A Game of Flags

Introdução à Programação
Enunciado do 1º Trabalho Prático, versão 1.1 – 2017-11-01
Texto adicionado é apresentado a vermelho

Notas importantes

Prazo de entrega até às 17h00 do dia 10 de Novembro de 2017.

Material a entregar neste trabalho INDIVIDUAL

- **Mooshak**: submissão e aceitação do código fonte pelo sistema de avaliação automática Mooshak. Consulte na página da disciplina as regras de submissão de trabalhos ao Mooshak. O código deve estar devidamente comentado.

Recomendações A boa documentação do seu código fonte será valorizada, tal como a utilização do melhor estilo de programação possível, para além, claro, do correcto funcionamento do projecto. Não se esqueça de comentar as declarações das classes e dos seus métodos com uma explicação do seu significado.

1 Introdução

Este documento descreve o 1º trabalho prático da disciplina de Introdução à Programação do 1º ano do Mestrado Integrado em Engenharia Informática. Descreve ainda uma metodologia de trabalho que os alunos devem seguir para desenvolver o referido trabalho.

O trabalho deve ser realizado **individualmente**.

Esta é uma oportunidade para desenvolver um projecto de maior dimensão, aprendendo com a prática. A experiência é fundamental, pois a programação é uma actividade que só se treina verdadeiramente com a prática continuada. Os alunos podem e devem tirar todas as dúvidas com os docentes da disciplina, bem como conversar com os colegas e discutir soluções, sempre respeitando os princípios do Código de Ética (disponível na página da cadeira no Moodle).

2 Desenvolvimento da Aplicação *A Game of Flags*

2.1 Descrição do Problema

O objectivo deste trabalho visa o desenvolvimento de um jogo de *Captura de Bandeira*. Este jogo é disputado por duas equipas. O objectivo de cada *equipa* é capturar a *bandeira* da equipa adversária, enquanto evita que a equipa adversária capture a sua bandeira. O jogo termina quando uma das bandeiras é capturada, ou quando todos os soldados de uma equipa forem "mortos". Cada equipa é composta por três elementos: um *cavaleiro*, um *lanceiro* e um *espadachim*.

O *mapa de jogo* pode ser visto como um plano em que todas as coordenadas são dadas por valores naturais positivos. Para iniciar um novo jogo, começa-se por estabelecer as dimensões do mapa. Por exemplo, podem-se estabelecer uma dimensão *horizontal* de *10* e uma dimensão

vertical de *20*, dando origem a um mapa com *10* x *20* posições (ou seja, em que as coordenadas horizontais válidas variam entre 1 e 10 e as verticais entre 1 e 20) - o canto inferior esquerdo do mapa tem sempre as coordenadas (1, 1). Seguidamente, cada jogador coloca a sua *bandeira* e os três elementos da equipa nas suas posições iniciais. Este posicionamento inicial obedece às seguintes regras:

- A bandeira tem de ocupar uma posição válida no mapa que não esteja na fronteira. No exemplo dado, as posições válidas são valores naturais positivos no intervalo [2, 9], na coordenada horizontal e no intervalo [2, 19], na coordenada vertical. Repare que usar as coordenadas horizontais 1, ou 10, colocaria a bandeira na fronteira do mapa, podendo o mesmo raciocínio ser aplicado às coordenadas verticais.
- Os elementos da equipa são colocados em posições adjacentes à respectiva bandeira em posições pré-determinadas:
  - O cavaleiro começa sempre na posição imediatamente a norte da bandeira.
  - O espadachim começa sempre na posição imediatamente a leste da bandeira.
  - O lanceiro começa sempre na posição imediatamente a sul da bandeira.
- As regras de posicionamento para as duas equipas são semelhantes, mas existe uma regra adicional no posicionamento da segunda equipa: a bandeira da segunda equipa tem de estar a uma distância superior a 5 posições da bandeira da primeira equipa. Esta distância obtém-se somando a distância vertical com a distância horizontal entre duas posições. Por exemplo, se a bandeira da primeira equipa estiver na posição (2, 3) e a bandeira da segunda equipa estiver na posição (8, 1), a distância entre as bandeiras é a soma da distância vertical ($|2 - 8|$, ou seja, 6) com a distância horizontal ($|3 - 1|$, ou seja, 2). A distância total, neste caso, seria 6+2, ou seja, 8.

As equipas jogam alternadamente. A primeira jogada é da primeira equipa. Em cada jogada, uma equipa pode deslocar apenas um dos seus elementos que ainda esteja "vivo". Sempre que o deslocamento desse elemento o levar a uma posição ocupada por um adversário, dá-se um combate "mortal". O elemento vencedor continua "vivo". O derrotado "morre" e deixa de estar disponível para ser jogado. As regras do combate são as seguintes:

- Os cavaleiros vencem sempre os espadachins.
- Os espadachins vencem sempre os lanceiros.
- Os lanceiros vencem sempre os cavaleiros.
- Quando dois elementos do mesmo tipo (e.g. dois cavaleiros) combatem, vence sempre o atacante, ou seja, o que se moveu.

Para movimentar um elemento, um jogador pode deslocar esse elemento para `norte`, `sul`, `este` ou `oeste`, uma posição de cada vez, não podendo ir ocupar a posição de um elemento "vivo" da mesma equipa, ou a posição da sua bandeira. Os cavaleiros fazem 3 deslocamentos por jogada, enquanto que os espadachins e os lanceiros apenas fazem um deslocamento. Se durante um movimento um elemento da equipa se deslocar para a posição da bandeira da equipa adversária, o jogo termina com a vitória da equipa que alcançou a bandeira adversária. Se após um deslocamento (mesmo que seja um deslocamento intermédio, no caso do cavaleiro) o elemento encontrar um adversário "vivo", dá-se um combate em que um dos adversários "morre". Se após esse combate uma das equipas ficar sem elementos "vivos", o jogo termina

com a vitória da equipa sobrevivente. No caso particular de um cavaleiro, caso sobreviva ao movimento e ainda tenha movimentos disponíveis, o cavaleiro continua o seu movimento.

No início do programa, este mostra os comandos disponíveis (tal como fará sempre que o utilizador escolher a opção `ajuda`). Quando se inicia a aplicação, o jogo não está activo. O utilizador tem de pedir para iniciar um novo jogo, através do comando `novo`. As opções `ajuda` e `sai` também devem estar disponíveis. Os restantes comandos apenas estão disponíveis durante o jogo.

Quando o utilizador pede um `novo` jogo começa um novo jogo. O pedido é acompanhado das dimensões do mapa de jogo e das características das equipas em disputa (nome e localização das respectivas bandeiras).

As duas equipas jogam alternadamente, começando sempre a primeira equipa. Em cada jogada, o jogador escolhe um dos seus elementos (`cavaleiro`, `lanceiro` ou `espadachim`) e indica se ele deve ir para `norte`, `sul`, `este` ou `oeste`. Convenciona-se que ir para `norte` corresponde a incrementar a coordenada vertical da posição, para `sul` corresponde a decrementar essa coordenada, ir para `este` incrementa a coordenada horizontal, e ir para `oeste` decrementa a coordenada horizontal. Se o movimento for inválido, o que pode acontecer se o elemento estiver "morto" ou, estando "vivo" se tentar deslocar para fora do mapa, ou para uma casa ocupada pela bandeira da própria equipa, ou por outro elemento "vivo" da mesma equipa, o jogador perde esse movimento, sendo apresentada uma mensagem adequada. Note que um cavaleiro tem 3 movimentos. Mesmo que falhe algum movimento, desde que continue vivo, pode ainda usar os movimentos que lhe restem. Ou seja, se falhar, por exemplo, o primeiro movimento, continua a usar os dois seguintes. Os restantes elementos têm apenas um movimento por jogada. No final de cada movimento, pode ocorrer um combate. Nesse caso, apenas um dos combatentes sobrevive. Terminada a jogada, passa-se a vez ao adversário. Finalmente, se na sequência de uma jogada houver uma captura de bandeira, o jogo termina. Os jogadores podem sempre começar um novo jogo.

A interação do utilizador com o programa é feita através de comandos, descritos na secção seguinte.

3 Comandos

Nesta secção apresentam-se os vários comandos que o sistema deve ser capaz de interpretar e executar. Nos exemplos apresentados, diferenciamos o `texto escrito pelo utilizador` da retroacção escrita pelo programa na consola, ao executar o comando. Pode assumir que o utilizador não cometerá erros na introdução de argumentos nos comandos, para além dos descritos neste enunciado, ou seja, apenas tem de tratar as situações de erro descritas aqui, pela ordem que são descritas.

Na leitura de comandos o interpretador não deve fazer distinção entre maiúsculas e minúsculas. Por exemplo, para o comando `sai`, o interpretador deve aceitar como válidas as alternativas `SAI`, `sai`, `Sai`, `sAI`, etc. Nos vários exemplos que se seguem, o símbolo ↵ denota a mudança de linha.

Caso o utilizador introduza um comando inexistente, o programa escreve na consola a mensagem Opcao inexitente.. Por exemplo o comando inexistente `desconhecido` teria este efeito:

```
> desconhecido↵
Opcao inexistente.↵
```

Vários comandos têm argumentos. Pode assumir que o utilizador apenas usa argumentos correctos. No entanto, esses e pode acontecer que os argumentos passados a um desses co-

mandos tenham algum valor incorrecto. Por esse motivo, teremos de testar esses argumentos exactamente pela ordem especificada neste enunciado.

3.1 Comando `sai`

Termina a execução do programa. O comando não necessita de argumentos.

```
> sai↵
Obrigado por jogar. Ate a proxima.↵
↵
```

Este comando tem sempre sucesso. Quando é invocado, o programa termina a sua execução, mesmo que isso interrompa um jogo.

3.2 Comando `ajuda`

Informa sobre os comandos disponíveis no programa. O comando não necessita de argumentos. O resultado do comando depende do contexto de utilização, sendo de considerar duas situações alternativas: quando o comando é invocado e não existe um jogo a decorrer, e quando o comando é invocado durante um jogo. Este comando tem sempre sucesso. No cenário seguinte, o comando é invocado num momento em que não está a decorrer nenhum jogo:

```
> ajuda↵
novo - Novo jogo↵
ajuda - Mostra a ajuda↵
sai - Termina a execucao do programa↵
```

No cenário seguinte, está a decorrer um jogo. Note que a *prompt* (equipa>) é diferente, quando há um jogo activo, indicando o nome da equipa que está a jogar.

```
equipa> ajuda↵
novo - Novo jogo↵
mapa - Mostra o mapa do jogo↵
cavaleiro - Move o cavaleiro↵
espadachim - Move o espadachim↵
lanceiro - Move o lanceiro↵
ajuda - Mostra a ajuda↵
sai - Termina a execucao do programa↵
```

3.3 Comando `novo`

Inicia um novo jogo. Pode-se iniciar um novo jogo em qualquer momento. A simples tentativa de iniciar um jogo, mesmo que falhe, implica que se estiver a decorrer outro jogo, ele é interrompido para tentar começar um novo jogo.

Quando se inicia um novo jogo, definem-se as dimensões limite do mapa do jogo, que têm de ser inteiros não inferiores a 10 (ou seja, o mapa mais pequeno é de 10x10 posições, com canto inferior esquerdo na posição (1, 1)). A seguir são lidos os dados iniciais de cada uma das equipas, cada qual numa linha composta pelo nome da equipa, seguido das coordenadas horizontal e vertical da bandeira. O nome da equipa é constituído por uma palavra apenas, sem espaços, e deve ser lido preservando as maiúsculas e minúsculas tal e qual como usadas pelo utilizador (e.g. "NightWatch"). Todas as coordenadas devem ser expressas por valores inteiros. O formato geral do comando é o seguinte (parâmetros com estilo **`bold`**):

```
> novo largura altura↵
equipa1 xBandeira yBandeira↵
equipa2 xBandeira yBandeira↵
```

Por exemplo, suponha que pretende iniciar um jogo com um mapa de 20x20, com equipas denominadas de "Lannisters" e "TheUnsullied", com os "Lannisters" com a bandeira na posição (2, 2) e os "TheUnsullied" com a bandeira na posição (15, 16). Neste caso, tudo está correcto, sendo devolvida a mensagem (Jogo iniciado, comeca a equipa Lannisters.).

```
> novo 20 20↵
Lannisters 2 2↵
TheUnsullied 15 16↵
Jogo iniciado, comeca a equipa Lannisters.↵
```

Caso algo esteja mal o jogo não é criado, mas se existia algum jogo a decorrer, ele é cancelado. Mostra-se uma mensagem de erro adequada:

- (1) Caso alguma das dimensões horizontal ou vertical seja inferior ao limiar mínimo (10), a mensagem de erro será (Mapa pequeno demais para o jogo.).
- (2) Se a bandeira for colocada fora do mapa, ou em alguma posição inválida dentro do mapa (e.g. uma posição na fronteira do mapa, ou uma posição demasiado próxima do adversário, a mensagem de erro apropriada será (**equipa** bandeira em posicao invalida **x** **y**), onde **equipa** representa o nome da equipa cuja bandeira foi mal colocada e **x** e **y** são valores inteiros com as coordenadas inválidas da bandeira. Note que se a primeira bandeira tiver coordenadas que impossibilitem a sua colocação, a segunda bandeira não terá problemas de proximidade com a primeira, porque a primeira não chegou a ser colocada. De igual modo, se uma das equipas não conseguiu colocar a sua bandeira, na prática, não chega a ser criada.
- (3) Se ambas as equipas tiverem o mesmo nome, a mensagem de erro deve ser (As equipas nao podem ter o mesmo nome.).

Os erros (2) e (3) são reportados pela ordem acima descrita, ou seja, a não colocação de uma ou das duas bandeiras implica que nem vale a pena testar se os nomes das equipas são iguais. Seguem-se alguns exemplos ilustrativos. Note que em todos estes casos o *feedback* é dado apenas no final de o comando completo (incluindo a parte das bandeiras) ser lido.

Mapa pequeno demais para o jogo:

```
> novo 5 6↵
Lannisters 2 2↵
Starks 15 16↵
Mapa pequeno demais para o jogo.↵
```

Bandeira colocada fora do mapa:

```
> novo 20 20↵
Lannisters 2 2↵
Starks 25 16↵
Starks bandeira em posicao invalida 25 16.↵
```

Bandeira colocada na fronteira:

```
> novo 20 20↵
Lannisters 1 1↵
Starks 15 16↵
Lannisters bandeira em posicao invalida 1 1.↵
```

Bandeiras demasiado próximas. Neste caso, a segunda bandeira a ser colocada é a que é considerada em posição inválida:

```
> novo 20 20↵
Lannisters 12 16↵
Starks 15 16↵
Starks bandeira em posicao invalida 15 16.↵
```

Nomes duplicados:

```
> novo 20 20↵
Lannisters 2 2↵
Lannisters 15 16↵
As equipas nao podem ter o mesmo nome.↵
```

Erros acumulados (ambas as bandeiras na fronteira):

```
> novo 20 20↵
Lannisters 1 1↵
Lannisters 20 3↵
Lannisters bandeira em posicao invalida 1 1.↵
Lannisters bandeira em posicao invalida 20 3.↵
```

3.4 Comando **mapa**

Informa sobre o estado actual do jogo. Este comando apenas está disponível quando existe um jogo a decorrer. O comando não necessita de argumentos. Descreve o estado actual do jogo, com o seguinte formato:

```
equipa > mapa↵
dimensaoHorizontal dimensaoVertical↵
equipa1 xBandeira yBandeira
cavaleiro(estadoCavaleiro) xCavaleiro yCavaleiro
espadachim(estadoEspadachim) xEspadachim yEspadachim
lanceiro(estadoLanceiro) xLanceiro yLanceiro↵
equipa2 xBandeira yBandeira
cavaleiro(estadoCavaleiro) xCavaleiro yCavaleiro
espadachim(estadoEspadachim) xEspadachim yEspadachim
lanceiro(estadoLanceiro) xLanceiro yLanceiro↵
```

Por exemplo, suponha que em dado momento, os "Lannister" perderam um cavaleiro e os "Stark" um lanceiro e um cavaleiro. É a vez de um jogarem os "Stark" O mapa seria apresentado assim:

Stark > `mapa`↵
20 20↵
Lannister 2 2 cavaleiro(morto) 2 3 espadachim(vivo) 3 4 lanceiro(vivo) 4 4↵
Stark 15 16 cavaleiro(morto) 5 7 espadachim(vivo) 8 8 lanceiro(morto) 12 10↵

Este comando apenas está activo durante o jogo:

- (1) Caso não esteja a decorrer nenhum jogo, o programa deve escrever a mensagem (Comando inactivo.).

> `mapa`↵
Comando inactivo.↵

3.5 Comando `cavaleiro`

Movimenta um cavaleiro. O comando necessita de três argumentos, correspondentes aos três movimentos permitidos a um cavaleiro. O cavaleiro tentará executar, sucessivamente, os três movimentos. Cada movimento consiste em indicar uma direcção para a qual o cavaleiro se deve deslocar. O formato do comando é o seguinte:

equipa > `cavaleiro` **`direccao direccao direccao`**↵

Cada **`direccao`** pode assumir os valores `norte`, `sul`, `este`, ou `oeste`. Por exemplo, o `cavaleiro` pode tentar deslocar-se, sucessivamente, para `norte`, `este` e `este`. Este comando terá, se tudo correr bem, o efeito de deslocar o cavaleiro uma posição para norte e duas para este. O output gerado, no final de cada deslocação parcial, deve reportar a nova posição e estado do cavaleiro, com o formato:

equipa cavaleiro **estado xCavaleiro yCavaleiro**↵

Por exemplo, se o cavaleiro partiu da posição (3, 7):

Lannister > `cavaleiro norte este este`↵
Lannister cavaleiro(vivo) 3 8↵
Lannister cavaleiro(vivo) 4 8↵
Lannister cavaleiro(vivo) 5 8↵

Este comando apenas está activo durante o jogo:

- (1) Caso não esteja a decorrer nenhum jogo, o programa deve escrever a mensagem (Comando inactivo.).

Se o jogo está a correr, o terreno de jogo é um local perigoso e várias coisas podem correr mal:

- (2) O `cavaleiro` pode ter morrido (O cavaleiro da ilustre casa de **equipa** ja nao esta entre nos.).
- (3) O o nosso valoroso `cavaleiro` pode tentar deslocar-se para fora do mapa. Não toleramos desertores. Se o cavaleiro tentar fugir do mapa, fica quietinho e ganha a fama de cobardolas sendo escrita a mensagem (O cavaleiro da ilustre casa de **equipa** e um cobardolas.).

- (4) Pode também acontecer que o `cavaleiro` se tente deslocar para uma posição já ocupada por um elemento da sua equipa que ainda esteja "vivo", ou para a posição da sua bandeira (nada impede que o `cavaleiro` ocupe a mesma posição que um colega de equipa já "morto"). Como a posição está ocupada, o movimento deve falhar, ou seja, o `cavaleiro` não se mexe. Se houver mais ordens, ele poderá cumpri-las, naturalmente. Cada vez que se tentar deslocar para uma posição já ocupada deve ser escrita a mensagem (O cavaleiro da ilustre casa de **equipa** devia tentar ir para outro sitio.). Ao se deslocar, o cavaleiro pode tentar invadir uma posição ocupada por um adversário.

- (5) Se o adversário for outro cavaleiro, ou um espadachim, o cavaleiro atacante vence a batalha, e o adversário "morre". A mensagem a escrever é (Muhahah, sou um **equipa**! Sou invencivel! Nenhum **soldado** me faz frente!), em que **equipa** representa a equipa do cavaleiro e **soldado** o oponente derrotado em batalha (cavaleiro ou espadachim).

- (6) No entanto, se encontrar em combate um lanceiro, o cavaleiro é derrotado e "morre". A mensagem adequada, nesse caso, é (Argh! A dor! Maldito sejas, lanceiro **rival**.) em que **rival** é a equipa oponente.

- (7) Na sequência de um combate, o `cavaleiro` pode derrotar o último adversário, vencendo assim o jogo. Nesse caso, além da mensagem descrita em (5), é escrita a mensagem (Sou um heroi **equipa**! A bandeira **rival** e nossa! Vitoria gloriosa!), onde **equipa** é a equipa do `cavaleiro` e **rival** é a equipa oponente.

- (8) Se o `cavaleiro` for derrotado e for o último da sua equipa, essa derrota resulta na perda do jogo. Nesse caso, além da mensagem descrita em (6), é escrita a mensagem de derrota no jogo, que é semelhante à de vitória, descrita em (7), mas com os papéis trocados (Sou um heroi **rival**! A bandeira **equipa** e nossa! Vitoria gloriosa!). É o rival quem fala por último, neste caso.

- (9) Finalmente, pode acontecer que o `cavaleiro` conquiste a bandeira rival deslocando-se para a posição ocupada por essa bandeira. A mensagem de vitória é semelhante à apresentada em (7): (Sou um heroi **equipa**! A bandeira **rival** e nossa! Vitoria gloriosa!), onde **equipa** é a equipa do `cavaleiro` e **rival** é a equipa oponente.

Neste exemplo, não existe nenhum jogo a decorrer:

```
> cavaleiro norte este este↵
Comando inactivo.↵
```

Neste exemplo, o jogador tenta mover um cavaleiro "morto" na posição (3, 7):

```
Lannister > cavaleiro norte este este↵
O cavaleiro da ilustre casa de Lannister ja nao esta entre nos.↵
Lannister cavaleiro(morto) 3 7↵
O cavaleiro da ilustre casa de Lannister ja nao esta entre nos.↵
Lannister cavaleiro(morto) 3 7↵
O cavaleiro da ilustre casa de Lannister ja nao esta entre nos.↵
Lannister cavaleiro(morto) 3 7↵
```

Neste exemplo num mapa de 20x20, o cavaleiro se encontra na posição (19, 12) e recebe a jogada este este norte. Nesse caso, ainda consegue fazer o primeiro movimento, mas falha o seguinte, conseguindo finalmente fazer o terceiro movimento:

```
Lannister > cavaleiro este este norte↵
Lannister cavaleiro(vivo) 20 12↵
O cavaleiro da ilustre casa de Lannister e um cobardolas.↵
Lannister cavaleiro(vivo) 20 12↵
Lannister cavaleiro(vivo) 20 13↵
```

No exemplo que se segue, o `cavaleiro` que se encontra na posição (13, 10) recebe as ordens `sul este sul`, mas vê o segundo movimento falhado por essa posição estar ocupada por um colega de equipa "vivo".

```
Lannister > cavaleiro sul este sul↵
Lannister cavaleiro(vivo) 13 9↵
O cavaleiro da ilustre casa de Lannister devia tentar ir para outro sitio.↵
Lannister cavaleiro(vivo) 13 9↵
Lannister cavaleiro(vivo) 13 8↵
```

No exemplo que se segue, o `cavaleiro` começa na posição (10, 10) tenta-se deslocar 3 vezes para `sul`, encontrando, sucessivamente, um cavaleiro, um espadachim e um lanceiro adversários. O `cavaleiro` derrota o cavaleiro e o espadachim adversários, sendo finalmente derrotado pelo lanceiro.

```
Lannister > cavaleiro sul sul sul↵
Muhahah, sou um Lannister! Sou invencivel! Nenhum cavaleiro me faz frente!↵
Lannister cavaleiro(vivo) 9 10↵
Muhahah, sou um Lannister! Sou invencivel! Nenhum espadachim me faz frente!↵
Lannister cavaleiro(vivo) 8 10↵
Argh! A dor! Maldito sejas, lanceiro Stark.↵
Lannister cavaleiro(morto) 7 10↵
```

No exemplo que se segue, o `cavaleiro` da equipa `Lannister`, começa na posição (10, 10) tenta-se deslocar 3 vezes para `sul`, encontrando no primeiro movimento um cavaleiro da equipa `Stark`. O `cavaleiro` derrota o cavaleiro da equipa `Stark`, que era o último sobrevivente dessa equipa. Assim, o jogo está ganho pela equipa `Lannister`. Neste cenário, os dois últimos movimentos são ignorados.

```
Lannister > cavaleiro sul sul sul↵
Muhahah, sou um Lannister! Sou invencivel! Nenhum cavaleiro me faz frente!↵
Sou um heroi Lannister! A bandeira Stark e nossa! Vitoria gloriosa!↵
Lannister cavaleiro(vivo) 10 9↵
```

No exemplo que se segue, o último soldado da equipa `Lannister`, o `cavaleiro`, começa na posição (10, 10) tenta-se deslocar 3 vezes para `sul`, encontrando, sucessivamente, um cavaleiro e um lanceiro da equipa `Stark`. O `cavaleiro` derrota o cavaleiro da equipa `Stark`, mas depois é derrotado pelo lanceiro. Por ser o último, perde o jogo, dando a vitória à equipa `Stark`. Neste cenário, a última jogada é ignorada.

```
Lannister > cavaleiro sul sul sul↵
Muhahah, sou um Lannister! Sou invencivel! Nenhum cavaleiro me faz frente!↵
Lannister cavaleiro(vivo) 9 10↵
Argh! A dor! Maldito sejas, lanceiro Stark.↵
Sou um heroi Stark! A bandeira Lannister e nossa! Vitoria gloriosa!↵
Lannister cavaleiro(morto) 7 10↵
```

No exemplo que se segue, o `cavaleiro` começa na posição (13, 16) tenta-se deslocar 3 vezes para `este`, encontrando no segundo movimento a bandeira adversária, na posição (15, 16). Neste cenário, a última jogada é ignorada.

```
Lannister > cavaleiro este este este↵
Lannister cavaleiro(vivo) 14 16↵
Sou um heroi Lannister! A bandeira Stark e nossa! Vitoria gloriosa!↵
Lannister cavaleiro(vivo) 15 16↵
```

3.6 Comando **espadachim**

Movimenta um espadachim. O comando necessita de um argumento, correspondente à direcção para a qual o espadachim se deve deslocar. O formato do comando é o seguinte:

equipa > `espadachim` **`direccao`**↵

Cada **`direccao`** pode assumir os valores `norte`, `sul`, `este`, ou `oeste`. O output gerado, no final de cada deslocação parcial, deve reportar a nova posição e estado do espadachim, com o formato:

equipa espadachim **estado xEspadachim yEspadachim**↵

Por exemplo, se o espadachim partiu da posição (3, 7):

```
Lannister > espadachim norte↵
Lannister espadachim(vivo) 3 8↵
```

Este comando apenas está activo durante o jogo.

- (1) Caso não esteja a decorrer nenhum jogo, o programa deve escrever a mensagem (Comando inactivo.).

Se o jogo está a correr, o terreno de jogo é um local perigoso e várias coisas podem correr mal:

- (2) O `espadachim` pode ter morrido (O espadachim da ilustre casa de **equipa** ja nao esta entre nos.).
- (3) O `espadachim` pode tentar deslocar-se para fora do mapa. Não toleramos desertores. Se o espadachim tentar fugir do mapa, fica quietinho e ganha a fama de cobardolas sendo escrita a mensagem (O espadachim da ilustre casa de **equipa** e um cobardolas.).
- (4) Pode também acontecer que o `espadachim` se tente deslocar para uma posição já ocupada por um elemento da sua equipa que ainda esteja "vivo", ou para a posição da sua bandeira (nada impede que o `espadachim` ocupe a mesma posição que um colega de equipa já "morto"). Como a posição está ocupada, o movimento deve falhar, ou seja, o `espadachim` não se mexe. Se houver mais ordens, ele poderá cumpri-las, naturalmente. Cada vez que se tentar deslocar para uma posição já ocupada deve ser escrita a mensagem (O espadachim da ilustre casa de **equipa** devia tentar ir para outro sitio.). Ao se deslocar, o espadachim pode tentar invadir uma posição ocupada por um adversário.
- (5) Se o adversário for outro espadachim, ou um lanceiro, o espadachim atacante vence a batalha, e o adversário "morre". A mensagem a escrever é (Muhahah, sou um **equipa**!

Sou invencivel! Nenhum **soldado** me faz frente!), em que **equipa** representa a equipa do espadachim e **soldado** o oponente derrotado em batalha (espadachim ou lanceiro).

- (6) No entanto, se encontrar em combate um cavaleiro, o espadachim é derrotado e "morre". A mensagem adequada, nesse caso, é (Argh! A dor! Maldito sejas, cavaleiro **rival**.) em que **rival** é a equipa oponente.
- (7) Na sequência de um combate, o `espadachim` pode derrotar o último adversário, vencendo assim o jogo. Nesse caso, além da mensagem descrita em (5), é escrita a mensagem (Sou um heroi **equipa**! A bandeira **rival** e nossa! Vitoria gloriosa!), onde **equipa** é a equipa do `espadachim` e **rival** é a equipa oponente.
- (8) Se o `espadachim` for derrotado e for o último da sua equipa, essa derrota resulta na perda do jogo. Nesse caso, além da mensagem descrita em (6), é escrita a mensagem de derrota no jogo, que é semelhante à de vitória, descrita em (7), mas com os papéis trocados (Sou um heroi **rival**! A bandeira **equipa** e nossa! Vitoria gloriosa!). É o rival quem fala por último, neste caso.
- (9) Finalmente, pode acontecer que o `espadachim` conquiste a bandeira rival deslocando-se para a posição ocupada por essa bandeira. A mensagem de vitória é semelhante à apresentada em (7): (Sou um heroi **equipa**! A bandeira **rival** e nossa! Vitoria gloriosa!), onde **equipa** é a equipa do `espadachim` e **rival** é a equipa oponente.

Neste exemplo, não existe nenhum jogo a decorrer:

```
> espadachim norte↵
Comando inactivo.↵
```

Neste exemplo, o jogador tenta mover um espadachim "morto" na posição (3, 7):

```
Lannister > espadachim norte↵
O espadachim da ilustre casa de Lannister ja nao esta entre nos.↵
Lannister espadachim(morto) 3 7↵
```

Neste exemplo num mapa de 20x20, o `espadachim` encontra-se na posição (20, 12) e recebe a jogada `este`. Nesse caso, falha o movimento:

```
Lannister > espadachim este↵
O espadachim da ilustre casa de Lannister e um cobardolas.↵
Lannister espadachim(vivo) 20 12↵
```

No exemplo que se segue, o `espadachim` que se encontra na posição (13, 10) recebe a ordem `sul`, mas vê o movimento falhado por essa posição estar ocupada por um colega de equipa "vivo".

```
Lannister > espadachim sul↵
O espadachim da ilustre casa de Lannister devia tentar ir para outro sitio.↵
Lannister espadachim(vivo) 13 10↵
```

No exemplo que se segue, o `espadachim` começa na posição (10, 10) tenta-se deslocar para `sul`, encontrando um lanceiro adversário. O `espadachim` derrota o lanceiro adversário.

Lannister > espadachim sul↵
Muhahah, sou um Lannister! Sou invencivel! Nenhum lanceiro me faz frente!↵
Lannister espadachim(vivo) 10 9↵

No exemplo que se segue, o espadachim começa na posição (10, 10) tenta-se deslocar para sul, encontrando um cavaleiro adversário. O espadachim é derrotado pelo espadachim adversário.

Lannister > espadachim sul↵
Argh! A dor! Maldito sejas, cavaleiro Stark.↵
Lannister espadachim(morto) 10 9↵

No exemplo que se segue o espadachim, começa na posição (10, 10) tenta-se deslocar para sul, encontrando o último soldado adversário, um lanceiro. O espadachim derrota o lanceiro adversário. Assim, o jogo está ganho pela equipa Lannister.

Lannister > espadachim sul↵
Muhahah, sou um Lannister! Sou invencivel! Nenhum lanceiro me faz frente!↵
Sou um heroi Lannister! A bandeira Stark e nossa! Vitoria gloriosa!↵
Lannister espadachim(vivo) 10 9↵

No exemplo que se segue o último soldado da equipa Lannister, um espadachim, começa na posição (10, 10) tenta-se deslocar para sul, encontrando um cavaleiro adversário. O espadachim é derrotado pelo cavaleiro adversário. Por ser o último, perde o jogo, dando a vitória à equipa Stark.

Lannister > espadachim sul↵
Argh! A dor! Maldito sejas, cavaleiro Stark.↵
Sou um heroi Stark! A bandeira Lannister e nossa! Vitoria gloriosa!↵
Lannister espadachim(morto) 10 9↵

No exemplo que se segue, o espadachim começa na posição (13, 16) e desloca-se para este, encontrando a bandeira adversária.

Lannister > espadachim este↵
Sou um heroi Lannister! A bandeira Stark e nossa! Vitoria gloriosa!↵
Lannister espadachim(vivo) 14 16↵

3.7 Comando `lanceiro`

Movimenta um lanceiro. O comando necessita de um argumento, correspondente à direcção para a qual o lanceiro se deve deslocar. O formato do comando é o seguinte:

equipa > lanceiro **direccao**↵

Cada **direccao** pode assumir os valores norte, sul, este, ou oeste. O output gerado, no final de cada deslocação parcial, deve reportar a nova posição e estado do lanceiro, com o formato:

equipa lanceiro **estado xLanceiro yLanceiro**↵

Por exemplo, se o lanceiro partiu da posição (3, 7):

Lannister > `lanceiro norte`↵
Lannister lanceiro(vivo) 3 8↵

Este comando apenas está activo durante o jogo.

- (1) Caso não esteja a decorrer nenhum jogo, o programa deve escrever a mensagem (Comando inactivo.).

Se o jogo está a correr, o terreno de jogo é um local perigoso e várias coisas podem correr mal.

- (2) O `lanceiro` pode ter morrido (O lanceiro da ilustre casa de **equipa** ja nao esta entre nos.).
- (3) O `lanceiro` pode tentar deslocar-se para fora do mapa. Não toleramos desertores. Se o lanceiro tentar fugir do mapa, fica quietinho e ganha a fama de cobardolas sendo escrita a mensagem (O lanceiro da ilustre casa de **equipa** e um cobardolas.).
- (4) Pode também acontecer que o `lanceiro` se tente deslocar para uma posição já ocupada por um elemento da sua equipa que ainda esteja "vivo", ou para a posição da sua bandeira (nada impede que o `lanceiro` ocupe a mesma posição que um colega de equipa já "morto"). Como a posição está ocupada, o movimento deve falhar, ou seja, o `lanceiro` não se mexe. Se houver mais ordens, ele poderá cumpri-las, naturalmente. Cada vez que se tentar deslocar para uma posição já ocupada deve ser escrita a mensagem (O lanceiro da ilustre casa de **equipa** devia tentar ir para outro sitio.). Ao se deslocar, o lanceiro pode tentar invadir uma posição ocupada por um adversário.
- (5) Se o adversário for outro lanceiro, ou um cavaleiro, o lanceiro atacante vence a batalha, e o adversário "morre". A mensagem a escrever é (Muhahah, sou um **equipa**! Sou invencivel! Nenhum **soldado** me faz frente!), em que **equipa** representa a equipa do lanceiro e **soldado** o oponente derrotado em batalha (lanceiro ou cavaleiro).
- (6) No entanto, se encontrar em combate um espadachim, o lanceiro é derrotado e "morre". A mensagem adequada, nesse caso, é (Argh! A dor! Maldito sejas, espadachim **rival**.) em que **rival** é a equipa oponente.
- (7) Na sequência de um combate, o `lanceiro` pode derrotar o último adversário, vencendo assim o jogo. Nesse caso, além da mensagem descrita em (5), é escrita a mensagem (Sou um heroi **equipa**! A bandeira **rival** e nossa! Vitoria gloriosa!), onde **equipa** é a equipa do `lanceiro` e **rival** é a equipa oponente.
- (8) Se o `lanceiro` for derrotado e for o último da sua equipa, essa derrota resulta na perda do jogo. Nesse caso, além da mensagem descrita em (6), é escrita a mensagem de derrota no jogo, que é semelhante à de vitória, descrita em (7), mas com os papéis trocados (Sou um heroi **rival**! A bandeira **equipa** e nossa! Vitoria gloriosa!). É o rival quem fala por último, neste caso.
- (9) Finalmente, pode acontecer que o `lanceiro` conquiste a bandeira rival deslocando-se para a posição ocupada por essa bandeira. A mensagem de vitória é semelhante à apresentada em (7): (Sou um heroi **equipa**! A bandeira **rival** e nossa! Vitoria gloriosa!), onde **equipa** é a equipa do `lanceiro` e **rival** é a equipa oponente.

Neste exemplo, não existe nenhum jogo a decorrer:

```
> lanceiro norte↵
Comando inactivo.↵
```

Neste exemplo, o jogador tenta mover um lanceiro "morto" na posição (3, 7):

```
Lannister > lanceiro norte↵
O lanceiro da ilustre casa de Lannister ja nao esta entre nos.↵
Lannister lanceiro(morto) 3 7↵
```

Neste exemplo num mapa de 20x20, o `lanceiro` encontra-se na posição (20, 12) e recebe a jogada `este`. Nesse caso, falha o movimento:

```
Lannister > lanceiro este↵
O lanceiro da ilustre casa de Lannister e um cobardolas.↵
Lannister lanceiro(vivo) 20 12↵
```

No exemplo que se segue, o `lanceiro` que se encontra na posição (13, 10) recebe a ordem `sul`, mas vê o movimento falhado por essa posição estar ocupada por um colega de equipa "vivo".

```
Lannister > lanceiro sul↵
O lanceiro da ilustre casa de Lannister devia tentar ir para outro sitio.↵
Lannister lanceiro(vivo) 13 10↵
```

No exemplo que se segue, o `lanceiro` começa na posição (10, 10) tenta-se deslocar para `sul`, encontrando um cavaleiro adversário. O `lanceiro` derrota o cavaleiro adversário.

```
Lannister > lanceiro sul↵
Muhahah, sou um Lannister! Sou invencivel! Nenhum cavaleiro me faz frente!↵
Lannister lanceiro(vivo) 10 9↵
```

No exemplo que se segue, o `lanceiro` começa na posição (10, 10) tenta-se deslocar para `sul`, encontrando um espadachim adversário. O `lanceiro` é derrotado pelo espadachim adversário.

```
Lannister > lanceiro sul↵
Argh! A dor! Maldito sejas, espadachim Stark.↵
Lannister lanceiro(morto) 10 9↵
```

No exemplo que se segue o `lanceiro`, começa na posição (10, 10) tenta-se deslocar para `sul`, encontrando o último soldado adversário, um cavaleiro. O `lanceiro` derrota o cavaleiro adversário. Assim, o jogo está ganho pela equipa `Lannister`.

```
Lannister > lanceiro sul↵
Muhahah, sou um Lannister! Sou invencivel! Nenhum cavaleiro me faz frente!↵
Sou um heroi Lannister! A bandeira Stark e nossa! Vitoria gloriosa!↵
Lannister lanceiro(vivo) 10 9↵
```

No exemplo que se segue o último soldado da equipa `Lannister`, um `lanceiro`, começa na posição (10, 10) tenta-se deslocar para `sul`, encontrando um espadachim adversário. O `lanceiro` é derrotado pelo espadachim adversário. Por ser o último, perde o jogo, dando a vitória à equipa `Stark`.

```
Lannister > lanceiro sul↵
Argh! A dor! Maldito sejas, cavaleiro Stark.↵
Sou um heroi Stark! A bandeira Lannister e nossa! Vitoria gloriosa!↵
Lannister lanceiro(morto) 10 9↵
```

No exemplo que se segue, o lanceiro começa na posição (13, 16) e desloca-se para este, encontrando a bandeira adversária.

```
Lannister > lanceiro este↵
Sou um heroi Lannister! A bandeira Stark e nossa! Vitoria gloriosa!↵
Lannister lanceiro(vivo) 14 16↵
```

4 Exemplo completo

Nesta secção apresentamos um exemplo de jogo completo. Note que este exemplo não pretende ser exaustivo de tudo o que pode acontecer, mas apenas ilustrar uma sessão de jogo típica.

```
novo - Novo jogo↵
ajuda - Mostra a ajuda↵
sai - Termina a execucao do programa↵
> ajuda↵
novo - Novo jogo↵
ajuda - Mostra a ajuda↵
sai - Termina a execucao do programa↵
> novo 10 10↵
Baratheon 2 2↵
Targaryen 2 8↵
Jogo iniciado, comeca a equipa Baratheon.↵
Baratheon > mapa↵
10 10↵
Baratheon 2 2 cavaleiro(vivo) 2 3 espadachim(vivo) 3 2 lanceiro(vivo) 2 1↵
Targaryen 2 8 cavaleiro(vivo) 2 9 espadachim(vivo) 3 8 lanceiro(vivo) 2 7↵
Baratheon > cavaleiro este este este↵
Baratheon cavaleiro(vivo) 3 3↵
Baratheon cavaleiro(vivo) 4 3↵
Baratheon cavaleiro(vivo) 5 3↵
Targaryen > ajuda↵
novo - Novo jogo↵
mapa - Mostra o mapa do jogo↵
cavaleiro - Move o cavaleiro↵
espadachim - Move o espadachim↵
lanceiro - Move o lanceiro↵
ajuda - Mostra a ajuda↵
sai - Termina a execucao do programa↵
Targaryen > cavaleiro sul sul sul↵
O cavaleiro da ilustre casa de Targaryen devia tentar ir para outro sitio.↵
Targaryen cavaleiro(vivo) 2 9↵
O cavaleiro da ilustre casa de Targaryen devia tentar ir para outro sitio.↵
Targaryen cavaleiro(vivo) 2 9↵
O cavaleiro da ilustre casa de Targaryen devia tentar ir para outro sitio.↵
Targaryen cavaleiro(vivo) 2 9↵
```

Baratheon > cavaleiro norte norte norte↵
Baratheon cavaleiro(vivo) 5 4↵
Baratheon cavaleiro(vivo) 5 5↵
Baratheon cavaleiro(vivo) 5 6↵
Targaryen > cavaleiro este este sul↵
Targaryen cavaleiro(vivo) 3 9↵
Targaryen cavaleiro(vivo) 4 9↵
Targaryen cavaleiro(vivo) 4 8↵
Baratheon > mapa↵
10 10↵
Baratheon 2 2 cavaleiro(vivo) 5 6 espadachim(vivo) 3 2 lanceiro(vivo) 2 1↵
Targaryen 2 8 cavaleiro(vivo) 4 8 espadachim(vivo) 3 8 lanceiro(vivo) 2 7↵
Baratheon > cavaleiro norte norte oeste↵
Baratheon cavaleiro(vivo) 5 7↵
Baratheon cavaleiro(vivo) 5 8↵
Muhahah, sou um Baratheon! Sou invencivel! Nenhum cavaleiro me faz frente!↵
Baratheon cavaleiro(vivo) 4 8↵
Targaryen > lanceiro sul↵
Targaryen lanceiro(vivo) 2 6↵
Baratheon > mapa↵
10 10↵
Baratheon 2 2 cavaleiro(vivo) 4 8 espadachim(vivo) 3 2 lanceiro(vivo) 2 1↵
Targaryen 2 8 cavaleiro(morto) 4 8 espadachim(vivo) 3 8 lanceiro(vivo) 2 6↵
Baratheon > cavaleiro oeste sul sul↵
Muhahah, sou um Baratheon! Sou invencivel! Nenhum espadachim me faz frente!↵
Baratheon cavaleiro(vivo) 3 8↵
Baratheon cavaleiro(vivo) 3 7↵
Baratheon cavaleiro(vivo) 3 6↵
Targaryen > mapa↵
10 10↵
Baratheon 2 2 cavaleiro(vivo) 3 6 espadachim(vivo) 3 2 lanceiro(vivo) 2 1↵
Targaryen 2 8 cavaleiro(morto) 4 8 espadachim(morto) 3 8 lanceiro(vivo) 2 6↵
Targaryen > lanceiro este↵
Muhahah, sou um Targaryen! Sou invencivel! Nenhum cavaleiro me faz frente!↵
Targaryen lanceiro(vivo) 3 6↵
Baratheon > espadachim norte↵
Baratheon espadachim(vivo) 3 3↵
Targaryen > lanceiro sul↵
Targaryen lanceiro(vivo) 3 5↵
Baratheon > mapa↵
10 10↵
Baratheon 2 2 cavaleiro(morto) 3 6 espadachim(vivo) 3 3 lanceiro(vivo) 2 1↵
Targaryen 2 8 cavaleiro(morto) 4 8 espadachim(morto) 3 8 lanceiro(vivo) 3 5↵
Baratheon > espadachim norte↵
Baratheon espadachim (vivo) 3 4↵
Targaryen > lanceiro sul↵
Argh! A dor! Maldito sejas, espadachim Baratheon.↵
Sou um heroi Baratheon! A bandeira Targaryen e nossa! Vitoria gloriosa!↵
Targaryen lanceiro(morto) 3 4↵
> ajuda↵
novo - Novo jogo↵
ajuda - Mostra a ajuda↵
sai - Termina a execucao do programa↵
> sai↵
Obrigado por jogar. Ate a proxima.↵
↵

5 Método de Desenvolvimento do Projeto

Esta secção fornece algumas recomendações sobre uma metodologia de trabalho adequada para um projecto de desenvolvimento de um sistema de software. É muito importante ser metódico em qualquer actividade com alguma duração, particularmente num projecto deste tipo. Acarreta uma sequência de actividades que devem ser feitas de forma disciplinada (ou seja, um processo de desenvolvimento), para se obter um "produto" final de qualidade, minorando a ocorrência de defeitos (bugs) e facilitando a sua detecção quando ocorrem, facilitando a legibilidade do programa, e... ter uma boa nota.

5.1 ETAPA 1 – Compreensão e Esclarecimento do Enunciado e dos Objectivos do Projecto

Leia bem o enunciado, anotando todas as questões que lhe ocorram - de preferência por escrito e em papel. Mantenha um documento escrito com as dúvidas sobre este trabalho e com as respostas que for obtendo. É essencial saber, em cada momento, que dúvidas tem, e quais as respostas às dúvidas que teve, mas já esclareceu.

Como esclarecer dúvidas? Com os docentes da disciplina. Aproveite os horários de dúvidas e as aulas práticas. Nesta fase eliminará as maiores dúvidas sobre o enunciado e sobre o que se pretende. Claro, algumas dúvidas podem persistir e podem surgir novas questões durante as fases seguintes. Por isso, prepare-se para manter o seu documento de dúvidas até ao fim do prazo de entrega.

5.2 ETAPA 2 – Análise da Estrutura Geral do Programa

Nesta fase, terá como objectivo definir a estrutura global do seu programa. Como bem sabe, um programa em Java consiste num conjunto de classes Java, cujos objectos representam as várias entidades do domínio do problema, e talvez uma ou outra classe do domínio da solução ou implementação.

No caso que temos em mão, além da classe Main, vai existir uma classe do domínio para cada entidade necessária para construir a aplicação. Lembre que aqui falamos de jogos, equipas, bandeiras, soldados, etc.

Os objectos dessas classes representarão os conceitos em actividade durante a execução do programa. Manterão a informação relativa a cada instância desse conceito, e fornecerão grupos de operações apropriadas. Defina nesta fase quais são as várias classes de que necessita e as suas interfaces no contexto do programa em causa. Certifique-se de que as operações que identificou nas interfaces pertencem mesmo à classe em que as colocou. Identifique também as variáveis e constantes que achar necessárias para representar o estado de cada objecto de cada classe, assim como o seu tipo.

Não se esqueça de pensar bem como representar a informação dentro de cada objecto. Tente explorar o facto de se poderem usar objectos de uma classe como valores de variáveis de outros objectos. Note que nesta fase não terá que programar, mas apenas que pensar e definir, para cada classe, a interface (conjunto de operações), assim como os construtores, as variáveis de instância e constantes que achar necessárias.

Quando tiver dúvidas sobre onde colocar uma determinada operação (em que classe), consulte os docentes da disciplina. Para que tudo fique bem documentado, para cada variável indique o seu tipo e explique a finalidade. Para cada método indique o tipo do seu resultado e o tipo dos seus parâmetros (se existirem). Explique ainda a sua finalidade (para que serve) de forma clara e intuitiva. Indique ainda que métodos são modificadores e que métodos são de consulta.

O resultado desta fase é um "esqueleto" do programa em Java, em que as classes só têm variáveis, métodos (cujo corpo é vazio), e comentários explicativos. É um programa que provavelmente não passa no compilador (que dá erros, pois faltam coisas), mas que é o esqueleto da sua solução. E o esqueleto é o mais importante. Se for sólido e bem pensado, o programa vai-se manter de pé quando concluído, caso contrário, vai sair uma "alforreca".

No final desta fase, mostre o resultado da sua planificação aos docentes da disciplina. Estes poderão dar alguns conselhos, ou transmitir-lhe confiança sobre a adequação da sua proposta. Apenas depois desta fase estar concluída é que deve começar a programar.

5.3 ETAPA 3 – Construção do Interpretador de Comandos

Nesta etapa deve focar-se em construir o interpretador de comandos que servirá para executar o seu programa. Deve replicar da forma mais exata que conseguir o enunciado e os comandos nele descritos. Desta forma, terá um programa com a possibilidade de invocar todas as funcionalidades pedidas.

Assim, nesta fase terá necessariamente de criar uma classe especial – `Main` – que contém o programa principal (método `main`), e diversos métodos auxiliares (por exemplo, gestão do input/output), o que permite ao programa interagir com o utilizador. Pode montar um método a representar cada comando especificado neste enunciado, mesmo que, numa primeira fase, alguns desses métodos ainda não façam nada. Depois, poderá preencher esses métodos usando as diversas funcionalidades das sua classes lógicas. Trate de testar as funcionalidades/comandos à medida que vai implementando.

5.4 ETAPA 4 – Construção do programa

Nesta fase, vai a programar as classes do domínio da sua aplicação. Desenvolva cada classe separadamente e teste-a bem antes de a integrar na aplicação. Para testar a classe faça um programa principal onde cria objectos e realiza operações nesses objectos, tal como fez nos primeiros exercícios desta unidade curricular.

Para programar as suas classes, comece por fazer um levantamento das constantes úteis, dos métodos da interface (observadores e modificadores/observadores) e das variáveis de instância. Desenvolva depois o algoritmo e finalmente o código de cada método.

Note que não necessita de ter a sua aplicação completa para a poder começar a testar. Os cenários de teste que lhe são fornecidos são construídos, propositadamente, de modo incremental, para que possa ir implementando e testando, aos poucos, as funcionalidades das suas classes. Os cenários serão publicados num documento juntamente com este enunciado na página do moodle.

Na página do moodle será publicado um conjunto de cenários de teste com possíveis interações com o utilizador para estes problemas. Este conjunto de inputs e outputs associados são semelhantes aos que serão testados na plataforma Mooshak.

4.5 Recomendações Gerais

É muito importante tentar ter sempre um programa funcional, que compile sem erros e faça alguma coisa, mesmo que ainda não tudo o que é requerido. Assim terá mais segurança e uma base sólida para partir para a fase seguinte. É a regra da versão estável, que deve ser seguida por qualquer desenvolvedor de software. Tenha sempre uma versão estável do seu sistema. Nunca comece a trabalhar em novas fases, sem que as anteriores estejam robustas, bem testadas.

Para facilitar o desenvolvimento do seu programa, e eliminar erros desnecessários, deverá ter em atenção os seguintes aspectos:

- Não se esqueça de mostrar o resultado da ETAPA 1 a um docente ou monitor. Não comece a programar antes de terminar a ETAPA 1.
- Teste as suas classes método a método. Se for preciso crie pequenos programas principais para testar (os métodos de) cada classe. Aproveite os problemas no Mooshak para testar as suas classes.
- Cada método deve realizar uma (e só uma) tarefa bem definida. Defina métodos auxiliares (privados) sempre que tal clarificar a divisão de tarefas e legibilidade do seu programa.
- Use bons nomes para os identificadores. Siga as regras de estilo adoptadas no standard da disciplina, disponível no moodle.
- Quando definir a classe Main e o respectivo método main, vá adicionando funcionalidades à classe conforme for desenvolvendo as várias funcionalidades do sistema.
- A classe Main deverá ser organizada em vários métodos estáticos e privados, que implementam cada comando do sistema, podendo também haver métodos auxiliares. O estilo da programação é **MESMO** muito importante.
- A classe Main é a única que pode utilizar operações de input/output de ou para o ecrã (println, etc.). Nenhum método de uma classe auxiliar pode conter operações de input/output.
- Todas as variáveis de estado dos objectos (variáveis de instância) TÊM que ser privadas (declaradas private).
- Não se esqueça da regra da "versão estável". No fim de cada fase de desenvolvimento ou alteração ao programa, ele deve compilar e funcionar como se espera nessa fase. Nunca esteja muito tempo sem que o seu programa compile sem erros, ou não faça o que era previsto fazer nessa fase! As versões estáveis são uma rede de segurança. Nunca apague o código de cada versão estável, para poder voltar a ela, se se perder no desenvolvimento ou se começar a ficar com um programa muito confuso.
- Se não percebe como o SEU programa funciona, dificilmente os docentes o vão perceber quando o forem avaliar! Não escreva código que não entende como funciona. Em caso de dúvida, estude, consulte livros, os docentes, os colegas. **MUITO IMPORTANTE**: comente sempre o seu código, incluindo pré-condições.
- Quando, ao compilar, o seu programa der um erro, tente perceber bem porque isso aconteceu, para não repetir o mesmo erro muitas vezes. Leia com atenção as mensagens que lhe são mostradas.

6 A Entrega

O seu programa deve ser entregue no concurso do mooshak para o efeito (Concurso IP201718_TP1), até ao dia 10 de Novembro de 2017 pelas 17:00 (hora de Lisboa).

Cada aluno deve registar-se neste concurso. O nome do utilizador deve ser o seu **número de aluno** no grupo correspondente ao seu turno (por exemplo, o aluno nº 3435 do turno p1 deve ter como nome utilizador no mooshak **3435** no grupo **TP1**). Só serão aceites como entregas para avaliação, os programas cujo utilizador no concurso do Mooshak siga estas regras.

Para o trabalho ser avaliado, o aluno deve ter cumprido o código de ética do DI (ver na página do moodle), e ter, no final do prazo, uma entrega no concurso do mooshak. Pode submeter o seu código mais que uma vez. Será a última submissão que será avaliada.

A avaliação do seu trabalho tem 3 componentes, sendo a nota final calculada como a soma das notas dessas componentes:

- Avaliação da correcção dos resultados produzidos: até 12 valores
  - Submissão ao concurso obtendo a pontuação máxima (100 pontos), com pelo menos duas classes (Main e Jogo) bem definidas terá 12 valores nesta componente. Note que muito provavelmente vai necessitar de mais do que duas classes para fazer este projecto **bem**.
  - Uma submissão com erros em algumas das funcionalidades terá uma classificação correspondente às funcionalidades correctamente implementadas.
- Avaliação da qualidade do código: 7 valores
- Submissões realizadas com as tarefas realizadas durante as aulas práticas ao longo do semestre, até ao momento de entrega do trabalho: 1 valor

Por exemplo, um aluno que entregue uma aplicação que receba 90 pontos terá 90% dos 12 pontos, a que se somam entre 0 e 7 pontos pela qualidade do código e entre 0 e um ponto pela entrega dos trabalhos das aulas práticas anteriores a este projecto.

7 Descrição dos ficheiros de teste

Nesta secção descrevemos os ficheiros de teste. Os ficheiros de teste disponibilizados são semelhantes aos que serão usados para testar a sua aplicação no mooshak. Assim, se o seu trabalho passar nos testes disponibilizados, também passa nos do mooshak.

Os testes do Mooshak verificam de forma incremental a implementação dos vários comandos:

Ficheiro de teste `01_in.txt` (5 pontos)
Comandos testados: `sai`
Apresenta o menu e sai da aplicação.

Ficheiro de teste `02_in.txt` (5 pontos)
Comandos testados: `ajuda, sai`
Testa o comando de ajuda um par de vezes, fora de um jogo, e sai.

Ficheiro de teste `03_in.txt` (5 pontos)
Comandos testados: `novo, mapa, sai`
Cria um novo jogo correctamente, mostra o mapa e sai.

Ficheiro de teste `04_in.txt` (5 pontos)
Comandos testados: `novo, mapa, sai`
Cria vários novos jogos correctamente, mostrando os mapas e sai.

Ficheiro de teste `05_in.txt` (5 pontos)
Comandos testados: `novo, mapa, ajuda, sai`
Cria vários novos jogos correctamente, mostrando os mapas, ajudas, e sai.

Ficheiro de teste `06_in.txt` (5 pontos)

Comandos testados: `novo, mapa, sai`

Cria novos jogos, alguns incorrectamente, com mapas pequenos demais, tenta mostrar os mapas e sai.

Ficheiro de teste `07_in.txt` (5 pontos)

Comandos testados: `novo, mapa, sai`

Cria novos jogos, alguns incorrectamente, por má colocação das bandeiras, tenta mostrar os mapas e sai.

Ficheiro de teste `08_in.txt` (5 pontos)

Comandos testados: `novo, mapa, sai`

Cria novos jogos, alguns incorrectamente, por duplicação dos nomes de equipas, tenta mostrar os mapas e sai.

Ficheiro de teste `09_in.txt` (5 pontos)

Comandos testados: `novo, mapa, espadachim, sai`

Cria um novo jogo correctamente, mostra o mapa, movimenta os espadachins para posições livres e sai.

Ficheiro de teste `10_in.txt` (5 pontos)

Comandos testados: `novo, mapa, lanceiro, sai`

Cria um novo jogo correctamente, mostra o mapa, movimenta os lanceiros para posições livres e sai.

Ficheiro de teste `11_in.txt` (5 pontos)

Comandos testados: `novo, mapa, cavaleiro, sai`

Cria um novo jogo correctamente, mostra o mapa, movimenta os cavaleiros para posições livres e sai.

Ficheiro de teste `12_in.txt` (5 pontos)

Comandos testados: `novo, mapa, cavaleiro, espadachim, lanceiro, ajuda, sai`

Cria um novo jogo correctamente, movimenta os elementos das equipas para posições livres e sai, usando também o mapa e a ajuda.

Ficheiro de teste `13_in.txt` (5 pontos)

Comandos testados:`novo, mapa, cavaleiro, espadachim, lanceiro, ajuda, sai`

Semelhante ao teste 12, mas detectando tentativas de saída do mapa.

Ficheiro de teste `14_in.txt` (5 pontos)

Comandos testados:`novo, mapa, cavaleiro, espadachim, lanceiro, ajuda, sai`

Semelhante ao teste 12, mas detectando colisões com a própria bandeira e com soldados da mesma equipa.

Ficheiro de teste `15_in.txt` (5 pontos)

Comandos testados:`novo, mapa, cavaleiro, espadachim, lanceiro, ajuda, sai`

Semelhante ao teste 12, mas detectando a captura de bandeira.

Ficheiro de teste `16_in.txt` (5 pontos)

Comandos testados:`novo, mapa, cavaleiro, espadachim, lanceiro, ajuda, sai`

Mistura os testes 12 a 15, com combates em que vence o atacante.

Ficheiro de teste `17_in.txt` (5 pontos)

Comandos testados:`novo, mapa, cavaleiro, espadachim, lanceiro, ajuda, sai`

Mistura os testes 12 a 15, com combates em que vence o defensor.

Ficheiro de teste `18_in.txt` (5 pontos)

Comandos testados:`novo, mapa, cavaleiro, espadachim, lanceiro, ajuda, sai`

Mistura os testes 16 e 17, com combates em que vence o atacante e outros em que vence o defensor.

Ficheiro de teste `19_in.txt` (5 pontos)

Comandos testados:`novo, mapa, cavaleiro, espadachim, lanceiro, ajuda, sai`

Semelhante ao 18, mas com a vitória a ser atingida, também, pela via da "morte" de todos os soldados de uma equipa.

Ficheiro de teste `20_in.txt` (5 pontos)

Comandos testados:`novo, mapa, cavaleiro, espadachim, lanceiro, ajuda, sai`

Programa completo, com todas as alternativas a ser testadas ao longo de vários jogos numa sessão.