IPS Instituto Politécnico de Setúbal
Escola Superior de Tecnologia de Setúbal

PA 2018/19

Programação Avançada

Enunciado de Projeto (Época Normal)

Aplicação para Emissão de Bilhete-Percurso num Parque Biológico

A. Notas

- O presente enunciado descreve as regras, problema, requisitos obrigatórios, entrega e avaliação do projeto de Programação Avançada (PA) para a época normal.
- Leia atentamente o enunciado. Dúvidas sobre o seu conteúdo poderão ser colocados no fórum Moodle destinado ao projeto.
- Quaisquer casos omissos na descrição do problema e requisitos ficam ao critério dos alunos.

B. Regras

- O trabalho é **desenvolvido em grupos de dois alunos** do mesmo docente de laboratório do mesmo docente de laboratório. Qualquer desvio a esta regra tem de ser aceite pelo respetivo docente de laboratório antes da inscrição.
- Projetos não funcionais não serão avaliados, i.e., **um projeto que não permita simular (usando uma interface gráfica) o cálculo de um percurso a pé e emitir um bilhete é automaticamente reprovado.**
- A entrega é realizada através de submissão no Moodle.
- É obrigatória a discussão oral do trabalho. A falta à discussão oral resulta na classificação com 0 (zero).

B1. Conduta Académica

- **Ao submeter o projeto para avaliação, o(s) aluno(s) declara(m), sob compromisso de honra, que o projeto desenvolvido é original e foi desenvolvido pelos mesmos.**
- Projetos copiados total ou parcialmente serão automaticamente cotados com 0 (zero), sem averiguação de qual o original e a cópia.
  - Todos os projetos serão submetidos à plataforma MOSS (https://theory.stanford.edu/~aiken/moss/) para deteção de cópias.

0. Introdução

O projeto consiste no desenvolvimento de uma aplicação de *desktop* para geração de percursos num parque biológico e respetiva emissão de bilhete.

A aplicação tem de ser desenvolvida utilizando TADs, padrões de software e interface gráfica em JavaFX.

A secção **I** explica o objetivo geral da aplicação e na secção II são apresentados os requisitos funcionais e não-funcionais que a aplicação deverá respeitar. A secção **III** informa sobre a data de entrega do projeto e sobre o conteúdo do relatório. Finalmente a secção **IV** contém a tabela de cotações que será aplicada aos projetos submetidos.

I. Problema

Pretende-se desenvolver uma aplicação que permita gerar percursos a pé e de bicicleta dentro de um parque biológico. A aplicação disponibiliza informação sobre o preço total a pagar pelo percurso selecionado e permite ainda a emissão de bilhetes e respetiva fatura.

O parque biológico é constituído por vários pontos de interesses, pontos esses que estão conectados ou por caminhos, ou por pontes; os caminhos podem ser percorridos em qualquer sentido, mas as **pontes só podem ser percorridas num sentido**. Existem **ainda conexões que não permitem a circulação de bicicletas**. Cada conexão tem um custo, e uma distância associada.

Cada percurso **inicia e termina no ponto de entrada do parque**, e poderá passar por vários pontos de interesse selecionados pelo utilizador. O utilizador deverá poder calcular o seu percurso de forma a minimizar a distância, ou o custo do mesmo.

II. Requisitos Obrigatórios

Os seguintes requisitos funcionais e não-funcionais são obrigatórios na resolução do problema.

II.1 Leitura do Mapa do parque

O mapa do parque deve ser configurável, através de um ficheiro de texto com o seguinte formato:

```
n_pontos
id_ponto, ponto
...
n_conexoes
id_c, tipo, nconexao, id_ponto1, id_ponto2, navegabilidade, custo, distancia
...
```

(disponibiliza-se um ficheiro ***mapa1.dat*** como exemplo)

Cada *placeholder* contém um valor de acordo com a seguinte descrição:

- **`n_pontos`**: número de pontos de interesse marcados no mapa; dita também quantos pontos são descritos em seguida no ficheiro – valor inteiro;
- **`id_ponto`**: identificador do ponto do mapa – valor inteiro;
- **`ponto`**: nome do ponto do mapa – texto;
- **`n_conexoes`**: número de conexoes marcadas no mapa; dita também quantas conexões são descritas de seguida no ficheiro – valor inteiro;
- **`id_c`**: identificador da conexao do mapa – valor inteiro;
- **`tipo`**: tipo de ligação – um dos seguintes valores {"ponte", "caminho"};
- **`conexao`** – nome da conexão - texto;
- **`id_ponto1`** e **`id_ponto2`**: identificadores dos pontos que a conexão liga, no caso de ser uma ponte, esta só pode ser atravessada no sentido do ponto1 para o ponto2 – valores inteiros, cujos pontos já foram definidos no início do ficheiro.
- **`navegabilidade`**: true se for passível de atravessar de bicicleta, false, caso contrário – um dos seguintes valores {"true", "false"};
- **`custo`**: custo (€) para a travessia dessa conexão – valor inteiro >= 0;
- **`distancia`**: distância (m) associada a atravessar essa conexão – valor inteiro >= 0;

Assume-se que **a entrada do parque é o primeiro ponto lido no ficheiro**.

O mapa do parque deve ser **guardado em memória através de grafo(s)** deve ser possível visualizá-lo na aplicação.

Um "**java properties file**" deve informar a aplicação no arranque de qual o ficheiro (filename) a carregar.

O projeto deve **incluir outros dois mapas** (a gerar manualmente pelos alunos), além daquele disponibilizado; cada mapa tem de ter no mínimo 7 locais e 14 ligações.

II.2 Cálculo do Percurso

O utilizador deve poder selecionar:

- Tipo de percurso (a pé ou de bicicleta);
- Critério a minimizar (custo ou distância);
- Pontos de interesse a visitar – a aplicação deverá obrigar a selecionar pelo menos um ponto de interesse, e permitir a introdução de pelo menos 3 pontos de interesse.

O percurso deverá ser calculado tendo como base **o algoritmo Dijkstra**. O percurso inicia-se e termina no **ponto de entrada**.

Deve ser possível visualizar o percurso calculado na interface da aplicação.

O utilizador deverá poder modificar os dados introduzidos para escolha do percurso, assim como iniciar o cálculo de um novo percurso.

II.4 Emissão de Bilhetes

O utilizador poderá solicitar a emissão de um bilhete, apenas após o cálculo de um percurso. O bilhete deverá conter informação sobre o tipo de percurso selecionado, uma descrição detalhada do mesmo, indicando todas as conexões a percorrer e a distância das mesmas, assim como o valor total do percurso e a data e hora de emissão do mesmo.

O utilizador poderá decidir se a fatura associada ao bilhete é passada com ou sem NIF. Toda a fatura tem uma data. A fatura é emitida automaticamente em conjunto com o bilhete, e resulta na criação de dois documentos PDF (um para o bilhete e outro para a fatura). O nome do ficheiro gerado deverá ter o formato <NIF>_<DATA>_bilhete.pdf ou <NIF>_<DATA>_fatura.pdf.

II.3 Retroceder

Se o utilizador ainda não escolheu a opção de emitir bilhete **é possível retroceder às simulações de percurso realizadas anteriormente**.

II.6 Estatísticas

A aplicação deverá permitir que o utilizador consulte estatísticas sobre:

- Os 10 pontos de interesse mais visitados (só os bilhetes emitidos para estes pontos de interesse devem ser contabilizados), visualizados sobre a forma de gráfico de barras, com o número total de visitas de cada um deles;
- O preço médio dos bilhetes vendidos;
- Um gráfico *pie chart* que mostre a percentagem e bilhetes vendidos para percursos a pé e de bicicleta.

II.6 Pesistência de Dados

A aplicação deverá permitir que a informação dos bilhetes, faturas emitidas e das estatísticas seja persistida, i.e., guardada, através de *java persistence (serialization)* ou numa base de dados SQLite.

Um "java properties file" (pode ser o mesmo referido na secção II.1) deve informar a aplicação no arranque que mecanismo utilizar.

Nota: A informação contida em cada um dos tipos de persistência não necessita de ser sincronizada. Consequentemente, o suporte de cada tipo de persistência podem conter diferentes dados.

II.7 Logging

A aplicação deverá permitir registar as ações realizadas na aplicação. **Este registo deve ser configurável**. A configuração deverá permitir decidir o que se quer registar: a emissão de bilhetes, os cálculos de percursos, a consulta às estatísticas.

Recomenda-se que a configuração seja efetuada no mesmo "java properties file" mencionado na secção II.1.

II.6 Padrões de Software

A utilização de padrões de software é essencial no desenvolvimento e cumprimento dos requisitos funcionais e anteriores.

É esperado que se utilizem alguns dos padrões de software lecionados na unidade curricular, sendo a sua aplicabilidade a cada requisito funcional da competência do(s) aluno(s) programador(es).

II.7Javadoc

O código deverá ser documentado com recurso ao Javadoc, precedendo todo e qualquer método da descrição do seu propósito (excetuam-se métodos construtores, seletores e modificadores). Comentários adicionais deverão ser colocados em trechos de código onde a compreensão por parte de terceiros possa ser menos óbvia.

III. Entrega e Discussões

O projeto obedece a uma **milestone** intermédia (25%) e a uma entrega **final** em duas fases (75%) - versão funcional e após *refactoring* de bad-smells detetados na versão anterior. **Após a penúltima submissão (versão funcional) não podem ser adicionadas novas funcionalidades nem correções às funcionalidades existentes.**

Chama-se a atenção o seguinte:

- A não entrega da **milestone** não impossibilita as restantes entregas, mas implica a perda total da pontuação referente à mesma (25% da nota total);
- **Se for entregue um projeto não-funcional (um projeto que não permita simular, usando uma interface gráfica, o cálculo de um percurso a pé e emitir um bilhete), o projeto é reprovado e não é aceite a entrega final;**

- As entregas serão penalizadas com 0,5 (meio) valor por cada hora de atraso. A primeira hora é considerada 15min após a hora limite (23h59), hora do servidor;
- As discussões são obrigatórias para todos os elementos do grupo de trabalho;
- O conteúdo das entregas correspondem aos respetivos projetos Netbeans desenvolvidos + respetivo relatório em PDF, contidos num ficheiro ZIP/RAR/7z e cujo nome obedeça ao seguinte formato: **<num1>__<num2>.{zip, rar, 7z}**.

Entrega Milestone via Moodle: 21 de novembro (quarta-feira) de 2018. Apresentações presenciais a agendar posteriormente.

Entrega Pré-final via Moodle: 9 de janeiro (quarta-feira) de 2019. Apresentações presenciais a agendar posteriormente.

Entrega Final via Moodle: 28 de janeiro (segunda-feira) de 2019. Discussões a serem agendadas posteriormente.

Para as entregas existirão links de submissão relativos a cada turma de laboratório. Em algum caso onde membros do grupo pertençam a turmas diferentes, a submissão deverá ser feita em apenas um dos links.

III.1 Milestone

A milestone (MS) corresponde à entrega de:

Um projeto NetBeans contendo:

1. Implementação de um TAD DiWeightedGraph (variante das apresentadas nas aulas TP);
2. Classe **GestorPercurso** que inclua uma instância da implementação anterior e que contenha métodos responsáveis por ler um mapa do ficheiro, inserir um pedido de percurso e calcular um percurso mais curto, para a versão base: a pé, sem pontes.
3. Testes JUnit para as classes implementadas nos pontos 1 e 2.

Um mini-relatório contendo:

4. uma mockup da interface gráfica que será desenvolvida (não é vinculativa para a fase seguinte), juntamente com uma breve descrição da forma como um utilizador emitirá um bilhete.

III.2 Entrega Final

A entrega final é efetuada em duas fases. Cada fase tem uma data de submissão distinta e correspondentes discussões.

Versão Pré-Final (funcional)

Corresponde à submissão de uma aplicação JavaFX que resolva o problema proposto em (I) e que obedeça aos requisitos estabelecidos em (II).

Projetos não-funcionais, i.e., um projeto que não permita simular, usando uma interface gráfica, o cálculo de um percurso a pé e emitir um bilhete não será avaliado.

Versão Final

A versão final corresponde à aplicação submetida após *refactoring* mais o relatório.

A adição de novas funcionalidades e/ou correção de problemas anteriores não será contabilizada.

O relatório deverá conter:

- A especificação de TADs implementados;
- Diagrama de classes;
- Documentação de todas as classes (sugere-se reutilizar a documentação Javadoc);
- **Descrição/uso/justificação dos padrões de software utilizados**;
- Refactoring:
  - Uma tabela com os tipos de *bad-smells* detetados, quantas situações deste tipo foram detetadas e técnica de *refactoring* aplicada para a sua correção.
  - Um exemplo de cada correção efetuada, com código antes e após *refactoring*.

IV. Critérios de Avaliação

A aplicação deverá ser implementada na linguagem Java segundo as boas práticas de POO, utilizando os Tipos Abstratos de Dados (TAD) lecionados e os Padrões de Software relevantes.

É fornecida, a título de suporte, a tabela de cotações do projeto. A cotação de cada item será ponderada pela qualidade da solução/código obtida.

Fase	Item	Cotação/ Penalização
Milestone (25%)	Implementação do TAD DiWeightedGraph	30
	Classe GestorPercurso	40
	Implementação do algoritmo Dijkstra	40
	Testes Junit	60
	Mini-Relatório	20
	Javadoc	10
	TOTAL	**200**

Entrega Final (75%)	Cálculo dos vários tipos de percurso	30
	Mapas de parque suplementares (criados manualmente)	10
	Undo	15
	Logger	15
	Persistência	20
	Estatísticas	20
	Emissao do Bilhete	20
	Interface Grafica (Usabilidade)	20
	Javadoc	10
	Existência de bad smells (por cada não tratado)	-5 (até -30)
	Não utilização de padrões de software adequados (por cada situação)	-15 (até -60)
	Relatório	40
	TOTAL	**200**

(fim de enunciado)