

Trabalho de Laboratório – Curso EI

Objetivos:

Classes Abstratas e Interfaces.

Programa:

Criação de uma Rede Social para utilização em contexto académico. Esta rede social será utilizada por Alunos e Professores. Permite a publicação de Posts textuais ou partilha de eventos com a restante comunidade.

Regras de implementação:

- Criar a aplicação utilizando o IDE BlueJ.
- Implementar o código necessário e testar no fim de cada nível.
- Não é necessário obter dados do utilizador. Forneça os dados ao nível do código.
- Use as convenções de codificação adotadas para a linguagem Java (ver **Notas**).

Implementação:

Nível 1:

- Implemente uma classe abstrata **User** com o seguinte construtor e métodos:
 - **User(String username, String password)** – construtor que recebe e inicia os atributos **username** e **password** do utilizador. Este construtor inicializa também um atributo booleano **authenticated**, inicialmente com o valor **false**;
 - **getInfo()** – método abstrato que irá retornar a informação específica de determinado utilizador;
 - **String booleanToString(boolean value)** – método auxiliar que recebe um valor booleano e retorna uma String com o valor “Sim” ou “Não”, mediante o valor do atributo;
 - **getUsername, getPassword e isAuthenticated** – seletores dos atributos **username, password e authenticated**, respetivamente;
 - **setAuthenticated** – modificador do atributo **authenticated**.
- Implemente uma classe **Teacher** que herda da classe **User** e tem o seguinte construtor e métodos:
 - **Teacher(String username, String password)** – construtor que passa os valores de **username** e **password** ao construtor da classe pai e inicializa um **ArrayList<String>** que irá armazenar a designação das unidades curriculares lecionadas pelo respetivo docente;
 - **addCourseUnit(String courseUnit)** – método que recebe a designação de uma UC e a acrescenta à respetiva lista;
 - **getInfo ()** – método que retorna informação referente à designação do utilizador, **username**, estado de autenticação e respetivas UCs no caso de as ter atribuídas.
- Implemente uma classe **Student** que herda da classe **User** e tem o seguinte construtor e métodos:
 - **Student(String username, String password, String course)** – construtor que passa os valores de **username** e **password** ao construtor da classe pai e inicializa um atributo que contém o nome do curso do estudante;
 - **getInfo ()** – método que retorna informação referente à designação do utilizador, **username**, estado de autenticação e respetivo curso.
- Nas duas últimas classes reestruture o método **getInfo()** de forma a obter o seguinte output:

Professor: Username: ana.marques Autenticado: Não Professor de: - Programação Orientada por Objetos - Introdução à Programação	Estudante: Username: rita.martins Autenticado: Não Estudante do curso Licenciatura em Engenharia Informática
---	---

Trabalho de Laboratório – Curso EI

Nível 2:

- Implemente uma classe abstrata **Post** com os seguintes atributos:
 - **author** – atributo do tipo **User** que representa o autor do post;
 - **timestamp** – atributo que contém o valor temporal da data e hora da criação do post;
 - **visible** – atributo que identifica o estado do post
- Nesta mesma classe, implemente um construtor sem parâmetros **Post()**. Inicializa o autor com valor *null*, a visibilidade com o valor *true* por *default* e o *timestamp* da altura em que o *post* foi criado (sugestão: utilizar [System.currentTimeMillis\(\)](#)).
- Para apresentar a informação de determinado post, adicione o método abstrato **Show()**.
- Adicione os seletores/modificadores **getAuthor**, **setAuthor** e **isVisible** para os respetivos atributos **author** e **visible**.
- Implemente uma classe **MessagePost** que herda da classe **Post** e adiciona o atributo **message (String)**. O valor do atributo **message** é recebido no construtor.
- Implemente também uma classe **EventPost** que herda da classe **Post** e adiciona os atributos **description**, **date** e **location**. Adicione o construtor que recebe e inicia estes atributos.
- Implemente nas duas últimas classes o método **show()** que deverá apresentar o seguinte output:

```
*****
| Author: ana.marques
| Date: 11/04/20 12:31
|
| Event:
| Seminário de P00
|
| 23/03/19 10:30 | ESTSetúbal
*****
```

```
*****
| Author: rita.martins
| Date: 11/04/20 12:31
|
| Message:
| Brevemente cartaz da semana académica!
*****
```

Nível 3:

- Implemente uma classe **SocialNetwork** com os seguintes atributos:
 - **ArrayList<User> users** – lista de utilizadores da rede social
 - **ArrayList<Post> posts** – lista de publicações na rede social
 - **User authenticatedUser** – representa o utilizador autenticado.
- Implemente um construtor que não recebe atributos e inicializa cada um dos atributos. **authenticatedUser** deverá ser inicializado com o valor *null*.
- Implemente o método **addUser(User user)** que verifica se o utilizador consta na lista de utilizadores, comparando pelo valor de **username**. Se não existe, adiciona e informa o utilizador.
- Implemente o método **loginSession(String username, String password)**. Este método deve procurar na lista de utilizadores por um que tenha valor do **username** recebido. Se não encontrar, informa o utilizador. Ao encontrar, verifica se este se encontra autenticado, se não, verifica correspondência da password, informando se não coincidir. Em caso de sucesso, informa o utilizador, dando-lhe uma mensagem de boas-vindas.
- Implemente o método **logoutSession()** que, no caso de existir um *user* autenticado, altera o seu estado de autenticação e coloca **userAuthenticated** a *null*.
- Adicione o método **showHeader()** que apresenta um cabeçalho. Este método deve ser chamado em cada listagem apresentada.
- Adicione o método **listUsers()** que deve apresentar uma listagem de todos os utilizadores existentes, conforme *output*:

Trabalho de Laboratório – Curso EI

```
*****
*      ACADEMIC SOCIAL NETWORK      *
*****
*              USERS              *
*****

Estudante:
  Username: rita.martins
  Autenticado: Sim
  Estudante do curso Licenciatura em Engenharia Informática

Professor:
  Username: ana.marques
  Autenticado: Não
  Professor de:
    - Programação Orientada por Objetos
    - Introdução à Programação
```

Nível 4:

- Ainda na classe **SocialNetwork**, implemente o método **publishPost(Post post)**. Este método recebe um post e verifica se existe um *user* autenticado, se existir atribui a autoria do post a esse *user* e adiciona o post à lista de publicações. Informa o utilizador em caso de sucesso ou insucesso.
- Na mesma classe implemente o método **showFeed()** que, após apresentar o cabeçalho, mostra todos os posts da lista de publicações que estejam visíveis.

```
*****
*      ACADEMIC SOCIAL NETWORK      *
*****
*              FEED              *
*****

| *****
| Author: pedro.guerra
| Date: 11/04/20 15:13
|
| Message:
| Ola a todos!
| *****
| *****
|
| Author: ana.marques
| Date: 11/04/20 15:13
|
| Event:
| Seminário de P00
|
| 23/03/19 10:30 | ESTSetúbal
| *****
| Author: rita.martins
| Date: 11/04/20 15:13
|
| Message:
| Brevemente cartaz da semana académica!
| *****
| Author: pedro.guerra
| Date: 11/04/20 15:13
|
| Event:
| ENEI - Encontro Nacional de E.I.
|
| 23/02/20 08:30 | Braga
```

Trabalho de Laboratório – Curso EI

- Alguns tipos de posts, para além de aparecerem no *Feed*, poderão aparecer numa lista de notificações. Para tal, implemente a interface **Notifiable** que contém o método **showNotification()**. Na classe **EventPost** implemente a interface criada e o respetivo método. No output da notificação, deve constar apenas o nome do evento, data, hora e local.

Nível 5:

- Para se poder procurar por determinado conteúdo no feed principal, implemente a interface **Searchable**, que contém o método **boolean search(String content)**. Implemente esta interface nas classes **MessagePost** e **EventPost**:
 - Em **MessagePost** o conteúdo pode ser procurado tanto na mensagem como no nome do autor;
 - Em **EventPost**, pode ser procurado na descrição, localização ou nome do autor;
 - Em ambos, devolve *true* no caso de algum dos campos conter, *false* caso contrário.
- Na classe **SocialNetwork**, crie o método **searchFor(String text)**. Este método deve procurar em todos os posts que permitam esta procura pelo termo recebido (deve recorrer ao **processamento funcional**). Em caso de sucesso, informa o utilizador e apresenta o post onde foi encontrado. No caso do termo não ser encontrado, informa o utilizador.

```
The term 'acad' was found in:

| *****
| Author: pedro.guerra
| Date: 11/04/20 15:13
|
| Message:
| Brevemente cartaz da semana académica!
| *****
```

- Ainda nesta classe, crie o método **showNotifications()**. Este método deve apresentar o cabeçalho e, de seguida, todas as notificações existentes. Para filtrar as notificações existentes na lista de publicações, deve recorrer ao **processamento funcional**.

```
*****
*          ACADEMIC SOCIAL NETWORK          *
*****
*                      NOTIFICATIONS                      *
*****
| Evento: Seminário de P00
| Em: 23/03/19 10:30, ESTSetúbal
| *****
| Evento: ENEI - Encontro Nacional de E.I.
| Em: 23/02/20 08:30, Braga
| *****
```

Notas:

Para os identificadores siga as convenções adotadas normalmente, em particular:

- 1) A notação **camelCase** para o nome das variáveis locais e identificadores de atributos e métodos.
- 2) A notação **PascalCase** para os nomes das classes.
- 3) Não utilize o símbolo '_', nem abreviaturas nos identificadores.