Ministério da Educação
Universidade Federal de Itajubá – *Itajubá*

COM221 – Computação Orientada a Objetos II

Exercícios – Aula 9 - Reflexão
Prof.: Phyllipe Lima

Instruções Gerais:

- Atividade Individual.
- Entregar a solução completa em formato ZIP pela tarefa do SIGAA.
- Data entrega: 08/11 as 22:40
- Versão mínima com Java 11

Tarefa – Implementando um sistema de criação de instâncias usando Reflexão

Utilizando a Reflexão podemos descobrir, em tempo de execução, quais os construtores que uma classe possui e com isso invocá-los para criarmos as instâncias. O código que irá criar essas instâncias não conhece nenhuma informação sobre suas classes, e precisar receber um objeto do tipo "***Class***" para conseguir invocar essas operações.

Podemos também recuperar os construtores parametrizados, basta passar duas informações. A primeira é um vetor contendo os "tipos" que estamos buscando, em outras palavras, esse vetor será do tipo Class. Também precisamos informar quais os parâmetros que serão usados, isto é, um vetor do tipo "***Object***".

Tendo acesso uma instância de "***Class***", conseguimos invocar a seguinte operação.

```
clazz.getConstructor(parameTypes).newInstance(parametros);
```

Onde `getConstructor(parameTypes)` irá buscar o construtor, parametrizado, contendo os tipos no vetor `parameTypes`. Em seguida, com o construtor recuperado, chamamos o método `newInstance(parametros)` passando o vetor com os parâmetros para se criar a instância. Observe que são duas informações distintas, a primeira é o "tipo" dos parâmetros, e outra são os parâmetros propriamente ditos. Para buscar o construtor vazio, se existir, basta não passar parâmetros durante a busca do construtor.

Para obter os tipos dos parâmetros basta invocar o método "getClass" em qualquer parâmetro. Observe o exemplo.

```
Object num = 10;
System.out.println(num.getClass());
```

A saída no terminal será "java.lang.Integer". Isso significa que a variável "num" é do tipo "Integer".

Nesse caso teremos um problema pois o método que irá buscar o construtor (getConstructor) não reconhece esse tipo, apenas o primitivo "int.class". Isso ocorre para todos os tipos primitivos: double, float, char, void e int.

Assim, antes de buscar o construtor precisamos converter os tipos do parâmetro recebido. Se for encontrado qualquer tipo "Wrapper" devemos converter para os respectivos primitivos seguindo o exemplo abaixo.

```
private static Class<?> getTipo(Object parametro) {
    if(parametro.getClass() == Integer.class)
        return int.class;
    else if(parametro.getClass() == Double.class)
        return double.class;
    return parametro.getClass();
}
```

Nesse exemplo foi considerado apenas a conversão de int e double, idealmente o método faria a conversão de todos.

Sua tarefa é criar uma classe `CriadorInstancia` que possui dois métodos estáticos com sobrecarga. O primeiro recebe uma instancia do tipo Class<?> e retorna uma instância concreta utilizando a reflexão para buscar o construtor vazio e devolver a instância. A sua sobrecarga recebe um segundo parâmetro que é um vetor com os parâmetros, na ordem adequada. Esse vetor deverá ser do tipo “Object”.

Assinatura método 1:

```
public static Object criaInstancia(Class<?> clazz)
```

Assinatura método 2:

```
public static Object criaInstancia(Class<?> clazz, Object[] parametros)
```

Para complementar, será fornecido também um arquivo com 4 (quatro) testes de unidade. A tarefa será considerada encerrada quando todos os testes passarem. Os testes lidam com a criação de instâncias do tipo Puzzle e BoardGame.