Lab III.

Objetivos

Os objetivos deste trabalho são:

- Aplicar conceitos de modulação de software necessários no desenvolvimento de uma solução
- Rever e consolidar competências de desenvolvimento de software

III.1 Jogo do Galo

Pretende-se desenvolver uma versão simples do "Jogo do Galo". Para tal são fornecidos os seguintes módulos (no dossier das aulas práticas):

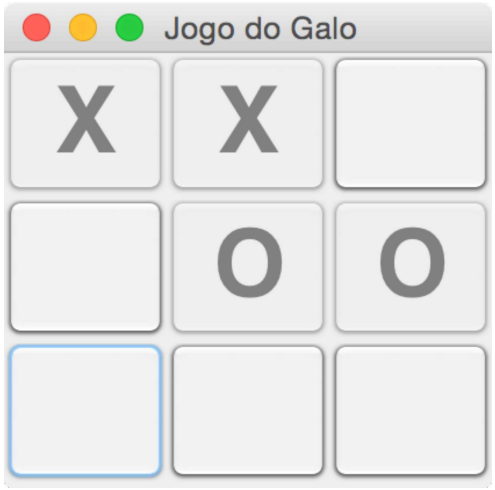
a) A aplicação visual do jogo, classe *JGalo*, desenvolvida sobre java Swing e que não precisa de ser modificada para este trabalho. Apesar disso, recomenda-se a sua análise cuidada.

b) A interface *JGaloInterface* que irá servir de ligação entre a classe *JGalo* e o módulo que terá de desenvolver.

```
public interface JGaloInterface {
    public abstract char getActualPlayer();
    public abstract boolean setJogada(int lin, int col);
    public abstract boolean isFinished();
    public abstract char checkResult();
}
```

Considere que o programa apenas executa o jogo uma vez, começando com cruzes (X) ou bolas (O) consoante o argumento inicial (por omissão, considere X).

Nota: No dossier existe ainda um ficheiro executável (`JogoDoGalo.jar`) que permite verificar o comportamento desejado para este programa.



III.2 Voos

O objetivo é desenvolver um programa para gerir voos e reservas. Analise os requisitos e planeie cuidadosamente as interfaces, classes, e estruturas de dados mais adequadas.

Requisitos iniciais

1. Um voo é identificado por um código alfanumérico e tem associado um avião com determinada configuração;
2. Cada avião tem lugares para a classe turística e, opcionalmente, lugares para a classe executiva;
3. O número de lugares é especificado, para cada classe, pelo número de filas e número de lugares por fila. Por exemplo '3x2' corresponde a 3 filas com 2 lugares em cada fila;
4. A classe executiva, se existir, ocupa as primeiras filas, começando na fila 1; a numeração das filas da classe turística continua esta numeração;
5. Os bancos em cada fila são identificados por letras, começando na letra A;
6. Uma reserva de lugares indica a classe (**T**uristíca / **E**xecutiva) e o número de passageiros;
7. Caso não existam lugares suficientes para uma reserva, esta não deve ser efetuada;
8. Na atribuição dos lugares para uma reserva deve primeiro procurar-se uma fila vazia (na classe correspondente), atribuindo os lugares de forma sequencial e continuando na fila seguinte caso necessário; caso não haja filas vazias, deve distribuir-se os lugares vagos sequencialmente, começando na primeira fila (da classe correspondente).

Comandos

O programa deve permitir comandos lidos da consola, conforme indicado:

- **H**: apresenta as opções do menu.

- **I *filename***: Lê um ficheiro de texto contento informação sobre um voo. A primeira linha do ficheiro deve começar com o caracter ">" e indicar o código de voo, o número de filas e lugares por fila em classe executiva (caso exista) e o número de filas e lugares por fila em classe turística. As linhas seguintes, caso existam, contêm reservas já efetuadas, no formato classe, número de lugares, como se vê nos exemplos.

Exemplos de ficheiros:

(flight1.txt)

```
>TP1920 3x2 15x3
T 2
T 4
E 3
T 1
E 2
E 2
E 1
```

(flight2.txt)

```
>TP1930 20x4
E 4
T 6
```

Exemplos de execução:

```
Escolha uma opção: (H para ajuda)
I flight1.txt
Código de voo TP1920. Lugares disponíveis: 6 lugares em
classe Executiva; 45 lugares em classe Turística.
Não foi possível obter lugares para a reserva: E 2

Escolha uma opção: (H para ajuda)
I flight2.txt
Código de voo TP1930. Lugares disponíveis: 80 lugares em
classe Turística.
Classe executiva não disponível neste voo.
Não foi possível obter lugares para a reserva: E 4
```

- **M *flight_code***: exibe o mapa das reservas de um voo, conforme mostra o exemplo. Os lugares reservados são identificados pelo número sequencial da reserva; os lugares livres são identificados pelo número 0.

  Exemplo de execução:

```
Escolha uma opção: (H para ajuda)
M TP1920
   1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18
A  3  3  5  1  2  2  4  0  0  0  0  0  0  0  0  0  0  0
B  3  6  5  1  2  0  0  0  0  0  0  0  0  0  0  0  0  0
C           0  2  0  0  0  0  0  0  0  0  0  0  0  0  0
```

- **F *flight_code num_seats_executive num_seats_tourist*:** acrescenta um novo voo, com código, lugares em executiva (p.ex. 4x3, representando 4 filas com 3 lugares por fila), e lugares em turística. Os lugares em classe executiva são opcionais, podendo existir apenas lugares em turística.

- **R *flight_code class number_seats***: acrescenta uma nova reserva a um voo, com indicação do código do voo, da classe (T / E), e do número de lugares. O programa deve verificar se há lugares disponíveis na classe pretendida. Caso a reserva seja efetuada deve ser apresentado no ecrã o código da reserva no formato *flight_code:sequential_reservation_number* e os lugares atribuídos.

  Exemplo de execução:

```
Escolha uma opção: (H para ajuda)
R TP1930 T 3
TP1930:2 = 3A | 3B | 3C
```

- **C *reservation_code***: cancela uma reserva. O código de reserva tem o formato *flight_code:sequential_reservation_number*.

- **Q**: termina o programa.

Deve permitir que o programa possa ser executado usando um ficheiro de comandos como argumento de entrada. Por exemplo, "`java lab3 ficheiro_de_comandos`".
No dossier da disciplina são fornecidos alguns ficheiros de exemplos de voos e reservas.

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!