Roteiro Aula Prática 4 - 23/03/2017
Programação de Computadores 2 - Programação Orientada a Objetos
Prof. Daniel Hasan Dalip – hasan@decom.cefetmg.br

Crie um projeto para o exercício. Crie um pacote por exercício, cada pacote chamará exercício**Num** em que **Num** é o número do exercício. Após finalizado os exercícios, salve o projeto em um zip e envie-o pelo Moodle (*Entrega Roteiro Aula Prática 4*). Não será aceito o recebimento via e-mail. Plágio não será tolerado e tanto a pessoa que forneceu quanto a que utilizou perderão os pontos da prática. **Entrega até: 29/03/2017**

Exercício 1: Implemente as classes ContaCorrente, SaldoInsuficienteException e ValorNegativoException apresentadas na aula teórica.

(a) Crie uma classe principal para testar a ContaCorrete. Perceba que, obrigatóriamente, você deverá tratar as exceções lançadas. Para isso, crie uma classe principal que use o try, catch usando o seguinte esqueleto:

```
try{
      (...)
}
catch(SaldoInsuficienteException se){
        System.out.println("Saldo insuficiente :-(");
}
catch(BancoException be){
        System.out.println("Erro: Tente novamente. "+be.getMessage());
}
finally{
       System.out.println("Obrigado por usar nosso serviço!");
}
```

Em "(...)" crie uma conta corrente e teste com valores para que seja lançado estas exceções e alguns que não. Entenda o porquê da exibição de cada mensagem.

(b) Agora, altere as classes SaldoInsuficienteException e ValorNegativoException para que elas sejam RunTimeException. Para isso, altere nas classes SaldoInsuficienteException e ValorNegativoException de "extends Exception" para "extends RunTimeException" . Por exemplo:

```
public class SaldoInsuficienteException extends Exception{
        public SaldoInsuficienteException(String msg){
                super(msg);
        }
}
```

Agora, você não será obrigado especificar e tratar tais exceções. Crie uma nova classe principal e faça os testes sem usar try catch. Além disso, você pode eliminar a claúsula "throws" do método sacar.

Exercício 2: Um médico possui nome (String), idade (int), CRM (String) e especialidade. A especialidade é composta por uma string e um código.
a) Implemente o construtor e getters, setters tanto de Medico tanto de Especialidade.

b) Implemente o toString da especialidade. A especialidade deve ser exibida da seguinte forma: “Especialidade: Cardiologista (1293)” onde 1293 é o código da especialidade.

c) Na classe Médico método toString deve apresentar os dados da seguinte forma:
“Nome: Alice Fernandes Idade: 30 anos
Especialidade Cardiologista (1293)”
Use o toString da especialidade para exibí-la. Dica: ao concaternarmos uma string com um objeto, é executado o toString deste objeto. Por exemplo, suponha que objMedico1 seja um objeto da classe Medico.
Caso seja feito: “Medico: “+objMedico1. Será produzida a seguinte string:
“Medico: Nome: Alice Fernandes Idade: 30 anos
Especialidade Cardiologista (1293)”

d) A classe IllegalArgumentException é uma exceção do tipo RunTimeException que pode ser usada quando o parâmetro de um método ou construtor está incorreto. Utilize-a para garantir que a idade seja sempre acima de 18 anos nos objetos criados. Ou seja, utilize apenas nos métodos que haverá modificação desta idade.

e) Crie uma classe principal para testar o funcionamento do código e o lançamento da exceção.