



# Nona lista de exercícios de Programação Orientada a Objetos

## Questões teóricas

1. Em POO, o que são exceções? Qual o principal motivo para se usar exceções em um programa?
2. Como no Java é possível criar uma exceção personalizada?
3. Qual a diferença entre **lançar** uma exceção e **capturar** uma exceção?
4. Um método pode capturar uma exceção lançada por ele mesmo?
5. Vimos em aula que toda exceção é, essencialmente, **uma classe**. Uma exceção poderia então ser **estática**? Explique.
6. Qual a diferença entre exceções *checked* e exceções *unchecked*?

## Questões práticas

7. Copie as duas classes a seguir:

```
public class Exercicio {
    public static void metodo1() {
        try {
            System.out.println("\tinicio do metodo1");
            Exercicio.metodo2();
            System.out.println("\tfim do metodo1");
        } catch(IndexOutOfBoundsException e) {
            System.out.println("\tExceção \"índice fora do limite\" capturada");
        }
    }
    public static void metodo2() {
        try {
            System.out.println("\t\tinicio do metodo2");
            Exercicio.metodo3();
            System.out.println("\t\tfim do metodo2");
        } catch(NullPointerException e) {
```



```
        System.out.println("\t\tExceção ponteiro nulo capturada");
    }
}

public static void metodo3() {
    try {
        System.out.println("\t\t\tinicio do metodo3");
        // comando(s) a ser(em) descrito(s)
        System.out.println("\t\t\tfim do metodo3");
    } catch(ArithmeticException e) {
        System.out.println("\t\t\tExceção erro aritmético capturada");
    }
}

}

public class Driver {
    public static int lista[] = new int[3];
    public static void main(String[] args) {
        try {
            System.out.println("inicio da main");
            Exercicio.metodo1();
            System.out.println("fim da main");
        } catch(Exception e) {
            System.out.println("Exceção genérica capturada");
        }
    }
}
```

- Apresente a saída do programa da forma que está, só com comentário.
- Apresente a saída do programa quando o comando no metodo3 for:  
`lista[0] = lista[4];`
- Apresente a saída do programa quando os comandos do metodo3 forem:  
`String s = null;`  
`char c = s.charAt(0);`
- Apresente a saída do programa quando o comando no metodo3 for:  
`int div = 2/0;`



8. Crie um programa com a classe Funcionario, detalhada abaixo (note que é o mesmo diagrama da Lista 3):

Funcionario
- nome : String - cargo : String - matricula : int - idade : int - chefe : Funcionario - salário : double
+mesmaChefia(f : Funcionario) : boolean

- a) Crie **todos** os getters e setters dos atributos privados.
- b) Assim como fizemos na lista 3, Crie um método construtor, o qual deve receber todos os atributos como parâmetros de entrada e então associá-los aos campos respectivos do objeto em construção. Para cada funcionário que não possui chefe, utilizar o valor **null** em sua criação. Por exemplo, se a construção de um novo objeto Funcionario precisasse apenas do nome do Funcionario e também da referência para o chefe, *mas ainda não existe um chefe*, um exemplo de construção seria

***Funcionario bill = new Funcionario("Bill Gates", null);***

- c) Modifique o método setChefe() para o código abaixo (copie-o):

```
public void setChefe(Funcionario chefe) {  
    if(chefe == this)  
        throw new RuntimeException();  
    this.chefe = chefe;  
}
```

Agora, na main, crie um cenário onde a exceção acima é lançada. **Apenas force a ocorrência do erro, sem capturá-lo com um bloco try/catch.** O código roda normalmente? Você saberia explicar o que ocorreu?

- d) Crie, na main, um bloco try/catch que envolve o método do item "c". Na captura (bloco **catch**), imprima a mensagem "Chefe inválido". O código compila e roda normalmente dessa vez? Por que?



- e) De volta ao item “c”, mude o lançamento: substitua

```
throw new RuntimeException();
```

por

```
throw new Exception();
```

Tente rodar o código. Por que ele sequer compila? **Corrija o erro para que a nova exceção seja suportada.**

- f) Agora crie a exceção `ChefeInvalidoException`, do tipo *checked*, e faça com que ela seja lançada no lugar da exceção atual do *setter* do atributo *chefe*.
- g) Na *main*, crie um novo `Funcionario`. Crie uma circunstância na *main* onde a exceção do item “f” ocorre. **Trate a exceção na main:** o programa deverá imprimir a mensagem *“Erro: um funcionário não pode ser chefe de si mesmo.”*.