IPS Instituto Politécnico de Setúbal
Escola Superior de Tecnologia de Setúbal

# PA 2020/21

Programação Avançada

**Enunciado de Projeto**
**(Época Normal)**

Social Network

## 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 três ou quatro alunos** do mesmo docente de laboratório. Qualquer desvio a esta regra tem de ser aceite pelo responsável da UC antes da inscrição.
- Existe um projeto template para desenvolvimento do projeto, este é acedido através da resposta ao "assigment" do GitHub-Classroom em: https://classroom.github.com/g/RRc3wsIb
- Projetos não funcionais não serão avaliados, i.e., **um projeto que não permita** (usando uma interface gráfica) a visualização de relações diretas e indiretas entre utilizadores da rede social em forma de grafo, **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 é o original e qual é a cópia.
  - Todos os projetos serão submetidos à plataforma MOSS (https://theory.stanford.edu/~aiken/moss/) para deteção de cópias.

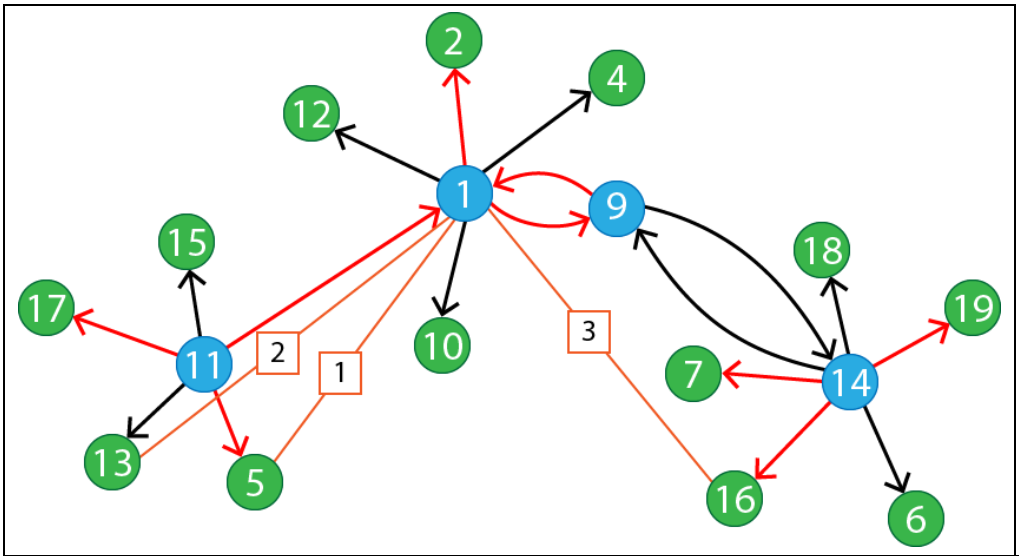
## 0. Introdução

O projeto consiste no desenvolvimento de uma aplicação de *desktop* para visualização e interação com o grafo social subjacente a uma rede social.

A aplicação tem de ser desenvolvida utilizando ADTs , 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




[relationships.csv]
1;2;4;9;10;12
2;1;12
3;2;4;18
4;5;7
5
6;20;18
7;9;18
8;2;4;20
9;1;14
10;4;8;2
11;1;5;13;15;17
12;2;13
13;16;19
14;6;7;9;16;18;19
15;2;3;4;16
16;15;13
17;6;15;19
18;4;5;10
19;2,17;18
20;2,3;5;7;19

[interests.csv]
1;2;
2;1;13;16;20
3;1;9;13;16
4;5;13
5;2;4;18
6;18;19
7;1;5;12;16
8;2;10
9;20
10;11;13;17

Legenda:

- Vértices azuis representam utilizadores adicionados
- Vértices verdes utilizadores incluídos, arestas pretas relações diretas,
- Arestas vermelhos relações diretas com partilha de interesses,
- Arestas laranja relações indiretas (anotadas com a quantidade de interesses partilhados).

O grafo representa parte da informação contida nos ficheiros relationships.csv e interests.csv exemplo.
A representação parcial foi gerada pela adição dos utilizadores 1, 9, 11 e 14, e pela adição das relações indiretas do utilizador 1.

**Figura 1 - Representação gráfica do modelo.**

Pretende-se desenvolver uma aplicação que represente relacionamentos de uma rede social num grafo. (ver Figura 1). Além de relacionamentos, deve ser possível representar relações entre utilizadores e partilha de interesses. A partilha de interesses entre utilizadores não relacionados representa uma oportunidade para recomendar novos relacionamentos.

A aplicação deve permitir inspecionar a estrutura do grafo e, através deste, permitir analisar as relações atuais de um utilizador, e identificar utilizadores não relacionados que partilham os mesmos interesses.

## II. Requisitos

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

## 1 Construção do modelo (grafo social)

A aplicação deve permitir carregar informação relativa a uma rede social de um conjunto de ficheiros e construir a representação em grafo dessa informação (ver figura 1), incluindo relacionamentos entre utilizadores, e os seus respetivos interesses -- este grafo será doravante denominado por **modelo**.

A informação contida nos ficheiros de dados (ver ponto 9) deve ser totalmente carregada para estruturas de dados auxiliares que irão apoiar a construção do modelo

Os vértices guardam informação sobre os utilizadores que podem ser do tipo adicionado ou incluído. As arestas representam relacionamentos entre utilizadores (e podem ser de três tipos Direto simples, Direto com partilha de interesses e Indireto. (ver tabela 1)

Os relacionamentos diretos são **unidirecionais**. Os relacionamentos indiretos são **bidirecionais**.

Com a informação carregada nas estruturas de dados auxiliares, devem ser suportadas duas estratégias para construção do modelo:

- **Carregamento total**: todos os utilizadores e todos relacionamentos diretos, são adicionados numa só operação.
- **Iterativa**: um utilizador é individualmente adicionado ao modelo a pedido, sendo os seus relacionamentos diretos adicionados automaticamente.

Na versão Iterativa a construção do modelo deve seguir as seguintes regras

1. O modelo deve ser inicializado vazio
2. A qualquer momento, deve ser possível adicionar utilizadores com base no seu identificador. Estes utilizadores são denominados **utilizadores adicionados.**
3. Ao acrescentar um utilizador ao modelo, os seus **relacionamentos diretos** são também acrescentados, implicando a inclusão dos utilizadores relacionados no modelo, que serão denominados como **utilizadores incluídos**.
4. Um utilizador adicionado vai ter todos os seus relacionamentos representados no modelo.
5. Um utilizador incluído tem apenas os seus relacionamentos com utilizadores adicionados.
6. É possível adicionar um utilizador que já está representado no modelo como incluído, passando neste caso a ser um utilizador adicionado e aplicando-se as regras 3 e 4.
7. Caso dois utilizadores que estão relacionados diretamente, partilhem interesses então o relacionamento entre estes designa-se por **relacionamento direto com partilha de interesses**
8. A qualquer momento, deve ser possível selecionar um utilizador adicionado e **solicitar a inserção no modelo dos seus relacionamentos indiretos**. Os relacionamentos indiretos identificados só devem incluir utilizadores existentes no modelo.

**Tabela 1 - Tipo de Utilizadores e Relacionamentos**

| | | |
|---|---|---|
| **Utilizador** | Adicionado | Utilizadores que foram explicitamente adicionados ao modelo, através do seu identificador. |
| | Incluído | Utilizadores incluído no modelo, porque estão relacionados com os utilizadores adicionados, e foram inseridos automaticamente. |
| **Relacionamento** | Diretos simples | Relacionamento direto entre dois utilizadores, estes relacionamentos existem definidos nos dados da rede social (ficheiro relantionships.csv). E estes utilizadores não partilham interesses. |
| | Diretos com partilha de interesses | Relacionamento direto entre dois utilizadores, estes relacionamentos existem definidos nos dados da rede social (ficheiro relantionships.csv). E estes utilizadores partilham pelo menos um interesse. |
| | Indiretos | Relacionamento "virtual" que representa a partilha de interesse entre dois utilizadores. Esta informação não vem explicita nos dados do ficheiro relantionships.csv. |

A funcionalidade de **Undo** deve ser disponibilizada, permitindo a qualquer momento da construção do modelo em modo iterativo remover o último elemento adicionado, seja este a adição de um utilizador, ou de um relacionamento indireto:

- Se o último elemento for a adição de um utilizador, esta ação deve remover os seus relacionamentos diretos, e eventuais utilizadores incluídos (não adicionados) nesses relacionamentos;
- Se os últimos elementos forem relacionamentos indiretos gerados simultaneamente (i.e., um ou mais), esta ação deve remover as arestas correspondentes aos relacionamentos gerados.

## 2 Visualização / Interação

a) Deve ser possível visualizar o modelo construído, tal como sugerido na figura 1.
b) Os vértices correspondentes a utilizadores adicionados devem ter uma cor distinta dos incluídos;
c) As arestas correspondentes a relacionamentos diretos devem ter uma cor distinta dos indireto
d) As arestas que suportam os relacionamentos indiretos devem vir anotadas com o número de interesses partilhados.
e) No caso da construção do modelo no modo iterativo, deve ser possível visualizar o modelo em tempo-real, i.e., à medida que utilizadores e relacionamentos são adicionados ao modelo (de acordo com as regras estabelecidas em 1).

f) Deve ser possível seleciona um vértice e visualizar a informação associada a esse, nomeadamente, o nome do utilizador, e a sua lista de interesses;

g) Deve ser possível selecionar uma aresta que represente uma relação indireta e visualizar a lista de interesse que a justificam.

## 3 Persistência

Deve ser possível **exportar** (guardar em ficheiro) um modelo.
Os tipos de ficheiro a suportar **devem ser**:

1. *java persistence (serialization)*
2. formato JSON.

Deve então ser também possível **importar** um modelo previamente guardado, em qualquer um dos dois formatos.

## 4 Estatísticas

Sobre um modelo gerado deve ser possível consultar a seguinte informação:

1. Utilizadores adicionados (número e listagem);
2. Utilizadores incluídos por um utilizador adicionado (número e listagem);
3. Utilizador com mais relacionamentos diretos;
4. Outro indicador que achar relevante;
5. Gráfico de barras com o número de relacionamentos dos 5 utilizadores com mais relacionamentos;
6. Outro gráfico que achar relevante.

## 5 Visualização do caminho mais curto

Deve ser possível visualizar no grafo o caminho mais curto entre dois utilizadores selecionados (adicionados ou incluídos), considerando apenas os relacionamentos diretos. As arestas e vértices que compõem o caminho devem ficar doutra cor, de forma a conseguir-se percecionar-se facilmente o caminho.

## 6 Logging

Deve ser mantido um log do processo de geração de um modelo. Este log é reiniciado a cada nova geração. Cada entrada do log deve ter o seguinte formato:

Ao adicionar um utilizador:
`<timestamp> | <id_utilizador_adicionado> | <número_de_relacionamentos_diretos> | <número_de_interesses>`

Ao adicionar um relacionamento direto (consequência de adição de utilizador):
`<timestamp> | <id_utilizador_adicionado> | <id_utilizador_existente> | <número_interesses_partilhados>`
ou
`<timestamp> | <id_utilizador_adicionado> | <id_utilizador_incluído> | <número_interesses_partilhados>`

Ao adicionar um interesse (consequência de adição de utilizador):
`<timestamp> | <id_utilizador_adicionado> | <id_interesse>`

Ao adicionar um relacionamento indireto:
`<timestamp> | <id_utilizador_adicionado> | <id_utilizador_incluído> | <id_interesse>`

## 7 Padrões de Software

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

## 8 Javadoc

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

## 9 Fonte de dados

São disponibilizados dois ficheiros com o enunciado com a informação necessária para representar a rede social:

- **relationships.csv**: contém uma matriz de utilizadores, onde a primeira coluna contém o identificador de um utilizador, e as colunas seguintes os identificadores dos seus relacionamentos diretos. Nenhum utilizador pode exceder 100 relacionamentos diretos;
- **interests.csv**: contém uma matriz que relaciona interesses com utilizadores, onde a primeira coluna contém o identificador de um interesse, e as colunas seguintes os identificadores dos utilizadores que partilham esse interesse. Nenhum interesse é partilhado por mais que 100 utilizadores;
- **user_names.csv:** a primeira coluna contém o identificador de um utilizador, e a segunda coluna o nome do utilizador;
- **interest_names.csv:** a primeira coluna contém o identificador de um interesse, e a segunda coluna o nome do interesse.

## III. Entrega e Discussões

O projeto obedece a uma **milestone** intermédia (15%) e a uma entrega **final** em duas fases (85%) - 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 (15% da nota total);
- **Se for entregue um projeto não-funcional (um projeto que não permita** a visualização de um modelo na forma de grafo, a partir de um identificador de utilizador a adicionar**), 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 (**09:00**), hora do servidor;
- As discussões são obrigatórias para todos os elementos do grupo de trabalho;
- O conteúdo das entregas correspondem aos respetivos projetos **IntelliJ IDEA** desenvolvidos, e respetivo relatório em PDF, contidos num ficheiro ZIP/RAR/7z, cujo nome obedeça ao seguinte formato: **<num1>__<num2>_<num3>.{zip, rar, 7z}**.

**Entrega Milestone via Moodle:** 30 de novembro (sábado) de 2020. Apresentações on-line a agendar posteriormente.

**Entrega Pré-final via Moodle:** 11 de janeiro (segunda-feira) de 2019. Apresentações a agendar posteriormente.

**Entrega Final via Moodle:** 29 de janeiro (quarta-feira) de 2019. Discussões a serem agendadas na semana seguinte.

Para as entregas existirão links de submissão por cada professor de laboratório.

## 1 Milestone

A milestone (MS) corresponde à entrega de:

Um projeto IntelliJ IDEA contendo:

- Implementação do TAD DiGraph – Fornecer uma implementação de Digraph<V,E> utilizando - **Lista de adjacências**- OU -**matriz de adjacências**
- Classe **SocialNetwork** que inclua uma instância da implementação anterior (e as classes que achar necessárias para representar os elementos que rotulam os vértices e as arestas) e que contenha métodos responsáveis por criar um modelo com relações diretas, a partir da leitura do ficheiro **relationships.csv.**

**Nota:** Sugere-se que o ficheiro seja lida para uma estrutura de dados auxiliar, e a partir daí construído o grafo.

- Criar uma pequena classe de Teste em JavaFX, onde é mostrado a visualização do grafo que corresponde ao modelo criado por leitura do ficheiro **relationships.csv.**
- Testes JUnit para a classe implementadas nos pontos 1. [Nota: para os testes poderão construir o vosso ficheiro "relationships.csv"]
- Ficheiro **README.md** com a descrição: estrutura de dados selecionadas, e algoritmos de carregamento de dados e construção do modelo.

## 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, não serão avaliados.**

### Versão Final

A versão final é composta por o (i) projeto IntelliJ IDEA e (ii) o Relatório.
O projeto IntelliJ IDEA corresponde à aplicação submetida anteriormente após a melhoria da estrutura do código após ter sido realizado refactoring ao mesmo.

**A adição de outras novas funcionalidades e/ou correção de problemas anteriores não serão contabilizadas.**

O relatório deverá conter:

1. A apresentação do(s) ADT(s) implementado(s) (descrição da interface Java e da implementação obtida);
2. Diagrama(s) de classes (diagramas ilegíveis serão contabilizados com 0 valores);
3. Documentação de todas as classes (sugere-se reutilizar a documentação Javadoc);
4. Descrição/uso/justificação dos padrões de software utilizados
    - deverá indicar para cada um dos padrões utilizados, a descrição do mesmo, qual a razão de esse ter sido selecionado e de que forma as classes do padrão foram mapeadas para a solução concreta;
5. 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 (15%)** | | Quiz sobre o enunciado (2 tentativas - com aprovação 90%) | 10 |
| | | Implementação do TAD DiGraph (DiGraphImpl) | 30 |
| | | Classe SocialNetwork – Construção do Modelo sem interesses | 50 |
| | | Visualização Inicial do Grafo | 20 |
| | | Testes Junit (Coverage 100% das classes Social Network e DiGraphImpl) | 50 |
| | | Mini-Relatório | 20 |
| | | Javadoc | 10 |
| | | Utilização da ferramenta GitHub através do GitHub classroom | 10 |
| | | **TOTAL** | **200** |
| **Entrega Final (85%)** | **Funcionalidades** | Modelo – Relações diretas | 20 |
| | | Modelo – Relações indiretas | 20 |
| | | Inspeção de Utilizadores e de Relações Indiretas | 10 |
| | | Undo (Utilizadores Adicionadas e Relações) | 10 |
| | | Logger | 10 |
| | | Persistência | 15 |
| | | Estatísticas | 10 |
| | | Caminho mais curto entre utilizadores | 10 |
| | | Interface Gráfica (Usabilidade geral, Vizualização do Modelo, Interação na construção do modelo) | 40 |
| | Javadoc | | 10 |
| | Utilização da ferramenta GitHub através do GitHub classroom | | 15 |
| | Relatório | | **30** |
| | | **TOTAL** | **200** |
| | Existência de bad smells (por cada não tratado) | | -5 (até -30) |
| | Não utilização de padrões de software adequados (por cada situação) | | -15 (até -60) |

(fim de enunciado)