Exercício Exceções – Programação Orientada a Objetos

1 - Na classe Conta, modifique o método deposita(double x): Ele deve lançar uma exception chamada IllegalArgumentException, que já faz parte da biblioteca do java, sempre que o valor passado como argumento for inválido (por exemplo, quando for negativo).

```
void deposita(double valor) {
    if (valor < 0) {
        throw new IllegalArgumentException();
    } else {
        this.saldo += valor;
    }
}
```

2 - Crie uma classe TestaDeposita com o método main. Crie uma ContaPoupanca e tente depositar valores inválidos:

```
public static void main(String[] args) {
    Conta cp = new ContaPoupanca();
    cp.deposita(-100);
}
```

O que acontece? Uma IllegalArgumentException é lançada uma vez que tentamos depositar um valor inválido. Adicione o try/catch para tratar o erro:

```
public static void main(String[] args) {
    Conta cp = new ContaPoupanca();

    try {
        cp.deposita(-100);
    } catch (IllegalArgumentException e) {
        System.out.println("Você tentou depositar um valor inválido");
    }
}
```

Atenção: se a sua classe ContaCorrente está reescrevendo o método deposita e não utiliza do super.deposita, ela não lançará a exception no caso do valor negativo! Você pode resolver isso utilizando o super.deposita, ou fazendo apenas o teste com ContaPoupanca.

3 - Ao lançar a IllegalArgumentException, passe via construtor uma mensagem a ser exibida. Lembre que a String recebida como parâmetro é acessível depois via o método getMessage() herdado por todas as Exceptions.

```
void deposita(double valor) {
    if (valor < 0) {
        throw new IllegalArgumentException("Você tentou depositar" +
                                           " um valor negativo");
    } else {
        this.saldo += valor  - 0.10;
    }
}
```

4 - Altere sua classe TestaDeposita para exibir a mensagem da exceção através da chamada do getMessage():

```
public static void main(String[] args) {
    Conta cp = new ContaPoupanca();

    try {
        cp.deposita(-100);
    } catch (IllegalArgumentException e) {
        System.out.println(e.getMessage());
    }
}
```

5 - Crie sua própria Exception, ValorInvalidoException. Para isso, você precisa criar uma classe com esse nome que estenda de RuntimeException.

```
class ValorInvalidoException extends RuntimeException {

}
```

Lance-a em vez de IllegalArgumentException.

Atenção: nem sempre é interessante criarmos um novo tipo de exception! Depende do caso. Neste aqui, seria melhor ainda utilizarmos IllegalArgumentException. A boa prática diz que devemos preferir usar as já existentes do Java sempre que possível.

6 - Coloque um construtor na classe ValorInvalidoException que receba valor inválido que ele tentou passar (isto é, ele vai receber um double valor).

Quando estendemos uma classe, não herdamos seus construtores, mas podemos acessá-los através da palavra chave super de dentro de um construtor. As exceções do Java possuem uma série de construtores úteis para poder populá-las já com uma mensagem de erro. Então vamos criar um construtor em ValorInvalidoException que delegue para o construtor de sua mãe. Essa vai guardar essa mensagem para poder mostrá-la ao ser invocado o método getMessage:

```
class ValorInvalidoException extends RuntimeException {

    ValorInvalidoException(double valor) {
        super("Valor invalido: " + valor);
    }

}
```

Dessa maneira, na hora de dar o throw new ValorInvalidoException você vai precisar passar esse valor como argumento:

```
if (valor < 0) {
    throw new ValorInvalidoException(valor);
}
```

7 - Declare a classe ValorInvalidoException como filha direta de Exception em vez de RuntimeException. Ela passa a ser checked. O que isso resulta?

Você vai precisar avisar que o seu método deposita() throws ValorInvalidoException, pois ela é uma checked exception. Além disso, quem chama esse método vai precisar tomar uma decisão entre try-catch ou throws. Depois, retorne a exception para unchecked, isto é, para ser filha de RuntimeException, pois iremos utilizá-la assim em exercícios dos capítulos posteriores.

Atividade Extra:

Imagine que a sua aplicação é composta pelo seguinte código:

```
Object o = null;
o.toString();
```

Se você executar este código irá perceber que uma exceção será lançada. Identifique que exceção é esta e altere este mesmo código para que ele exiba uma mensagem amigável de erro e termine normalmente.