ESCOLA POLITÉCNICA DA UNIVERSIDADE DE SÃO PAULO
Departamento de Engenharia de Computação e Sistemas Digitais

PCS 3111 - Laboratório de Programação Orientada a Objetos para Engenharia Elétrica

Aula 09 – Programação defensiva

Atenção

1. Código inicial para resolução dos exercícios encontra-se disponível no e-Disciplinas.
2. Para a resolução dos exercícios, adicione atributos privados às classes conforme necessário, desde que se mantenha as assinaturas e funcionamento especificados no enunciado.

Exercício 01
Altere o código fornecido da classe **Atividade** seguindo as seguintes características:

1. O construtor deve:
   - Gerar uma exceção do tipo `invalid_argument` (da biblioteca padrão) caso as horas necessárias ou o valor de `maximoPessoas` sejam menores ou iguais a zero.
2. O método `adicionar` deve:
   - Não ter valor de retorno (ser `void`).
   - Gerar uma exceção do tipo `invalid_argument` (da biblioteca padrão) caso a **Pessoa** passada como argumento seja inválida, isto é, NULL.
   - Gerar uma exceção do tipo `overflow_error` (da biblioteca padrão) caso o vetor `pessoas` esteja cheio.
3. O método `getDuracao` deve:
   - Gerar uma exceção do tipo `logic_error` (da biblioteca padrão) caso não existam **Pessoas** adicionadas na **Atividade**.

Atenção: Não crie outros métodos públicos além dos fornecidos. Teste os possíveis casos de erro na main utilizando os comandos try-catch.

Exercício 02
Crie e implemente a classe `ErroRecursosRepetidos` de modo que:

- Seja uma classe filha de `invalid_argument` (da biblioteca padrão).
- A classe contenha apenas seu destrutor e construtor, o qual recebe como argumento uma mensagem (string), como exposto em aula.

Modifique a classe **Atividade** fornecida para que o método `adicionar` gere um erro do tipo `ErroRecursosRepetidos` caso tente-se adicionar uma **Pessoa** previamente adicionada – isto é, o mesmo objeto.

Exercício 03

Altere a classe **Projeto**, utilizando comandos try-catch dentro desta classe, de forma que o método `getDuracao` retorne 0 caso uma de suas **Atividades** não possua **Pessoas** adicionadas. Caso contrário, a duração de um **Projeto** equivale a:

$$duração = \sum duração\ de\ cada\ Atividade$$

Por exemplo, caso o projeto tenha 3 Atividades adicionadas (a1, a2 e a3) e uma delas (a1) jogue `logic_error` ao chamar `getDuracao`, o método deve retornar 0 – mesmo que a2 e a3 tenham valores de duração.

Testes do Judge

Exercício 1

- Atividade: construtor, adicionar e getDuracao com valores válidos
- Atividade: horas necessárias e maximoPessoas inválidos
- Atividade: adicionar com Pessoa inválida
- Atividade: adicionar com Pessoa válida e vetor cheio
- Atividade: getDuracao sem Pessoas

Exercício 2

- ErroRecursosRepetidos é filha de invalid_argument
- Atividade: adicionar Pessoa repetida
- Atividade: adicionar Pessoa não repetida

Exercício 3

- Projeto: getDuracao com uma Atividade sem Pessoas
- Projeto: getDuracao com várias Atividades sem Pessoas
- Projeto: getDuracao com Pessoas