Lab XIII.

Objetivos

Os objetivos deste trabalho são:

- Aplicar conceitos de programação adquiridos em aulas anteriores:
- Aplicar conceitos de modulação de software necessários no desenvolvimento de uma solução
- Rever e consolidar competências de desenvolvimento de software

Condições

Este trabalho deve ser realizado no prazo de duas semanas. Na primeira semana, é necessário implementar a parte de leitura da configuração (XML) e o gerador de tabuleiro (com a distribuição dos barcos). O restante trabalho deverá ser realizado durante a segunda semana.

XIII.1 Jogo batalha naval

O objetivo deste projeto é desenvolver uma versão Java do popular jogo de tabuleiro "Batalha Naval". Deverá ser desenvolvida uma solução completa; preste atenção à modulação; decomponha a solução em diferentes módulos para tomar partido de todas as capacidades da programação orientada por objetos; e use padrões de desenho de software adequados à solução.

O jogo deverá iniciar com o carregamento das configurações de tabuleiro de um ficheiro de entradas. Neste ponto, é apenas requerido desenvolver uma versão para um único jogador. Nesta versão, o jogador irá tentar afundar todos os navios do computador no menor número de jogadas possível. O jogo deverá iniciar com a pergunta do nome do jogador. De seguida, o computador deverá colocar os seus navios aleatoriamente. Nesse momento, o jogador começa a disparar aos barcos do computador, um único tiro por jogada. O jogo termina quando todos os navios sejam "afundados". É atribuída uma classificação de acordo com a dificuldade do jogo, assim como o desempenho do jogador.

O programa deverá disponibilizar um menu onde é possível:
a) Iniciar novo jogo; b) Continuar o jogo anterior; e c)Ver a tabela de classificações

Não é necessário seguir uma interface predefinida. No entanto, o jogo deverá respeitar a configuração bem como as especificações do *logger*. Todas as jogadas deverão ficar registadas no *logger* e, ao fazê-lo, o jogo poderá ser interrompido e retomado mais tarde. Para além disso, o jogo terá de ser compatível com a versão do jogo de outros colegas. Os ficheiros de configuração e *logging* deverão ser chamados, config.xml e log.txt, respectivamente.

Para mais informações sobre este jogo, por favor visite: Battleship at Wikipedia

(a) Configurações

O projeto deverá ter um XML como ficheiro de configuração. Juntamente com esta proposta é disponibilizado um exemplo correto do que é pretendido. Deverá cumprir integralmente com as suas especificações.
De modo a carregar a configuração, é sugerida a utilização da Apache Commons Configuration Library. Esta biblioteca é a interface de configuração em Java mais utilizada. Disponibiliza um conjunto de ferramentas para lidar com os múltiplos formatos de configuração. No entanto, para este problemas, apenas é necessário a *Hierarchical Configuration* (*XMLConfiguration* class) – ver exemplo no user's guide. A secção "g" também disponibiliza um exemplo para ajudar no arranque do projeto.

- Tamanho do tabuleiro: Número de linhas e colunas no tabuleiro (<=25, cada). Ex. "8x10".
- Navios: (Nome, número, tamanho)
- Navios complexos: (nome, número, matriz). Estes navios não estão incluídos no jogo original. Podem assumir vários formatos, em vez de apenas um segmento com uma linha simples. A sua forma é mapeada como uma matriz no ficheiro de configuração. Cada dígito mapeia uma única célula, "0" são as células não ocupadas, enquanto "1" são as células ocupadas. Todas as linhas devem conter o mesmo número de células.
- Abaixo encontra-se um exemplo do ficheiro de configuração. Também é disponibilizado separadamente no repositório. O resto deste documento tem como referência esta configuração.

```
<configuration>
	<board>
		<rows>8</rows>
		<collumns>10</collumns>
	</board>
	<ships>
		<ship>
			<name>Aircraft carrier</name>
			<number>1</number>
			<size>5</size>
		</ship>
		<ship>
			<name>Battleship</name>
			<number>1</number>
			<size>4</size>
		</ship>
		<ship>
			<name>Submarine</name>
			<number>1</number>
			<size>3</size>
		</ship>
		<ship>
			<name>Destroyer</name>
			<number>1</number>
			<size>2</size>
		</ship>
		<ship>
			<name>Patrol boat</name>
			<number>2</number>
			<size>1</size>
		</ship>
		<complex-ship>
			<name>SuperShip</name>
			<number>1</number>
			<matrix>
				<row>0100</row>
				<row>1111</row>
				<row>0100</row>
			</matrix>
		</complex-ship>
	</ships>
</configuration>
```

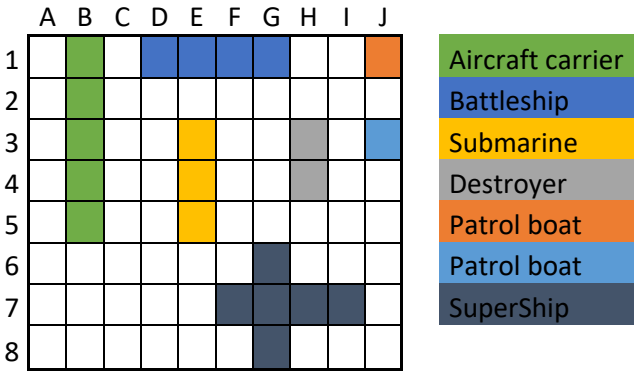
(b) Colocação de navios

Deve desenvolver um distribuidor aleatório dos navios. Quando o jogador inicia novo jogo, deve colocar os navios de acordo com as seguintes regras;

- Os tipos de navios devem ser descarregados do ficheiro de configuração. Note que é possível ter vários navios do mesmo tipo (*number*).
- O tamanho do navio indica o número de células necessárias.
- Navios normais devem ser colocados horizontal ou verticalmente (em qualquer direção). Navios complexos não podem ser rodados, devem ser colocados exatamente seguinte a matriz.
- Os navios não devem ser colocados em células contíguas.

(c) Tabuleiro

- O tamanho do tabuleiro deve ser descarregado das configurações. Os endereços das células devem seguir o exemplo de um tabuleiro "8x10".
- O sistema de coordenadas deve seguir o exemplo dado abaixo. As colunas são identificadas pelas letras (A a Z), enquanto as linhas são identificadas por números (1 a 25). Assim sendo, a primeira célula deverá ser "A1".



a)

(d) Logger/Registo

O ficheiro log é a interface de saída mais importante do projeto. Todos os jogos, e cada movimento, deverão ser guardados no log file. Ao fazer isso, deverá ser possível parar e continuar jogos, guardar estatísticas de jogos anteriores e fazer o mesmo para os jogos/projetos de colegas. Deverá respeitar integralmente o seguinte formato.

- Cada entrada de jogo começa por logging: *"*`===GAME N===`*"*, em que "N" é o número de jogo.
- Depois de identificar o jogo, deve carregar as definições de jogo de acordo com o seguinte esquema:

a)

```
Board:ROWSxCOLUMNS
Player1:ThePlayerName
Boat Name:[CELL1,CELL2,...,CELL(n)]
```

b)

- Para fins de depuração, é obrigatório imprimir a matriz de tabuleiro. Primeiro imprimindo a tag "===Matrix===". De seguida, imprimindo cada linha numa linha separada. Use a mesma anotação do ficheiro de configuração, substituindo os zeros por espaços.
- Escrever **"**`===START===`**"** para marcar o início do registo de movimentos.
- Os movimentos do jogador devem ser registados da seguinte forma:

c)

```
MovementNumber:Player1:[CELL]:Custom Message!! as you like.
```

d)

e)

- Escrever **"**`===END===`**"** para marcar o fim do registo desse jogo. Esta linha final só deve aparecer se o jogo for terminado com sucesso. Se o jogo for interrompido com pelo menos um navio no tabuleiro, este marcador não deve ser escrito, para permitirá continuar o jogo mais tarde.
- Caso se inicie um novo jogo deverá ser anotado pela tag *Start*.
- Abaixo encontra-se um exemplo de um ficheiro log. A maioria dos movimentos dos jogadores está cortada por razões de simplicidade.

```
===GAME 0===
Board:8x10
Player:ThePlayerName
Aircraft carrier:[B1;B2,B3,B4,B5]
Battleship:[D1,E1,F1,G1]
Submarine:[E3,E4,E5]
Destroyer:[H3,H4]
Patrol boat:[J1]
Patrol boat:[J3]
SuperShip:[H6,F7,G7,H7,I7,J7,G8]
===MATRIX===
0101111001
0100000000
0100100101
0100100100
0100100000
0000001000
0000011110
0000001000
===START===
0:Player1:[A1]:Missed Shot
1:Player1:[J1]:Bull's eye!! Patrol Boat Sunk
2:Player1:[G7]:Enemy's SuperShip was hit, (1/7)
…
===END===
```

f)
g)

- Se o utilizador quiser continuar o jogo "N", deve iniciar fazendo registo da tag de jogo ("GAME2", por exemplo), seguido da tag de continuação ("RESUME").

h)

```
===GAME 2===
===RESUME===
25:Player1:[J5]:Bull's eye!! Patrol Boat Sunk.
Congratulations, all the ships have been sunk.
===END===
```

i)

(e) Estatísticas

- O projeto deve ser capaz de carregar jogos anteriores e disponibilizar algumas estatísticas.
- Sistema de ranking: Fornecer uma classificação contendo o nome e pontuação do jogador. Os pontos devem ser calculados usando a seguinte fórmula:

a)

```
NumberOfPlayedRound * DificultyIndex;
DificultyIndex = SizeBoard/TotalBoatSize + 1/AverageBoatSize;
```

b)
c)
d) Em que:
   - SizeBoard → número de células no tabuleiro
   - TotalBoatSize → número de células ocupadas pelos barcos
   - AverageBoatSize → média ponderada do tamanho dos barcos

(f) Características especiais

Incentivamos a inclusão de características especiais no jogo:

- Radar, com as tentativas anteriores do jogador. Tal como no jogo de tabuleiro, isto deverá ajudar os jogadores a relembrar os seus movimentos, evitando

disparar na mesma célula.

- Games's Hit Ratio: Fornecer uma estatística simples, com os tiros certeiros e os falhados de cada jogador.
- Navios restantes: dar informação sobre o número de navios restantes do adversário.
- Indicar o número da ronda
- Desenvolver uma interface gráfica de utilizador utilizando SWING, HTML ou outra ferramenta.

(g) Exemplo de configuração Apache XML

Abaixo encontra-se um pequeno exemplo da utilização de ficheiros XML com a biblioteca Apache Commons Configuration.

```
try {
     //Load the configuration from the xml file
     XMLConfiguration config = new XMLConfiguration("config.xml");

     //Retrieve a simple configuration: (int)
     int rows = config.getInt("board.rows");

     //Retrieve multiple nested configuration.
     // A: Using the configurationsAt function.
     List<HierarchicalConfiguration> shipCnfs = config.configurationsAt("ships.ship");
     for(HierarchicalConfiguration shipC : shipCnfs){
             String name = shipC.getString("name");
     }

     // B: Assembling manually the properties key.
     int maxIndex = config.getMaxIndex("ships.ship");
     for(int i = 0; i<= maxIndex; i++){
             String key = String.format("ships.ship(%d).", i);
             String name = config.getString(key+"name");
     }
} catch (ConfigurationException e) {}
```

Nota importante: para cada guião prático, deverá ser usada no *git* uma nomenclatura uniforme (*lab01, lab02, lab03,…*) para permitir uma identificação mais fácil dos projetos.

Bom trabalho!