Exercício Programa 2
Gerenciador de Salas para o Marcador de Reuniões

Prof. Dr. Daniel Cordeiro
Escola de Artes, Ciências e Humanidades
Universidade de São Paulo

Entrega: **24 de junho de 2019**

Introdução

Este é um programa[1] que será utilizado para auxiliar na marcação de uma reunião com vários participantes. Um dos participantes, responsável pela organização da reunião, indica em qual período ele gostaria de marcar a reunião (por exemplo, entre 10 e 20 de abril); ele determina também a lista de participantes identificados pelos seus endereços eletrônicos. A seguir, cada um dos participantes indica os horários de disponibilidade dentro do período determinado pelo organizador. O organizador então visualiza a sobreposição dos horários de todos os participantes e escolhe um horário para a reunião.

O trabalho deve ser feito preferencialmente em grupos de 2. Grupos individuais são aceitos (a contragosto ☹). Recomendo fortemente que os pares desenvolvam os programas usando programação pareada (os dois programadores sentados lado a lado compartilhando o mesmo computador).

Fase 1: disponibilidade de horários

Para a Fase 1, que deve ser entregue até o final de abril, a implementação deve se basear em uma interface de texto somente, ou seja, a visualização dos horários dos participantes será feita em modo texto utilizando-se, por exemplo, o objeto `Console`.

A definição dos participantes da reunião será feita utilizando-se do seguinte método que deverá ser implementado na classe `MarcadorDeReuniao`:

```
public void marcarReuniaoEntre(LocalDate dataInicial,
                               LocalDate dataFinal,
                               Collection<String> listaDeParticipantes)
```

[1] Adaptado de um EP de MAC0441 gentilmente cedido pelo prof. Fábio Kon (IME).

As datas devem ser do tipo `java.time.LocalDate` e a `listaDeParticipantes` deve ser uma `Collection` de Strings que identificam os participantes.

Cada participante define os seus horários disponíveis através do seguinte método:

```
public void indicaDisponibilidadeDe(String participante,
                                    LocalDateTime inicio,
                                    LocalDateTime fim)
```

onde cada participante é identificado com uma String e o início e fim da disponibilidade é indicado com dias e horários dados por instâncias de `java.time.LocalDateTime`.

Finalmente, você deve implementar o método

```
public void mostraSobreposicao()
```

que, provavelmente, vai dar o maior trabalho e vai exigir mais criatividade de vocês para informar os dados de uma forma clara e elegante. Ele deve exibir um relatório com as escolhas realizadas e indicar em quais horários todos os participantes poderiam participar da reunião.

Fase 2: Reserva de Salas

Na fase 2, que deverá ser entregue até o dia da segunda prova, em 24 de junho, faremos uma classe cujo objetivo é reservar as salas para as nossas reuniões. A classe `GerenciadorDeSalas` deve implementar, pelo menos, os seguintes métodos:

- `adicionaSalaChamada`, que deve receber o nome da sala, a capacidade máxima da sala, e uma descrição;
- `removeSalaChamada`, que deve receber o nome da sala a ser removida;
- `listaDeSalas`, que deve devolver uma instância de `List` com objetos do tipo `Sala`;
- `adicionaSala`, que deve receber uma instância de `Sala`;
- `reservaSalaChamada`, que recebe um nome de sala, um `LocalDateTime` que indica o início da reserva e um outro `LocalDateTime` para indicar o final da reserva. O método deve devolver uma instância de `Reserva`;
- `cancelaReserva`, que recebe um objeto devolvido pelo método `reservaSalaChamada` e cancela esta reserva;
- `reservasParaSala`, que devolve uma `Collection` de objetos `Reserva` que representam as reservas da respectiva sala.

- `imprimeReservasDaSala`, que recebe uma instância de `Sala` e imprime todas as suas reservas.

  Objetos do tipo `Sala` possuem métodos de acesso para os seguintes atributos: `nome`, `local`, `capacidade` e `observacoes`; `capacidade` é um inteiro, os demais atributos são `String`s.

  Se a reserva for efetuada com sucesso, o método `reservaSalaChamada` devolve uma instância do objeto `Reserva` (com métodos `sala`, `inicio`, e `fim`, que devolvem, respectivamente, uma instância de `sala` e dois `LocalDateTime`, do início e do fim da reserva). Se a reserva falhar por qualquer motivo (p.ex. sala inexistente ou sala já reservada) o gerenciador deve obrigatoriamente lançar uma exceção e opcionalmente imprimir uma mensagem de erro.

  Não há necessidade de implementar uma interface gráfica para o programa. Os exemplos de uso das classes que devem estar presentes no relatório (veja abaixo) são suficientes.

Entrega

Um arquivo compactado no formato `.tar.xz` ou `.zip` deve ser entregue no eDisciplinas contendo o código-fonte produzido, uma versão compilada (arquivos `.class`) do programa e um relatório (nos formatos abertos ODT[2] ou PDF). Seu relatório deve descrever em detalhes a implementação proposta e deve conter, obrigatoriamente:

1. o nome e número USP de todos os integrantes do grupo;
2. instruções sobre como compilar e executar seu programa;
3. exemplos de utilização do sistema e suas respectivas saídas;
4. um diagrama de classes UML com a descrição das classes e relacionamentos implementadas no projeto;
5. uma lista com os Padrões de Projeto que foram aplicados no desenvolvimento do projeto e como eles foram aplicados no seu EP.

O seu EP será recompilado e testado em um computador com o seguinte software instalado:

```
$ lsb_release -a
No LSB modules are available.
Distributor ID: Debian
Description: Debian GNU/Linux buster/sid
Release: testing
Codename: buster

$ java -version
```

[2] OpenDocument Format: `https://www.libreoffice.org/discover/what-is-opendocument/`.

```
openjdk version ”12.0.1” 2019-04-16
OpenJDK Runtime Environment (build 12.0.1+12-Debian-2)
OpenJDK 64-Bit Server VM (build 12.0.1+12-Debian-2, mixed mode)
```

Dúvidas, problemas, etc.

Quais dúvidas ou problemas relacionados ao EP devem ser discutidos no fórum de Discussões no eDisciplinas: `https://edisciplinas.usp.br/mod/forum/view.php?id=2534318`.