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**		**Semestre**		**2021.1**
Professor	Anderson Gonçalves Uchôa				
Aluno			DRE		
AP1		**Data**	18/08/2021	**Nota**	

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**. Uma **classe** é normalmente composta por: **atributos** (propriedades), **construtores** e **métodos** (operações). Com suas palavras descreva o que você entende por **classe** deixando claro qual é a diferença entre **classe** e **objeto.** Além disso, descreve os conceitos de **atributos**, **construtores** e **métodos**. (1.0)

```
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 }
```

2. Considerando a classe `Pilha` descrita anteriormente e o trecho de código abaixo. Indique, marcando com um **X**, onde ocorre: (i) instanciação; (ii) mudança de estado; e (iii) invocações a métodos de acesso (isto é, métodos que não modificam o estado do objeto, apenas retornam valores): (1.0)

Código fonte	(i)	(ii)	(iii)
`01 Pilha pilha = new Pilha(2);`			
`02 pilha.empilhar("A");`			
`03 pilha.desempilhar();`			
`04 pilha = new Pilha(4);`			
`05 pilha.empilhar("A");`			
`06 pilha.topo();`			
`07 pilha.empilhar("B");`			
`08 pilha.empilhar("C");`			
`09 pilha.vazia();`			

3. Em relação a tipo estático e tipo dinâmico e observando a **Figura 2**, avalie as afirmações marcando V para verdadeiro e F para falso. (1.0)

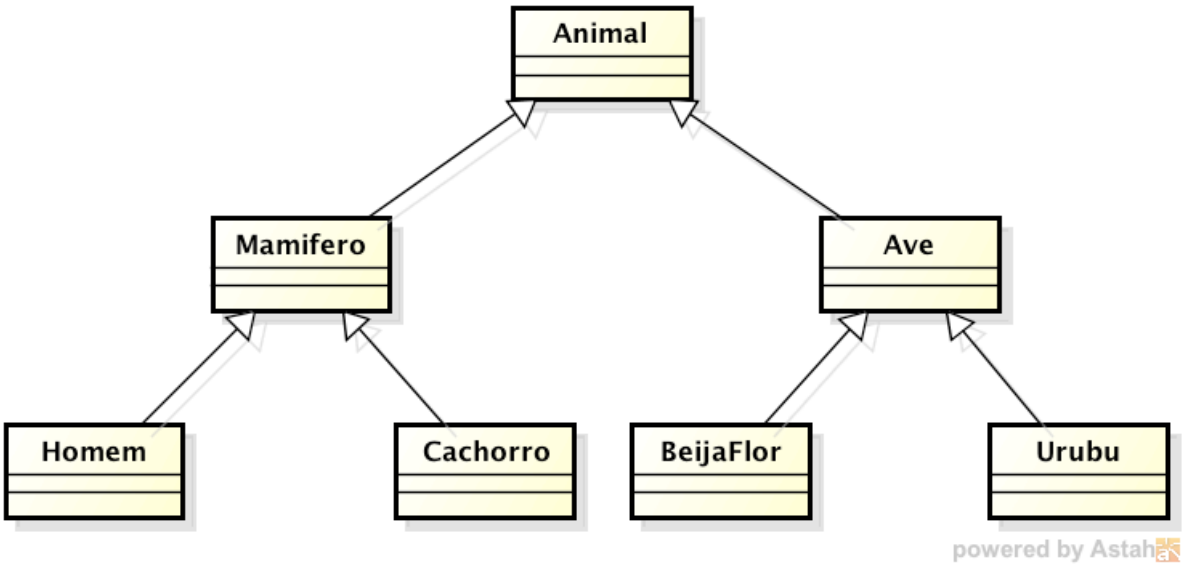

Figura 2. Hierarquia de Classes.

a. ( ) Um `Cachorro` pode possuir um tipo dinâmico `Animal`.

b. ( ) Uma variável `Animal a` pode possuir tipos dinâmicos `Homem` e `Urubu`.
c. ( ) É possível atribuir um tipo dinâmico `BeijaFlor` à variável `Urubu u`.
d. ( ) É possível atribuir um tipo dinâmico `Cachorro` à variável `Urubu u`.
e. ( ) O tipo estático da variável a é `Animal`. Logo, essa variável pode possuir como tipo dinâmico qualquer tipo de hierarquia de classes.

4. Utilizando os conceitos de **herança**, **subtipo** e **polimorfismo** de **tipo/classe**, responda se os trechos de código abaixo estão corretos (do ponto de vista do compilador Java), marcando V, para os corretos, e F, para os não corretos. Em todos os casos (verdadeiro ou falso), **JUSTIFIQUE** a sua resposta. (1.5)

(__) Trecho 01	`01 Animal a = new Animal();`
	`02 Urubu u = new Urubu();`
	`03 a = u;`

(__) Trecho 02	`01 Animal a = new Mamifero();`
	`02 Homem h = new Homem();`
	`03 a = h;`

(__) Trecho 03	`01 Animal a = new Ave();`
	`02 Mamifero c = new Cachorro();`
	`03 a = c;`

(__) Trecho 04	`01 Animal a = new Homem();`
	`02 Homem h = new Homem();`
	`03 h = a;`

(__) Trecho 05	`01 Mamifero m = new Cachorro();` `02 Cachorro h = m;`

Descrição do Programa: Estante de Livros
Em uma estante, os livros são acomodados em prateleiras obedecendo a uma ordem espacial, isto é, enquanto houver espaço livre, livros podem ser adicionados nas prateleiras. Os livros possuem vários atributos como, por exemplo: ISBN, título, autor, ano de publicação, número da edição, número de páginas e espessura. Uma prateleira pode acomodar vários livros. Da mesma forma, uma estante pode acomodar várias prateleiras. O objetivo desse programa é permitir que os livros possam ser inseridos, removidos e consultados nas prateleiras das estantes.

5. Crie uma classe `Livro`. A classe livro deve possuir os atributos: isbn, titulo, autor, ano de publicação, número da edição, número de páginas e espessura. Além disso, para todos os atributos, deve ser codificado métodos de acesso do tipo `getXXX`, onde `XXX` é o nome do atributo. Por exemplo, para o campo `private String titulo` teremos o método `public String getTitulo()`. Um método `public String descricao()`, que **retorna** os detalhes do livro em uma `String`, deve ser implementado. Um exemplo desse tipo de descrição seria: "Livro: Programação Orientada a Objetos | Autor: Anderson Uchôa | Ano: 24021| Edição: 1ed | Páginas: 200 | Espessura: 10cm.". A classe `Livro` deve possuir um construtor que recebe como argumento os valores para os seus campos no instante da instanciação. (1.0)

6. Uma estante de livros é composta por prateleiras. Uma prateleira deve possuir um atributo que descreve o tamanho máximo, em centímetros, (`private int tamanhoMaximo`) de espaço para armazenamento de livros. Além disso, ela deve possuir um atributo que é uma coleção de livros (`private Vector<Livro> livros`). Os métodos da classe `Prateleira` devem seguir a especificação abaixo: (2.0)
    a. `public Prateleira(int tamanho)`: é o construtor da classe que inicializa o atributo `tamanhoMaximo` com o valor do parâmetro tamanho e instancia o atributo livros;
    b. `private int espacoLivre()`: calcula o espaço livre na prateleira;
    c. `public boolean adicionarLivro(Livro livro)`: adiciona um livro na prateleira observando o espaço livre disponível. Ele deve retornar verdadeiro se a operação for concluída com êxito ou falso em caso contrário;
    d. `public boolean removerLivro(String isbn)`: remove o livro cujo ISBN for igual ao parâmetro informado. Ele deve retornar verdadeiro se a operação for concluída com êxito ou falso em caso contrário;
    e. `public Livro selecionarLivro(String isbn)`: seleciona um livro na prateleira. Ele retorna o livro procurado se o livro for encontrado e null em caso contrário;
    f. `public void imprimirLivros()`: imprime os detalhes de todos os livros da prateleira.

7. Construa a classe `Estante`. Essa classe deve possuir o atributo privado `prateleiras` do tipo `Vector<Prateleira>` onde são colocadas as prateleiras que guardam os livros. A implementação dos métodos da classe `Estante` deve ser feita seguindo as especificações abaixo: (2.0)
    a. `public Estante()`: é o construtor da classe que inicializa o atributo prateleiras;
    b. `public void adicionarPrateleira(Prateleira prateleira)`: faz a adição de prateleiras à estante de livros;
    c. `public boolean adicionarLivro(Livro livro)`: faz a adição de um

livro na primeira prateleira com espaço livre disponível. Ele deve retornar verdadeiro se a operação for concluída com êxito ou falso em caso contrário;

d. `public boolean removerLivro(Livro livro)`: faz a remoção de um livro da prateleira onde ele se encontra. Ele deve retornar verdadeiro se a operação for concluída com êxito ou falso em caso contrário;
e. `public Livro selecionarLivro(String isbn)`: procura o livro pelo seu ISBN pelas prateleiras até encontrar. Ele deve retornar o livro procurado, se o livro for encontrado, ou `null` em caso contrário;
f. `public void imprimirLivros()`: imprime os detalhes de todos os livros de todas as prateleiras.