barcos na sua grelha (sem sobreposições). Joga um jogador de cada vez, alternadamente.

No início do jogo, os jogadores têm zero pontos. Enquanto o jogo não tiver terminado, o jogador que tem o direito a jogar dá um tiro numa posição da grelha do adversário, ocorrendo uma das três seguintes situações:

- Se existe um barco nessa posição, esse barco vai ao fundo (deixando de existir). O jogador ganha tantos pontos quanto o número de canos do barco a multiplicar por 100.
- Se nunca existiu um barco nessa posição, a pontuação do jogador mantém-se.
- Se não existe um barco nessa posição porque o barco que existia foi afundado, o jogador perde tantos pontos quanto o número de canos do barco que existia a multiplicar por 30.

O jogo termina quando a frota de um dos jogadores estiver toda afundada. A partir desse momento, nenhum jogador pode dar tiros; consequentemente, a pontuação e o estado da frota dos jogadores já não podem mudar. O *vencedor* é o jogador que tiver a maior pontuação quando o jogo termina; se ambos tiverem o mesmo número de pontos, ganha quem tiver afundado toda a frota do adversário. Note que a pontuação pode ser positiva, zero ou negativa.

3 Especificação do Sistema

Pretende-se que a interface da aplicação seja simples, para poder ser utilizada em ambientes diversos e permitir automatizar o processo de teste. Por estes motivos, a entrada e a saída têm de respeitar o formato preciso especificado nesta secção. Pode assumir que o input obedece às restrições de valor e de formato indicadas, ou seja, que o utilizador não comete erros, para além dos previstos neste enunciado.

O programa lê linhas da entrada padrão (`System.in`), escreve linhas na saída padrão (`System.out`) e distingue maiúsculas de minúsculas (por exemplo, as palavras "quit" e "Quit" são diferentes).

Forma do Input

A entrada tem a seguinte estrutura (onde o símbolo ↩ representa uma mudança de linha):

nomeJogador$_1$↩
disposiçãoFrota$_1$↩
nomeJogador$_2$↩
disposiçãoFrota$_2$↩
comando↩
comando↩
.....
comando↩
quit↩

onde:

- *nomeJogador*$_1$ e *nomeJogador*$_2$ são duas sequências diferentes de caracteres, de comprimentos entre 1 e 40 (possivelmente constituídas por várias palavras, como "Fulano Jr.");
- *disposiçãoFrota*$_1$ e *disposiçãoFrota*$_2$ são duas sequências de caracteres com o mesmo comprimento (que é um número entre 2 e 100), onde cada elemento é uma letra maiúscula (de 'A' a 'Z', sem acentos nem cedilhas) ou o carácter '.' (ponto), que têm pelo menos uma letra;
- *comando* é um de quatro comandos (chamados *comando-jogador*, *comando-pontuação*, *comando-frota* e *comando-tiro*) ou um comando inválido, explicados a seguir.

A primeira linha da entrada tem o nome do primeiro jogador, que é quem começa a jogar, e a segunda linha descreve a disposição da sua frota. A terceira e a quarta linhas especificam, respetivamente, o nome e a disposição da frota do outro jogador. Segue-se um número arbitrário de comandos. A última linha tem um comando especial, o *comando-sair*, que só pode ocorrer na última linha porque faz terminar a execução do programa.

Comando-Jogador

O comando-jogador indica que se pretende saber quem é o próximo jogador a dar um tiro, no momento corrente do jogo. Este comando não altera o estado do jogo. As linhas com comandos-jogador têm:

`player`↩

O programa escreve uma linha na consola, distinguindo dois casos:

- Se o jogo já tiver terminado, a linha tem:

  `The game is over`↩

- Nos restantes casos, a linha tem a seguinte forma, onde *nomeJogador* representa o nome do próximo jogador a dar um tiro:

  `Next player:` *nomeJogador*↩

Comando-Pontuação

O comando-pontuação indica que se pretende saber a pontuação do jogador com o nome dado, no momento corrente do jogo. Este comando não altera o estado do jogo. As linhas com comandos-pontuação têm a seguinte forma (com um espaço a separar as duas componentes):

`score` *nomeJogador*↩

onde *nomeJogador* é uma sequência não vazia de caracteres.

O programa escreve uma linha na consola, distinguindo dois casos:

- Se *nomeJogador* não for o nome de um dos jogadores, a linha tem:

  `Nonexistent player`↩

- Nos restantes casos, a linha tem a seguinte forma, onde *pontuação* representa a pontuação do jogador referido no comando:

  nomeJogador `has` *pontuação* `points`↩

Comando-Frota

O comando-frota indica que se pretende visualizar o estado da frota do jogador com o nome dado, no momento corrente do jogo. Este comando não altera o estado do jogo. As linhas com comandos-frota têm a seguinte forma (com um espaço a separar as duas componentes):

`fleet` *nomeJogador*↩

onde *nomeJogador* é uma sequência não vazia de caracteres.

O programa escreve uma linha na consola, distinguindo dois casos:

- Se *nomeJogador* não for o nome de um dos jogadores, a linha tem:

  `Nonexistent player`↩

- Nos restantes casos, a linha tem a seguinte forma, onde *estadoFrota* representa o estado da frota do jogador referido no comando:

  estadoFrota↩

  O estado da frota corresponde à disposição da frota do jogador onde todas as letras que representam barcos afundados por tiros do adversário foram substituídas pelo carácter '*' (asterisco).

Comando-Tiro

O comando-tiro indica que, no momento corrente do jogo, o jogador que tem o direito a jogar deu um tiro na grelha do adversário e qual a célula que escolheu. As linhas com comandos-tiro têm a seguinte forma (com um espaço a separar as duas componentes):

`shoot` *posição*↩

onde *posição* é um número inteiro.

O programa deve usar esta informação para atualizar o estado do jogo, mas não deve escrever qualquer resultado, exceto nos dois casos seguintes, nos quais o estado do jogo não muda e o programa escreve uma linha:

- Se o jogo já tiver terminado, a linha tem:

  `The game is over`↩

- Se o jogo ainda não tiver terminado, mas *posição* não for um número entre 1 e o tamanho da grelha, a linha tem:

  `Invalid shot`↩

Comando-Sair

O comando-sair indica que se pretende terminar a execução do programa. A linha com o comando-sair tem:

`quit`↩

O programa termina, escrevendo uma linha na consola. Distinguem-se dois casos:

- Se o jogo ainda não tiver terminado, a linha tem:

  `The game was not over yet...`↩

- Se o jogo já tiver terminado, a linha tem a seguinte forma, onde *`nomeVencedor`* denota o nome do jogador que ganhou o jogo:

  `nomeVencedor` `won the game!`↩

Comandos Inválidos

Sempre que o utilizador escrever uma linha que não comece com as palavras "player", "score", "fleet", "shoot" ou "quit", o estado do jogo não deve ser alterado e o programa deve escrever uma linha com:

`Invalid command`↩

4 Exemplos

Apresentam-se alguns exemplos. A coluna da esquerda ilustra a interação: o input está escrito a azul e o output a preto. Todas as linhas do input e do output terminam com o símbolo de mudança de linha, que se omitiu para aumentar a legibilidade. A coluna da direita tem informação para o leitor do enunciado, servindo apenas para relembrar as regras descritas anteriormente. Nessa coluna, "**c**" abrevia "cano(s)" e "pts" abrevia "pontos".

Exemplo 1

`John`	Nome do jogador que começa a jogar.
`RRRGG.R.`	Disposição da frota de John.
`Doe`	Nome do outro jogador.
`DD.U.TTT`	Disposição da frota de Doe.
`player`	
`Next player: John`	
`shoot 8`	John atinge barco de 3 **c**; ganha 300 pts.
`fleet Doe`	
`DD.U.***`	
`score John`	
`John has 300 points`	
`shoot 6`	Doe acerta na água.
`shoot 6`	John acerta em barco afundado (de 3 **c**); perde 90 pts.
`shoot 8`	Doe acerta na água.
`fleet John`	
`RRRGG.R.`	
`shoot 3`	John acerta na água.
`shoot 7`	Doe atinge barco de 1 **c**; ganha 100 pts.
`shoot 1`	John atinge barco de 2 **c**; ganha 200 pts.
`score John`	
`John has 410 points`	
`shoot 5`	Doe atinge barco de 2 **c**; ganha 200 pts.
`quit`	
`The game was not over yet...`	

Exemplo 2

`007 The Spy`	Nome do jogador que começa a jogar.
`...X...XX...`	Disposição da frota de 007 The Spy.
`Alice`	Nome do outro jogador.
`I.........KK`	Disposição da frota de Alice.
`Score 007`	O comando é "score" (todas as letras são minúsculas).
`Invalid command`	
`score 007`	"007" não é o nome de nenhum jogador.
`Nonexistent player`	
`fleet Alice Zoe`	"Alice Zoe" não é o nome de nenhum jogador.
`Nonexistent player`	
`shoot 0`	
`Invalid shot`	
`shoot 12`	007 The Spy atinge barco de 2 **c**; ganha 200 pts.
`shoot 1`	Alice acerta na água.
`shoot 11`	007 The Spy acerta em barco afundado (de 2 **c**); perde 60 pts.
`shoot 2`	Alice acerta na água.
`shoot 12`	007 The Spy acerta em barco afundado (de 2 **c**); perde 60 pts.
`shoot 3`	Alice acerta na água.
`shoot 11`	007 The Spy acerta em barco afundado (de 2 **c**); perde 60 pts.
`shoot 13`	
`Invalid shot`	
`player`	
`Next player: Alice`	
`shoot 8`	Alice atinge barco de 2 **c**; ganha 200 pts.
`score 007 The Spy`	
`007 The Spy has 20 points`	
`score Alice`	
`Alice has 200 points`	
`fleet 007 The Spy`	
`...X...**...`	
`fleet Alice`	
`I.........**`	
`shoot 1`	007 The Spy atinge barco de 1 **c**; ganha 100 pts; o jogo termina.
`shoot -5`	
`The game is over`	
`shoot 6`	
`The game is over`	
`player`	
`The game is over`	
`fleet Alice`	
`*.........**`	
`fleet 007 The Spy`	
`...X...**...`	
`score 007 The Spy`	
`007 The Spy has 120 points`	
`score Alice`	
`Alice has 200 points`	
`quit`	Alice venceu porque tem mais pontos que 007 The Spy.
`Alice won the game!`	

5 Entrega do Trabalho

O trabalho é entregue no Mooshak. Deverá submeter um arquivo `.zip` ao Problema A do concurso **IP2324-P1**. Não se esqueça que:

- O arquivo deve conter apenas todos os ficheiros `.java` que tiver criado para resolver o problema.
- O arquivo tem necessariamente de conter um ficheiro `Main.java`, onde está o método `main`.
- A classe `Main` tem de estar na raiz do arquivo (tem de pertencer ao pacote principal (`default`)).
- A versão de Java instalada no Mooshak é a 17.[1]

Cada aluno receberá as credenciais de acesso ao concurso IP2324-P1 (que deverão ser diferentes das credenciais de acesso ao concurso IP2324-Aulas), no seu endereço de email institucional.

O concurso IP2324-P1 abre no dia 16 de outubro e encerra às **20h00** do dia **27 de outubro de 2023** (sexta-feira). Pode ressubmeter o trabalho as vezes que entender, até à hora limite de submissão. Apenas será avaliado o programa que obtiver a **maior pontuação** no Mooshak; se houver vários programas com a maior pontuação, será avaliado, de entre esses, o **último** que tiver sido submetido. Se quiser que o programa avaliado seja outro, tem de enviar uma mensagem à Professora Margarida Mamede (mm@fct.unl.pt) até uma hora após o concurso fechar, indicando o número da submissão que pretende que seja avaliada.

6 Critérios de Avaliação do Trabalho

De acordo com o Regulamento de Avaliação de Conhecimentos da FCT NOVA:

- Existe fraude quando:
  (a) Se utiliza ou tenta utilizar, sob qualquer forma, num teste, exame, ou outra forma de avaliação, presencial ou a distância, informação ou equipamento não autorizado;
  (b) Se presta ou recebe colaboração não autorizada na realização dos exames, testes, ou qualquer outra prova de avaliação de conhecimentos individuais;
  (c) Se presta ou recebe colaboração, não permitida pelas regras aplicáveis a cada caso, na realização de trabalhos práticos, relatórios ou outros elementos de avaliação.
- Os estudantes diretamente envolvidos numa fraude são liminarmente reprovados na disciplina, sem prejuízo de eventual procedimento disciplinar ou cível.

Em IP, o documento Colaboração Permitida e Não Permitida, disponibilizado no Moodle, clarifica as alíneas transcritas acima. Os alunos que cometerem fraude num trabalho não obterão frequência.

A avaliação do trabalho tem duas componentes independentes, cujas notas se somam para obter a nota do trabalho:

- **Funcionalidade** (correção dos resultados produzidos)**: 12 valores**

  Um programa submetido ao concurso que só use as classes da biblioteca permitidas e que obtenha P pontos no Mooshak terá $P/10$ valores.[2] Se o programa usar classes da biblioteca não permitidas, a nota da funcionalidade será inferior $P/10$ valores.

[1] Esta informação é irrelevante se só usar as instruções de Java dadas nas aulas porque, nesse caso, o programa deverá ter o mesmo comportamento na sua máquina e no Mooshak.

[2] O trabalho pode ser realizado incrementalmente. Por exemplo, pode começar por assumir que só há barcos de um cano e que nenhum jogador dá tiros em barcos afundados. É claro que, enquanto o programa não produzir os resultados corretos em todos os testes do Mooshak, não obterá 120 pontos.

As classes da biblioteca permitidas são **apenas as que foram usadas nas aulas teórico-práticas** que decorreram até ao dia 11 de outubro de 2023.

Serão fornecidos testes de treino, semelhantes aos utilizados pelo Mooshak. Se o programa produzir os resultados corretos com todos os testes de treino, é provável (mas não é garantido) que obtenha 120 pontos no Mooshak.

- **Qualidade do código: 8 valores**[3]

  Um código com qualidade tem, entre outras, as seguintes características:

  - **Várias classes que caracterizem bem as diferentes entidades do problema**;
  - Classes, métodos, variáveis e constantes com objetivos bem definidos e as restrições de acesso apropriadas;
  - Algoritmos simples e bem estruturados, implementados com as instruções mais adequadas;
  - Identificadores que expressem os conceitos que representam, escritos de acordo com as convenções ensinadas (por exemplo, o nome de uma classe deve ser um substantivo que começa com uma letra maiúscula);
  - Precondições nos construtores e nos métodos públicos (à exceção do método main);
  - Indentação correta,[4] linhas com 100 caracteres (no máximo)[5] e métodos com 25 linhas (no máximo);
  - Um comentário antes de cada método (à exceção do método main) que indique resumidamente o que o método faz.

Bom trabalho!

[3] Note que a qualidade do código tem um peso muito grande na nota do trabalho.
[4] No Eclipse, pode usar o comando Ctrl I (Windows) ou command I (Mac).
[5] Na contagem do número de caracteres de uma linha de código, considere que um tab equivale a 4 caracteres.