Programação Orientada a Objetos

Bacharelado em Ciência da Computação

Prof. Dr. Eduardo Takeo Ueda

11ª Lista de Exercícios

(Entregar apenas o indicado com (1.0 ponto))

1. Implemente uma classe chamada ConverteTemperatura com dois métodos: converteCelsiusParaFahrenheit e converteFahrenheitParaCelsius. Em seguida utilize JUnit para realizar testes unitários na classe ConverteTemperatura.

2. Considere uma caderneta de cromos. Os cromos têm um número e uma imagem. Não é necessário modelar a imagem, considere a imagem como sendo uma instância da seguinte classe:

   ```
   class Image { /* conteúdo omitido */ }
   ```

   A caderneta guarda os cromos pela ordem de numeração e não permite guardar cromos repetidos. É possível adicionar cromos a uma caderneta (método add) e é possível remover um cromo se for indicado o seu número (método remove). Duas cadernetas dizem-se iguais (equals) se tiverem o mesmo número de cromos (independentemente das características dos cromos individuais). É possível obter uma lista ordenada (por número) contendo os cromos de uma caderneta (método getAll).

   (a) Implemente as classes dos cromos (Card) e da caderneta (Album) (pode utilizar classes do pacote de coleções do Java).

   (b) Implemente uma classe de teste (com JUnit) que contenha um teste: (*i*) que verifique se a inserção de um cromo na caderneta funciona; (*ii*) e outro que verifique as propriedades associadas ao método equals.

3. Considere a classe AndGate, definida como indicado no código da figura a seguir:

```java
public class AndGate {
    private boolean _a = false;
    private boolean _b = false;

    public AndGate()                    { /* empty */ }
    public AndGate(boolean v)           { _a = _b = v; }
    public AndGate(boolean a, boolean b) { _a = a; _b = b; }

    public boolean getA()           { return _a; }
    public void    setA(boolean a)  { _a = a; }
    public boolean getB()           { return _b; }
    public void    setB(boolean b)  { _b = b; }

    public boolean getOutput()      { return _a && _b; }

    public boolean equals(Object other) { /* ... */ }
}
```

   Implemente uma classe de teste (com JUnit) que verifique o correto funcionamento da porta AND, tal como implementada acima. Teste, em particular, o funcionamento da operação lógica (definição de entradas e obtenção de saída) e o do método de comparação (equals).

4. Considere a classe Fracao, definida como indicado no código da figura a seguir. Implemente uma classe teste (com JUnit) para testar o construtor da classe Fracao quando a exceção é lançada.

```java
public class Fracao {

    private int numerador;
    private int denominador;

    public Fracao(int n, int d) {

        if (d == 0) {
           throw new IllegalArgumentException(
           "Denominador não pode ser igual a zero");
        }
        this.numerador = n;
        this.denominador = d;
    }
}
```

5. (1.0 ponto) Considere a classe Calculadora, definida como indicado no código da figura a seguir.

```java
public class Calculadora {

    private double n1, n2;

    public Calculadora(double n1, double n2) {
        this.n1 = n1;
        this.n2 = n2;
    }

    public double getSoma() {
        return n1 + n2;
    }
    public double getSubtracao() {
        return n1 - n2;
    }
    public double getMultiplicacao() {
        return n1 * n2;
    }
     public double getDivisao() {
         if (n2 == 0) {
            throw new ArithmeticException();
         }
            return n1 / n2;
    }

}
```

Implemente uma classe teste (com JUnit) para testar todos os métodos da classe Calculadora. Teste, por exemplo, o caso em que 0 é somado com 0. Note que para a operação de divisão existe a possibilidade de lançamento de uma exceção.