Universidade Federal do Rio Grande do Sul
Departamento de Informática Aplicada

Programação Orientada a Objeto

22. Tratamento de Erros e Exceções (Prática)

Prof. Dr. Thiago L. T. da Silveira
tltsilveira@inf.ufrgs.br

2° Semestre de 2020

Objetivos

- Exercitar os conceitos de POO vistos na aula anterior!
  - Escrevendo, compilando e interpretando (executando) aplicações em Java;
  - Identificação e Tratamento de Exceções.

Exercício #1 Parte #0

- Vamos reformular a implementação de um sistema de controle de abastecimento de uma *cafeteria*. Uma **`Cafeteira`** industrial permite preparar *Americanos*, *Expressos* e *Lattes* e tem controle sobre níveis de café, água e leite (matérias primas).
- O sistema da cafeteria (**`Aplicacao`**) ainda gerencia os **`Estoque`**s das matérias primas e trata potenciais exceções!
- Há exceções dos tipos **`NivelMuitoBaixo*Exception`**, **`NivelMuitoAlto*Exception`** e **`FaltaEstoqueException`** (o comportamento básico das classes é provido!).

Imagens livres de https://pixabay.com/

Exercício #1 Parte #1

- Faça o *download* do arquivo **codigos.zip** - que contém **Aplicacao.java**, **Cafeteria.java** e **Estoque.java** e uma série de classes do tipo exceção;
- As exceções que agem sob `Cafeteira` são descritas em **CafeteriaException.java** e subclasses (**NivelMuitoBaixoAguaException.java**, **NivelMuitoBaixoCafeException.java**, **NivelMuitoBaixoLeiteException.java**, **NivelMuitoAltoAguaException.java**, **NivelMuitoAltoCafeException.java** e **NivelMuitoAltoLeiteException.java**);
- A exceção que age sob `Estoque` é descrita em **FaltaEstoqueException.java**;

Exercício #1 Parte #2

- Analise o funcionamento das classes que descrevem as exceções da nossa aplicação;
- Veja que algumas exceções têm construtor que não recebe nada (`void`) e outras têm construtor que recebe `String`;
    - Todas as exceções chamam o construtor da classe mãe com `super(...)`;

Exercício #1 Parte #2

- `Estoque`s têm uma `quantidade` (`double`) e uma `descricao` (`String`) - os quais devem ser informados no construtor;
  - `quantidade` representa uma quantidade de matéria prima em gramas ou mililitros;
- `Estoque`s têm um método `reporEstoque` que recebe uma `quantidade` (`double`) e a acumula em `this.quantidade`;
- `Estoque`s têm um *getter* para `descricao`;
- `Estoque`s têm um método para `get100MililitrosOUGramas` que retira (se possível) de `quantidade` e retorna 100 mililitros ou gramas (`double`);
  - Esse método pode lançar uma exceção do tipo `FaltaEstoqueException`;

Exercício #1 Parte #2

- `Cafeteira` tem `nivelCafe` (`double`), `nivelAgua` (`double`) e `nivelLeite` (`double`);
    - Os atributos são inicializados com valor zero no construtor (que não recebe nada);
- `Cafeteira` sobrescreve o método `toString` e retorna (por exemplo) "`Café: 0.0 | Água: 0.0 | Leite: 0.0`" na inicialização (isto é, quando `nivelCafe` = 0, `nivelAgua` = 0 e `nivelLeite` = 0;
    - A `String` retornada deve mostrar o status atual da `Cafeteira`;

Exercício #1 Parte #2

- **`Cafeteira`** tem um método **`pedir`** que recebe um tipo de **`cafe`** (**`String`**) e subtrai (se possível) quantidades de **`nivelCafe`**, **`nivelAgua`** e **`nivelLeite`**;
    - Esse método pode levantar exceções dos tipos `NivelMuitoBaixoCafeException`, `NivelMuitoBaixoAguaException` e `NivelMuitoBaixoLeiteException`;

cafe	nivelCafe	nivelAgua	nivelLeite
“Latte”	- 30	-40	-130
“Expresso”	-30	-70	-0
“Americano”	-40	-160	-0

Exercício #1 Parte #2

- `Cafeteira` ainda tem o método `completarNivelAgua` que recebe uma `quantidade` (`double`) de água;
  - Esse método adiciona `quantidade` a `nivelAgua` se `nivelAgua` for menor que 900 mililitros;
  - Esse método pode lançar exceções do tipo `NivelMuitoAltoAguaException`;
- `Cafeteira` ainda tem o método `completarNivelLeite` que recebe uma `quantidade` (`double`) de leite;
  - Esse método adiciona `quantidade` a `nivelLeite` se `nivelLeite` for menor que 400 mililitros;
  - Esse método pode lançar exceções do tipo `NivelMuitoAltoLeiteException`;
- `Cafeteira` ainda tem o método `completarNivelCafe` que recebe uma `quantidade` (`double`) de café;
  - Esse método adiciona `quantidade` a `nivelCafe` se `nivelCafe` for menor que 400 gramas;
  - Esse método pode lançar exceções do tipo `NivelMuitoAltoCafeException`;

Exercício #1 Parte #2

- A `Aplicacao` tem dois métodos de classe: `main` e `completaNivel`;
- O método `completarNivel` recebe uma `cafeteira` (`Cafeteira`) e um `estoque` (`Estoque`);
  - Esse método deve usar os métodos `get100MililitrosOUGramas` e `completarNivel*` a depender da `descricao` de `estoque` (se de “Água”, “Leite” ou “Café”);
  - Esse método deve tirar 100 mililitros/gramas de `estoque` e colocar em `cafeteira`;
  - Esse método deve fazer isso ***até que*** ocorra uma `CafeteiraException` (exceção capturada com polimorfismo) ou `FaltaEstoqueException` (mostrar mensagem nesse último caso);
    - Para mostrar a mensagem de uma exceção ver slide 25 da Aula 21;
  - O método deve imprimir `cafeteria` antes e depois do preenchimento do nível da `cafeteria`;

Exercício #1 Parte #2

- O método **`main`** deve instanciar uma **`cafeteira`** (**`Cafeteira`**) e três **`Estoque`**s: **`cafe`**, **`agua`** e **`leite`**;
  - Deve-se identificar os estoques corretamente (lembrar de **`descricao`**);
  - Há 5 quilogramas de café, 100 litros de água e 20 litros de leite em estoque;
- O método **`main`** deve usar **`completarNivel`** para encher os reservatórios de **`cafe`**, **`agua`** e **`leite`** da **`cafeteira`**;
- O método **`main`** deve atender a 12 pedidos (usar **`pedir`**) e tratar exceções!
  - Os pedidos são (nessa sequência): Latte, Americano, Latte, Americano, Americano, Expresso, Americano, Americano, Americano, Latte, Americano e Latte.
- O método **`main`** deve, por fim, imprimir **`cafeteira`**;

Exercício #1 Parte #3

- A implementação dada não declara, não lança e não trata exceções;
- Você deve implementar as declarações, lançamento e tratamento das exceções nas classes `Cafeteira`, `Estoque` e `Aplicacao`!

Exercício #1 - Exemplo de Execução

- A execução de **`Aplicacao`**, conforme descrita, deve imprimir na tela:

```
Café: 0.0 | Água: 0.0 | Leite: 0.0
Café: 0.0 | Água: 0.0 | Leite: 400.0
Café: 0.0 | Água: 0.0 | Leite: 400.0
Café: 0.0 | Água: 900.0 | Leite: 400.0
Café: 0.0 | Água: 900.0 | Leite: 400.0
Café: 400.0 | Água: 900.0 | Leite: 400.0
Nível de água muito baixo!
Café: 150.0 | Água: 110.0 | Leite: 140.0
Café: 150.0 | Água: 910.0 | Leite: 140.0
Nível de café muito baixo!
Café: 0.0 | Água: 390.0 | Leite: 10.0
Café: 400.0 | Água: 390.0 | Leite: 10.0
Nível de leite muito baixo!
Café: 400.0 | Água: 390.0 | Leite: 10.0
Café: 400.0 | Água: 390.0 | Leite: 410.0
Café: 370.0 | Água: 350.0 | Leite: 280.0
```

Atividades

- Entrega de um arquivo **.zip** (contendo a estrutura de diretórios da aplicação e os arquivos .java)
    - Entregue a reimplementação descrita no Exercício #1 Parte #3
    - Entrega até às 23:55h de 15/04/2021

Universidade Federal do Rio Grande do Sul
Departamento de Informática Aplicada

Programação Orientada a Objeto

22. Tratamento de Erros e Exceções (Prática)

Prof. Dr. Thiago L. T. da Silveira
tltsilveira@inf.ufrgs.br

2° Semestre de 2020