Universidade Federal do Rio de Janeiro
Centro de Ciências Matemáticas e da Natureza
INSTITUTO DE MATEMÁTICA
Departamento de Ciência da Computação

Disciplina	**MAB240 - Computação II**	
Professor	Anderson Gonçalves Uchôa	
Lista de Revisão		09/08/2021

1. A Programação Orientação a Objetos (POO) é um paradigma de programação largamente utilizado nos dias de hoje. Na POO a unidade de modularização é a **classe**. Descreva o que você entende por classe deixando claro qual é a diferença entre classe e objeto (OBS: utilize o conceito de **construtor** e **instanciação** para dar a sua resposta)

2. Na OO os objetos devem colaborar entre si para realizar algum processamento útil no sistema. Essa colaboração é feita através da troca de mensagens entre os objetos participantes da colaboração. Descreva, dando exemplo, como essa troca de mensagens ocorre na linguagem Java

3. Uma classe é composta por: **atributos** (campos ou propriedades), **construtores** e **métodos** (operações). Explique cada um desses conceitos. Faça uma relação entre **atributos**, **métodos** e o **estado** de um objeto.

```
01 public class Livro {
02
03 private String isbn;
04 private String titulo;
05 private String autor;
06 private String ano;
07 private int espessura;
08
09 public Livro(String isbn, String titulo, String autor, String
ano, int espessura) {
10 this.isbn = isbn;
11 this.titulo = titulo;
12 this.autor = autor;
13 this.ano = ano;
14 this.espessura = espessura;
15 }
16
17 public String descricao() {
18 return autor + ". " + titulo + ". " + ano + ". ISBN: " + isbn +
". (" + " cm.");
```

```
19 }
20
21 public String getISBN() {
22 return isbn;
23 }
24
25 public int getEspessura() {
26 return espessura;
27 }
28
29 public void setEspessura(int espessura) {
30 this.espessura = espessura;
```

4. Com base no trecho de código acima (classe **Livro**) responda o que se pede:
    a. Informe: i) o nome da classe; ii) o nome, o tipo e a visibilidade dos campos; iii) o nome, a visibilidade, os parâmetros e o tipo de retorno dos métodos; iv) todas as variáveis locais.
    b. Essa classe possui construtor? Se sim, qual é a sua assinatura dele?
    c. Objetos oriundos dessa classe podem sofrer alterações no seu estado interno através de chamadas a qual(is) método(s)?
    d. Na classe Livro é possível modificar a espessura do livro? De acordo com o mundo real, isso é correto? (Justifique sua resposta.)

5. Sobre valor verdade de expressões e operações com String, marque **V** para verdadeiro e **F** para falso:
    a. ( ) A expressão “A” == “A” é equivalente a “A”.equals(“A”)
    b. ( ) A expressão (2 > 8) || (true == !false) && (3 > 2) retorna true
    c. ( ) A expressão ‘’10” + 105 retorna “10105”
    d. ( ) A expressão ‘’100” + (80 + 20) retorna “100100”
    e. ( ) A expressão ‘’100” + -10 + 20 retorna “100-1020”

6. A **Pilha** é um tipo abstrato de dados bem conhecido. Basicamente, ela possui uma disciplina de acesso que permite adicionar um elemento no topo da pilha (**empilha**), remover um elemento apenas do topo da pilha (**desempilha**), consultar o elemento do topo da pilha (**topo**) e verificar se a pilha está vazia (**vazia**). Assuma que possamos construir uma pilha modular usando POO na linguagem Java como descrito abaixo:

```
01 public class Pilha {
02
03 private String[] pilha;
04 private int topo;
05
06 public Pilha(int tamanho) {
07 pilha = new String[tamanho];
08 topo = -1;
09 }
```

```
10
11 public boolean empilha(String elemento) {
12 pilha[++topo] = elemento;
13 return true;
14 }
15
16 public String desempilha() {
17 return pilha[topo--];
18 }
19
20 public String topo() {
21 return pilha[topo];
22 }
23
24 public boolean vazia() {
25 return topo < 0;
26 }
27 }
```

Considerando a classe **Pilha** acima, suponha o trecho de código abaixo que representa a manipulação de uma instância dessa **Pilha**. Qual é o estado da pilha após a execução desse trecho de código? Para cada comando **System.out.println(...)** diga o que é impresso (informe o número da linha). Descreva textualmente quais cenários (casos de teste), no mínimo três, que fariam com que a pilha manifestasse algum tipo de defeito (isto é comportamento indesejado).

```
01 Pilha pilha = new Pilha(5);
02 System.out.println(pilha.vazia());
03 pilha.empilha("A");
04 pilha.empilha("B");
05 System.out.println(pilha.topo());
06 pilha.desempilha();
07 pilha.empilha("B");
08 pilha.empilha("C");
09 pilha.desempilha();
10 System.out.println(pilha.topo());
11 pilha.empilha("C");
12 pilha.desempilha();
13 System.out.println(pilha.topo());
14 System.out.println(pilha.vazia());
```

7. Com base no trecho de código abaixo (classe **BlocoDeNotas**) responda o que se pede:

```
01 import java.util.ArrayList;
02
03 public class BlocoDeNotas {
04 private String proprietario;
05 private ArrayList<String> notas;
06
07 public BlocoDeNotas(){
```

```
08 this.proprietario = “Zé ninguém”;
09 this.notas = new ArrayList<String>();
10 }
11
12 public BlocoDeNotas(String proprietario){
13 this.proprietario = proprietario;
14 this.notas = new ArrayList<String>();
15 }
16
17 public void setProprietario(String proprietario){
18 this.proprietario = proprietario;
19 }
20
21 public String getProprietario(){
22 return this.proprietario;
23 }
24
25 public boolean adicionarNota(String nota){
26 return this.notas.add(nota);
27 }
28
29 public int numeroDeNotas(){
30 return this.notas.size();
31 }
32
33 }
```

Crie um projeto no Intellij chamado bloco-de-notas. Crie a classe **BlocoDeNotas** (igual a classe descrita). Em seguida crie a classe **Nota**. A classe Nota deve possuir dois campos do tipo String. O primeiro deve guardar o **tempo de agendamento**, isto é, a data e a hora (Ex.: “07/10/2009 - 12:00”) e o segundo a **descrição** da nota. A classe **Nota** deve possuir um método que retorne a descrição completa da nota, isto é, o tempo de agendamento e a descrição (o método deve ter o nome **descricaoDetalhada**).

Agora, altere a classe **BlocoDeNotas** para trabalhar com notas do tipo **Nota** e não mais com notas do tipo **String**. Depois disso, adicione métodos na classe **BlocoDeNotas** que permitam: i) imprimir todas as notas informando a data, hora e a descrição de cada uma das notas (o nome do método deve ser **imprimirNotas**); ii) selecionar (e retornar) uma nota específica, isto é, em uma dada posição (o método deve ter o nome **exibirNota**); iii) remover uma nota específica, numa dada posição, (o método deve ter o nome **removerNota**).