

## Lista de Revisão – Computação 2

Aluno: Riquelme Gomes

DRE: 120032785

---

### Questão 1.

Uma classe representa um conjunto de objetos com características afins, e a mesma define o comportamento desses objetos, através das suas operações/métodos, e quais estados eles podem alcançar, através das suas propriedades/atributos. Já o objeto, por sua vez, é definido como uma instância de uma classe, sendo uma instância de uma classe um novo objeto criado dessa classe e sendo o construtor o responsável por instanciar essa classe que foi definida e por determinar que ações devem ser executadas quando da criação de um objeto. Além disso, um objeto é capaz de armazenar estados através de seus atributos e reagir a mensagens enviadas a ele, assim como se relacionar e enviar mensagens para outros objetos.

---

### Questão 2.

Essa troca de mensagens na linguagem Java ocorre por meio de métodos, os quais permitem que objetos interajam e se comuniquem com outros objetos. Um exemplo dessa troca ocorre com o método “debitar” da classe Banco, visto em aula. Nesse exemplo, um objeto da classe Banco (“meuBanco”) debita um valor, e portanto altera o saldo, de um outro objeto da classe Conta (“conta”), como pode ser visto abaixo.

```
// MÉTODO DEBITAR
public void debitar(String numero, double valor) {
    Conta conta = procurar(numero);
    if (conta != null) {
        if (valor <= conta.saldo()) conta.debitar(valor);
        else System.out.println("Saldo insuficiente!");
    }
    else System.out.println("Conta Inexistente!");
}
```

```
meuBanco.debitar( numero: "04.2021-2", valor: 500);
```

---

### Questão 3.

Atributos são as características/propriedades de um objeto, basicamente a estrutura de dados que vai representar a classe. Construtores são os responsáveis por instanciar a classe que foi definida e por determinar que ações devem ser executadas quando da criação de um objeto. E os métodos definem as “habilidades” dos objetos, isto é, são as ações que os objetos podem exercer quando solicitados, onde podem interagir e se comunicarem com outros objetos.

Com isso, métodos podem ser utilizados para alterar o valor de determinados atributos, gerando, conseqüentemente, uma mudança de estado de um objeto.

---

### Questão 4.

a. i) Livro.

ii) isbn. Tipo String. Visibilidade private.  
titulo. Tipo String. Visibilidade private.  
autor. Tipo String. Visibilidade private.  
ano. Tipo String. Visibilidade private.  
espessura. Tipo int. Visibilidade private.

iii) descricao. Visibilidade public. Nenhum parâmetro. Retorna String.  
getISBN. Visibilidade public. Nenhum parâmetro. Retorna String.  
getEspessura. Visibilidade public. Nenhum parâmetro. Retorna int.  
setEspessura. Visibilidade public. Parâmetro: int espessura. Retorna void.

\* ↓ Se considerarmos o construtor como sendo um método ↓ \*

Livro. Visibilidade public. Parâmetros: String isbn, String titulo, String autor, String ano e int espessura. Não possui tipo de retorno.

iv) Como parâmetros de um método são variáveis locais a ele, temos como variáveis locais ao construtor Livro: String isbn, String titulo, String autor, String ano e int espessura. Além disso, temos a variável local(parâmetro) ao método “setEspessura” que é: int espessura.

b. Sim, sua assinatura é:

```
public Livro(String isbn, String titulo, String autor, String ano, int espessura) { . . }
```

- c. Através de chamadas ao método “setEspessura”.
  - d. Sim, e de acordo com o mundo real isso não é correto, pois na realidade não conseguimos alterar a espessura de um livro, diferentemente do que ocorre na Classe Livro, onde podemos modificar a espessura do livro por meio do método “setEspessura”.
- 

### Questão 5.

a. (V)

Verdadeiro, pois quanto ao valor verdade, nesse caso, ambas as expressões retornam true. Embora o uso de “==” e “equals” não seja equivalente no geral.

Se considerarmos a equivalência entre o uso de “==” e “equals” em casos mais gerais e não apenas nas expressões presentes na questão, então a resposta será: Falso (F).

b. (V)

Verdadeiro, como `(true == !false)` e `(3 > 2)` são expressões verdadeiras, a expressão `(2 > 8) || (true == !false) && (3 > 2)` retorna true.

c. (V)

Verdadeiro, pois a expressão realmente retorna “10105”.

d. (V)

Verdadeiro, pois a expressão realmente retorna “100100”.

e. (V)

Verdadeiro, pois a expressão realmente retorna “100-1020”.

---

### Questão 6.

Após a execução desse trecho de código, temos empilhados: “A” e “B”, estando “B” no topo da pilha.

Na linha 02, é impresso “true” pois a pilha está vazia. Na linha 05 é impresso “B”, pois ele está no topo da pilha. Na linha 10 é impresso “B”, pois ele está no topo da pilha. Na linha 13 é


impresso “B”, pois ele está no topo da pilha. Na linha 14, é impresso “false” pois a pilha não está vazia.

1º Caso: Colocar o “tamanho” da pilha como um número negativo ou igual a zero faria a pilha manifestar um comportamento indesejado, pois seria criado um array de String com tamanho menor ou igual a zero e, com isso, não teríamos o espaço necessário para o funcionamento da nossa classe Pilha e seus métodos.

2º Caso: Utilizar o método “desempilha” em uma pilha vazia faria a pilha manifestar um comportamento indesejado, pois assim a variável “topo” teria valor negativo e, com isso, não seria possível colocar e/ou acessar elementos no array pilha.

3º Caso: Empilhar mais elementos que o “tamanho” da pilha comporta faria a pilha manifestar um comportamento indesejado, pois não haveria espaço suficiente no array “pilha” para comportar todos os elementos, causando problemas no funcionamento da nossa classe Pilha e de seus métodos.

---

**Questão 7.** Feita 

---