Exercícios Teóricos – u00i: Ponteiros

Catarina F. M. Castro (803531) – AEDs II

1-

- O que significa cada um dos trechos de código abaixo?

```
int [ ] vet
```

```
= new int [5]
```

```
int [ ]  vet = new  int [5];
```

O primeiro trecho representa um ponteiro denominado “vet”. O segundo trecho representa a alocação de um 5 espaços de memória para armazenamento de números inteiros. Por fim, o terceiro trecho representa “vet” apontando para os espaços de memória alocada.

2-

- Explique o que o programa abaixo imprime na tela

```
class Ponteiro01Array {

    public static void main (String[] args) {
        int[] vet = new int [5];
        escrever(vet);

        vet = new int [5];
        escrever(vet);
    }

}
```

Na tela, será impresso o primeiro endereço de memória dos 5 alocados na declaração do ponteiro e, em seguida, um outro endereço equivalente ao primeiro dos outros 5 que foram posteriormente alocados.

3-

- Faça o quadro de memória do programa abaixo

```
class Ponteiro02PassagemTipoPrimitivo {
    public static void passagemDeTipoPrimitivo(int a){
        escrever("a: " + a);
        a = 10;
        escrever("a: " + a);
    }
    public static void main(String[] args) {
        int x = 5;
        escrever("x: " + x);
        passagemDeTipoPrimitivo(x);
        escrever("x: " + x);
    }
}
```

x	a	Tela
5	5	x: 5
	10	a: 5
		a: 10
		x: 5

4-

- Faça o quadro de memória e mostre a saída na tela

```
class Ponteiro03PassagemArray {
    public static void passagemDeArray(int[ ] b){
        for (int i = 0; i < 5; i++){
            b[i] *= 5;        escrever("b[" + i + "]: " + b[i]);
        }
        b = new int [5];
        for (int i = 0; i < 5; i++){
            b[i] = i;         escrever("b[" + i + "]: " + b[i]);
    }   }
    public static void main(String[] args) {
        int [] y = new int [5];
        for (int i = 0; i < 5; i++){
            y[i] = i;         escrever("y[" + i + "]: " + y[i]);
        }
        passagemDeArray(y);
        for (int i = 0; i < 5; i++){
            escrever("y[" + i + "]: " + y[i]);
}   }   }
```

y[0]	y[1]	y[2]	y[3]	y[4]	b[0]	b[1]	b[2]	b[3]	b[4]
0	1	2	3	4	0	5	10	15	20
0	5	10	15	20	0	1	2	3	4
0	1	2	3	4	5				

Tela
y[0] : 0
y[1] : 1
y[2] : 2
y[3] : 3
y[4] : 4
b[0] : 0
b[1] : 5
b[2] : 10
b[3] : 15
b[4] : 20
b[0] : 0
b[1] : 1
b[2] : 2
b[3] : 3
b[4] : 4
y[0] : 0
y[1] : 5
y[2] : 2
y[3] : 3
y[4] : 4

5-

- Seja a classe abaixo ...

```
class Cliente {
    private int codigo;
    private String nome;
    public Cliente (){
        this.codigo = 0;            this.nome = "";
    }
    public Cliente (int codigo, String nome){
        this.codigo = codigo;   this.nome = nome;
    }
    public int getCodigo(){                 return codigo;          }
    public void setCodigo(int codigo){      this.codigo = codigo;   }
    public String getNome(){                return nome;            }
    public void setNome(String nome){       this.nome = nome;       }
}
```

- ... o que significa cada um dos trechos de código abaixo?

```
Cliente c;
```

```
= new Cliente ();
```

```
Cliente c = new Cliente ();
```

O primeiro trecho indica a criação de um ponteiro chamado “c”. O segundo trecho representa a alocação de um espaço na memória para guardar um objeto da classe “Cliente”. E, por fim, o terceiro trecho faz “c” apontar para esse espaço na memória.

6-

- Faça o quadro de memória do programa abaixo

```
class Ponteiro04Objeto {
   public static void main (String[] args){
       Cliente c1 = null, c2 = null, c3 = null;
       escrever("ADDRs:\nc1(" + c1 + ")\nc2(" + c2 + ")\nc3(" + c3 + ")");
       c1 = new Cliente(1, "aa");        c2 = c1;    c3 = new Cliente(2, "bb");
       escrever("ADDRs:\nc1(" + c1 + ")\nc2(" + c2 + ")\nc3(" + c3 + ")");
       c2.setCodigo(3);
       escrever("ATRIBUTOs:");
       escrever("c1(" + c1.getCodigo() + " / " + c1.getNome() + ")");
       escrever("c2(" + c2.getCodigo() + " / " + c2.getNome() + ")");
       escrever("c3(" + c3.getCodigo() + " / " + c3.getNome() + ")");
}    }
```

c1 (7Ah)	c2	c3 (A5h)
3, aa	7Ah	2, bb

Tela
ADDRs: c1(null) c2(null) c3(null) ADDRs: c1(7Ah) c2(7Ah) c3(A5h) ATRIBUTOs: c1(3/aa) c2(3/aa) c3(2/bb)

7-

- Faça o quadro de memória do programa abaixo

```
class Ponteiro05PassagemObjeto {
   public static Cliente setar2(Cliente y){
      y.setCodigo(6);    y.setNome("ff");
      return y;
   }
   public static void setar1(Cliente x){
      x.setCodigo(4);    x.setNome("dd");    x = new Cliente (5, "ee");
   }
   public static void main (String[] args){
      Cliente c1 = new Cliente(1, "aa"), c2 = null; c3 = new Cliente(2, "bb");
      c2 = c1;
      setar1(c1);
      c3 = setar2(c2);
}     }
```

c1 (33h)	c2	c3	x
1, aa	null	2, bb	5, ee
4, dd	(33h)	(33h)	
6, ff			

8-

- Na verdade, no comando c2 = c1 do exercício anterior, o programador gostaria que os atributos do objeto apontado por c2 fossem iguais aos do objeto apontado por c1, contudo, apontando para objetos distintos. Como podemos ajudá-lo?

Para copiar os conteúdos de c1 em c2, é preciso copiar cada atributo individualmente. A fim de facilitar isso é possível criar um método "clone", que já realiza esse tipo de cópia.

```
public Cliente clone (){
        Cliente resp = new Cliente();
        resp.codigo = this.codigo;
        resp.nome = this.nome;
        return resp;
}
```

9-

- Mostre a alteração anterior na classe Ponteiro04Objeto

c1 (7Ah)	c2 (9Ah)	c3 (A5h)
1, aa	1, aa	2, bb
	3, aa	

Tela
ADDRs:
c1(null)
c2(null)
c3(null)
ADDRs:
c1(7Ah)
c2(9Ah)
c3(A5h)
ATRIBUTOs:
c1(3/aa)
c2(3/aa)
c3(2/bb)

10-

- Mostre o quadro de memória para o programa abaixo

```
class  Ponteiro08Objeto {
   public static void main (String[] args){
     Cliente c1 = new Cliente(1, "aa");
     Cliente vet[] = new Cliente [5];
     sop(c1 + "/" + c1.getCodigo() + "/" + c1.getNome());
     for (int i = 0; i < vet.length; i++){
       vet[i] = c1.clone();
       System.out.println(vet[i] + "/" + vet[i].getCodigo() + "/" + vet[i].getNome());
     }
   }
}
```

c1 (3Ah)	vet[0] (77h)	vet[1] (78h)	vet[2] (79h)	vet[3] (7Ah)	vet[4] (7Bh)
1, aa	1, aa	1, aa	1, aa	1, aa	1, aa

Tela
3Ah/1/aa
77h/1/aa
78h/1/aa
79h/1/aa
7Ah/1/aa
7Bh/1/aa

11-

• Um estudante de Algoritmos e Estruturas de Dados (em JAVA) implementou uma classe Hora, cujo construtor recebe e armazena uma hora, minuto e segundo. O que acontece se a classe X abaixo for colocada na mesma pasta que a classe Hora?

```
class X {
  public static void main (String[] args){
    Hora h1 = new Hora(12, 30, 30);
    Hora h2 = new Hora(12, 30, 30);
    if (h1 == h2)
       System.out.println("Identicos!");
    else
       System.out.println("Diferentes!");
  }
}
```

A) Escreve na tela "Identicos!".

B) Escreve na tela "Diferentes".

C) Erro de compilação.

D) Erro de execução na linha do if.

E) Erro de execução na declaração objetos.

RESPOSTA: letra D, pois a comparação entre h1 e h2 dentro do IF está relacionando os endereços dos dois objetos, e não o seu conteúdo. Dessa forma, como os endereços são diferentes, irá ser impresso "Diferentes!" na tela.

12-

• Seja a classe X abaixo e a Animal implementada e não mostrada, avalie as afirmações listadas a seguir.

```
class X {
  public static void main (String[] args){
    Animal a = new Animal ("Cao", 32, 'a');
    Animal b = new Animal ("Cao", 'x');
    Animal c = b;
    c.nome = "Gato";
    System.out.println(b.nome);
    c.setIdade(45);
} }
```

I – Possivelmente, a Classe Animal tem três ou mais atributos. Além disso, no construtor com três parâmetros, o atributo que recebe valor do primeiro parâmetro pode ser do tipo String e os que recebem os outros dois podem ser do tipo int.

II - O comando System.out.println(b.nome) imprime a palavra "Gato".

III - A classe Animal deve ter um atributo idade e esse será obrigatoriamente privado.

IV - Na classe animal o atributo nome tem que ser estático.

É correto apenas o que se afirma em: A) I e II. B) II e III. C) III e IV. D) I, II e III.

RESPOSTA: letra A.