Pergunta 1 (1 ponto) ✓ *Salvo*

Questão 05) O trecho de código abaixo mostra um exemplo da utilização de vetores, em Java:

MeuPrograma.java

```java
import java.util.Arrays;

public class MeuPrograma {

    public static void main(String[] args) {
        // Declara vetor
        int vetor[] = new int[5] ;

        // Guarda valores no vetor
        vetor[0] = 20 ;
        vetor[1] = 40 ;
        vetor[2] = 10 ;
        vetor[3] = 60 ;
        vetor[4] = 50 ;

        // Mostra conteudo do vetor
        System.out.println("Antes: " + Arrays.toString(vetor));

        // Efetua operacao
        for(int i=0; i < vetor.length; i++) {
            for(int j=i+1; j < vetor.length; j++) {
                if(vetor[i] > vetor[j]) {
                    int aux = vetor[i] ;
                    vetor[i] = vetor[j] ;
                    vetor[j] = aux ;
                }
            }
        }

        // Mostra conteudo do vetor
        System.out.println("Depois: " + Arrays.toString(vetor));
    }
}
```

Tendo como referência esse código, assinale a alternativa INCORRETA:

- A operação executada entre as linhas 19 e 28 ordena os elementos do vetor, de acordo com o conhecido "algoritmo da bolha".
- Vetores, em Java, são objetos. Por isso usamos o operador "new", na linha 7, para criar um objeto do tipo vetor. O vetor criado, por sua vez, tem capacidade para armazenar 5 valores inteiros.
- Vetores são variáveis que podem armazenar mais de um valor. Cada valor armazenado no vetor, por sua vez, é referenciado por uma determinada posição, ou "índice". Em Java, o primeiro elemento de um vetor está armazenado na posição 1.
- Nesse programa, Se trocarmos o comando "int vetor[] = new int[5]" por "int vetor[] = { 0, 0, 0, 0, 0 }" o programa vai funcionar como antes, produzindo a mesma saída.
- O programa, quando executado, imprime na tela as mensagens "Antes: [20, 40, 10, 60, 50]" e "Depois: [10, 20, 40, 50, 60]"

Pergunta 2 (1 ponto)

Questão 04) O trecho de código abaixo mostra um exemplo da utilização do comando "while", em Java:

MeuPrograma.java

```
1  import java.util.Scanner;
2
3  public class MeuPrograma {
4
5      public static void main(String[] args) {
6          Scanner in = new Scanner(System.in) ;
7
8          // Inicializa variavel
9          int a = 20 ;
10
11         // Executa repeticao
12         while(a > 10) {
13             // Solicita valor para a variavel
```

```
14              System.out.print("Entre com o valor de a: ");
15              a = in.nextInt() ;
16
17              // Mostra valor
18              System.out.printf("Foi informado o valor %d \n", a);
19          }
20      }
21  }
22
```

Tendo como referência esse código, assinale a alternativa INCORRETA:

- (●) Podemos trocar o comando "while", nesse programa, pelo comando "for" mantendo a mesma condição de loop, porque o número de repetições é conhecido.
- ( ) Nesse programa, os comandos que aparecem dentro do "while" serão executados enquanto o usuário informar valores maiores que 10.
- ( ) Durante a execução do programa, se o usuário informar o valor 10, a repetição vai parar.
- ( ) Podemos trocar o comando "while", nesse programa, pelo comando "do-while", mantendo a mesma condição de loop. Ao fazer isso o programa vai funcionar como antes, produzindo as mesmas saídas.
- ( ) Nesse programa, os comandos que aparecem dentro do "while" serão executados pelo menos uma vez.

Pergunta 3 (1 ponto)

Questão 07) Por meio de uma classe podemos agrupar em uma mesma entidade (um objeto) um conjunto de dados relacionados. Para isso, declaramos dentro do corpo da classe diversas "variáveis", que no jargão da orientação a objetos chamamos de "atributos". O programa abaixo mostra como declarar uma classe "Aluno" contendo os atributos "nome" e "matricula", e como fazer uso dessa classe para armazenar dados.

```
1  class Aluno {
2      // Atributos da classe aluno
3      private String nome ;
4      private String matricula ;
5
6      // Métodos get/set
7      public String getNome() {
8          return nome;
9      }
10     public void setNome(String nome) {
11         this.nome = nome;
12     }
13     public String getMatricula() {
14         return matricula;
15     }
16     public void setMatricula(String matricula) {
17         this.matricula = matricula;
18     }
19 }
20
21 public class Programa{
22   public static void main(String args[]){
23         Aluno aluno = new Aluno() ;
24         aluno.setNome("Luiz Reginaldo");
25         aluno.setMatricula("123456");
26         System.out.printf("Nome: %s \n", aluno.getNome());
27         System.out.printf("Matricula: %s \n", aluno.getMatricula());
28   }
```

Tendo como referência esse código, que foi apresentado na solução de um dos exercícios resolvidos da nossa disciplina, assinale a alternativa INCORRETA:

- A classe "Programa", nesse exemplo, não pode referenciar diretamente os atributos "nome" e "matricula" da classe "Aluno". Isso acontece porque os atributos foram declarados com o modificador "private".

- Na linha 23 é criada uma instância da classe "Aluno", por meio da instrução "new Aluno()". Como essa instrução está sendo atribuída à variável "aluno", a instância criada passa a ser referenciada por essa variável.

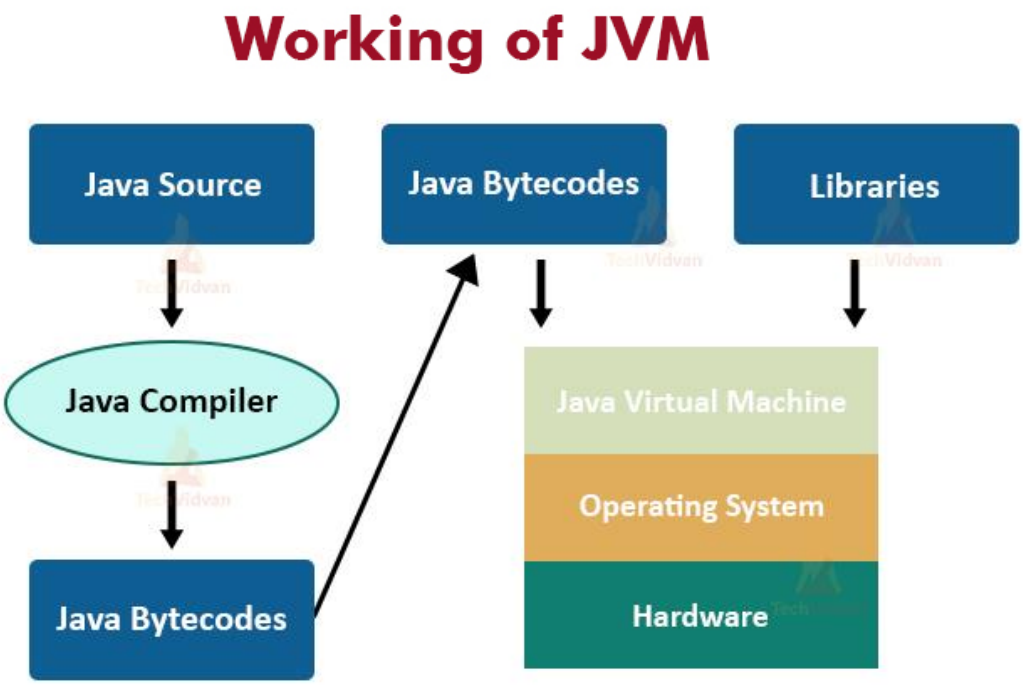
- Na classe "Aluno" os atributos "nome" e "matricula" são declarados nas linhas 3 e 4 do código. Ambos atributos tem o tipo "String", e correspondem a variáveis que serão utilizadas para guardar os dados de um aluno.

- O acesso aos atributos "nome" e "matricula" é realizado por meio de métodos "get" e "set" declarados na classe "Aluno". Podemos observar isso nas linhas 24 e 25, onde os métodos "setNome" e "setMatricula" são chamados para definir valores para os atributos, e nas linhas 26 e 27, onde os métodos "getNome" e "getMatricula" são chamados para obter os valores dos atributos.

- Nesse exemplo, poderíamos substituir a chamada "aluno.getNome()", na linha 26, por "aluno.nome", e "aluno.getMatricula()", na linha 27, por "aluno.matricula". Ao fazer isso, o programa continuará funcionando como antes.

Pergunta 4 (1 ponto)

Questão 01) Um dos grandes benefícios da linguagem Java é que ela permite executar um mesmo código compilado em diferentes plataformas e sistemas operacionais. Assim, depois que um programa Java é compilado, independentemente do sistema operacional em que ocorreu o processo de compilação, ele pode então ser executado em Linux, Windows, Mac OS ou qualquer outro sistema operacional que tenha disponível uma Máquina Virtual Java (Java Virtual Machine, ou JVM). Esse conceito de multiplaforma, provido pelo Java, é frequentemente referido pelo slogan "Write once, run anywhere", cuja tradução é "Escreva uma vez, execute em qualquer lugar". A figura abaixo ilustra essa ideia.



Tendo como referência esse tema, assinale a alternativa INCORRETA:

- O arquivo fonte contém instruções da linguagem Java, enquanto o arquivo compilado contém bytecodes, que são instruções para uma Máquina Virtual Java (Java Virtual Machine).
- A Máquina Virtual Java (Java Virtual Machine) executa sobre um Sistema Operacional, provendo para os programas Java um ambiente de execução comum a diversas plataformas, como Linux e Windows.

○ Outras linguagens de programação, como C e Pascal, geram instruções para um sistema operacional específico. Por isso o código compilado em Linux, por exemplo, não pode ser executado no Windows. O compilador Java, por sua vez, produz instruções para uma Máquina Virtual Java (JVM). Como essas instruções são executadas pela JVM, basta então que a plataforma (Linux, Windows) tenha disponível uma JVM para que o programa compilado possa ser executado nessa plataforma.

◉ A Máquina Virtual Java pode ser vista como "o ambiente onde os programas Java são executados". Por ser escrita em Java, a mesma máquina virtual que executa em ambiente Linux também executa em ambiente Windows. Essa independência de plataforma, por sua vez, ilustra bem o slogan "Write once, run anywhere".

○ A independência de plataforma provida pelo Java se baseia no conceito de máquina virtual. Nesse contexto, o compilador Java transforma o código fonte escrito em Java em instruções que podem ser reconhecidas por uma Máquina Virtual Java.

Pergunta 5 (1 ponto)

Questão 02) O trecho de código abaixo mostra um exemplo da utilização de comandos de leitura de variáveis, em Java:

MeuPrograma.java

```
1  import java.util.Scanner;
2
3  public class MeuPrograma {
4
5      public static void main(String[] args) {
6          Scanner in = new Scanner(System.in) ;
7
8          // Solicita valores para as variaveis
9          System.out.print("Entre com o valor de a: ");
10         int a = in.nextInt() ;
11
12         System.out.print("Entre com o valor de b: ");
13         int b = in.nextInt() ;
14
15         // Efetua operacao
```

```
16          a = a + b ;
17          b = a - b ;
18          a = a - b ;
19
20          // Mostra resultados
21          System.out.printf("Depois da operacao o valor de a = %d \n", a);
22          System.out.printf("Depois da operacao o valor de b = %d \n", b);
23      }
24  }
25
```

Tendo como referência esse código, assinale a alternativa INCORRETA:

- Nesse programa, usamos um objeto do tipo "Scanner" para fazer a leitura de valores para as variáveis. Esse objeto, por sua vez, é instanciado na linha 6.
- Podemos escrever dados na tela usando "System.out.print" (como ilustrado nas linhas 9 e 12), ou ainda usando "System.out.printf" (como ilustrado nas linhas 21 e 22). O uso de "System.out.printf", nesse caso, é semelhante ao uso da função "printf" em C.
- A instrução "in.nextInt()" é usada para solicitar um valor ao usuário do programa.
- Se trocarmos o tipo das variáveis "a" e "b" para "String", e a instrução "in.nextInt()" para "in.nextLine()", a operação que ocorre entre as linhas 15 e 18 produzirá o mesmo resultado que aquele observado quando as variáveis tinham o tipo "int".
- A operação que ocorre entre as linhas 15 e 18 faz a troca dos valores armazenados nas variáveis a e b.

Pergunta 6 (1 ponto)

Questão 06) As funções, estudadas na disciplina de Programação Estruturada, no jargão da Programação Orientada a Objetos são chamadas de "métodos". Ao criar um método estamos criando uma estrutura de código que contém comandos e que pode ser chamada diversas vezes dentro do nosso programa, favorecendo assim o reuso de código.

O programa apresentado abaixo mostra uma solução para a conversão de temperatura de celsius para fahrenheit, e que faz uso de um método para fazer a conversão.

```
1  import java.util.Scanner;
2
3  public class Programa {
4
5      public static void main(String[] args) {
6          // Configura a leitura de dados do teclado
7          Scanner in = new Scanner( System.in );
8
9          // Solicita a temperatura em Celsius
10         System.out.print("Entre com a temperatura em Celsius: ");
11         double celsius = in.nextDouble() ;
12
13         // Converte para Farenheit
14         double tempConvertida =  fahrenheit(celsius) ;
15         System.out.printf("A temperatura em fahrenheit = %.2f ", tempConvertida);
16     }
17
18     /** Converte a temperatura */
19     public static double fahrenheit(double celsius) {
20         double tempFahrenheit = (9 * celsius + 160) / 5 ;
21         return tempFahrenheit ;
22     }
23 }
```

Tendo como referência esse código, que foi apresentado na solução de um dos exercícios resolvidos da nossa disciplina, assinale a alternativa INCORRETA:

- ○ Na linha 21 encontramos um comando "return". Esse comando, por sua vez, faz com que o método em questão retorne o valor armazenado na variável "tempFahrenheit". Podemos observar que esse valor tem o tipo "double", que corresponde ao tipo de retorno do método.

- ○ Um método, em Java, possui um nome, um tipo de retorno e argumentos. Nesse exemplo, o método declarado na linha 19 tem o nome "fahrenheit", o tipo de retorno "double" e um argumento do tipo "double", que corresponde ao parâmetro "celsius".

- ◉ Se removermos o modificador "static" da declaração do método "fahrenheit", na linha 19, esse programa continuará funcionando como antes.

- ◯ O método declarado na linha 19, nesse exemplo, possui o modificador "static". Isso quer dizer que o método pertence à classe, e por isso pode ser chamado diretamente de "main" sem a necessidade de ser criado um objeto do tipo "Programa".

- ◯ Na linha 14 podemos observar uma chamada ao método "fahrenheit". Nesse caso, estamos passando para o método o valor armazenado na variável "celsius". O valor retornado pelo método, por sua vez, está sendo armazenado na variável "tempConvertida".

Pergunta 7 (1 ponto)

Questão 10) Algumas linguagens de programação definem o tipo "texto" como um tipo pré-definido da linguagem. Em Java, no entanto, esse tipo de dados é representado pela classe "String". A vantagem dessa abordagem é que, além de prover o armazenamento de cadeias de caracteres, o tipo provê também diversos métodos utilitários para manipulação e tratamento das cadeias de caracteres. Tendo como referência a classe String, em Java, assinale a alternativa INCORRETA:

- ◯ O método "indexOf" retorna a posição onde está armazenado um caractere na cadeia. Caso o caractere não esteja presente na cadeia, "indexOf" retorna -1.

- ◉ O método "equals" é utilizado para comparar duas cadeias de caracteres, e possui o mesmo comportamento do operador relacional == (igual), em Java.

- ◯ O método "toUpperCase" retorna uma nova cadeia contendo todos os caracteres em caixa alta (maiúsculas).

- ◯ O método "length" retorna a quantidade de caracteres da cadeia.

- ◯ O método "charAt" retorna o caractere que está armazenado em uma dada posição da cadeia.

Pergunta 8 (1 ponto)

Questão 03) O trecho de código abaixo mostra um exemplo da utilização do comando "if", em Java:

MeuPrograma.java

```
1  import java.util.Scanner;
2
3  public class MeuPrograma {
4
5      public static void main(String[] args) {
6          Scanner in = new Scanner(System.in) ;
7
8          // Solicita valor para o usuário
9          System.out.print("Entre com o valor de a: ");
10         int a = in.nextInt() ;
11
12         // Efetua operacao
13         if(a < 15) {
14             System.out.printf("Executa 1");
15         } else if(a < 30) {
16             System.out.printf("Executa 2");
17         } else if(a < 45) {
18             System.out.printf("Executa 3");
19         } else if(a < 60) {
20             System.out.printf("Executa 4");
21         } else {
22             System.out.printf("Executa 5");
23         }
24     }
25 }
26
```

Tendo como referência esse código, assinale a alternativa INCORRETA:

- ○ Nesse programa, se o usuário informar o valor -10, será impresso na tela a mensagem "Executa 1".
- ○ Nesse programa, se o usuário informar o valor 20, será impresso na tela a mensagem "Executa 2".
- ○ Esse código mostra um exemplo do uso de "ifs" aninhados. Nesse tipo de estrutura, será executado o comando que corresponde à primeira condição verdadeira do conjunto, partindo-se da análise das condições do primeiro ao último "if".
- ◉ Nesse programa, se o usuário informar o valor 60, será impresso na tela a mensagem "Executa 4".
- ○ Nessa estrutura de "ifs" aninhados, se nenhuma condição for verdadeira, será executado o comando do último "else".

Pergunta 9 (1 ponto)

Questão 08) Durante nossos estudos aprendemos que uma classe pode conter atributos e métodos. O construtor de uma classe, segundo alguns autores, pode ser considerado como um tipo especial de método, porque embora não tenha um retorno explícito (como os métodos comuns) é chamado para criar objetos (ou instâncias) de uma classe. O código abaixo mostra um exemplo de classe contendo atributos, métodos e um construtor:

```
1  class Aluno {
2      // Atributos da classe aluno
3      private String nome ;
4      private double nota1, nota2 ;
5
6      /** Construtor que recebe os dados de um aluno */
7      public Aluno(String nome, double nota1, double nota2) {
8          this.nome = nome ;
9          this.nota1 = nota1 ;
10         this.nota2 = nota2 ;
11     }
12
```

```
13      // Métodos get
14      public String getNome() {
15          return nome;
16      }
17
18      public double getNota1() {
19          return nota1;
20      }
21
22      public double getNota2() {
23          return nota2;
24      }
25  }
26
27  public class Programa{
28    public static void main(String args[]){
29          Aluno aluno = new Aluno("Joãozinho", 7.0, 6.0) ;
30          System.out.printf("Nome: %s \n", aluno.getNome());
31          System.out.printf("Notas: %.2f e %.2f \n", aluno.getNota1(), aluno.getNota2());
32    }
33  }
```

Tendo como referência esse código, que corresponde a um dos exercícios resolvidos da nossa disciplina (com adaptações), assinale a alternativa INCORRETA:

- Nas linhas 14, 18 e 22 declaramos métodos "get". Esses métodos são usados para recuperar o valor armazenado nos atributos da classe, e seguindo a regra do encapsulamento foram declarados com visibilidade "pública". Esses métodos são chamados nas linhas 30 e 31 do código.
- Na linha 7 declaramos um construtor para a classe "Aluno". Esse construtor nos informa que, para criarmos um objeto do tipo "Aluno", é necessário informar o nome e as duas notas do aluno. Isso é feito na chamada que aparece na linha 29 do código.
- Nas linhas 8, 9 e 10 guardamos nos atributos da classe "Aluno" os valores que foram informados para o construtor. A palavra reservada "this", nesse caso, serve para diferenciar um atributo da classe de um parâmetro do construtor, visto que os dois tem o mesmo nome.

- ○ Nessa classe não declaramos métodos "set". Esses métodos, quando declarados, servem para informar novos valores para os atributos da classe, e seguindo a regra do encapsulamento devem ser declarados com visibilidade "pública". Nesse exemplo, como um objeto do tipo "Aluno" não terá seus dados alterados, depois de criado, optou-se por não incluir na classe métodos "set".

- ◉ Nas linhas 3 e 4 declaramos os atributos da classe "Aluno". Esses atributos são declarados com visibilidade "privada". No entanto, para estar de acordo a regra do encapsulamento, esses atributos deveriam ter sido declarados com visibilidade "pública".

Pergunta 10 (1 ponto)

Questão 09) Com o uso da classe ArrayList podemos construir listas dinâmicas de objetos. Essas listas, diferente dos vetores, não tem um tamanho fixo, e por isso são utilizadas quando não conhecemos, a priori, o número de elementos que serão armazenados no conjunto. O código abaixo mostra um exemplo de utilização da classe ArrayList. A classe "Aluno" (omitida no código), foi declarada em outro arquivo e não está sendo exibida aqui.

```
1 import java.util.ArrayList;
2 import java.util.List;
3 import java.util.Scanner;
4
5 public class Programa{
6   public static void main(String args[]){
7     // Configura a leitura de dados do teclado
8     Scanner in = new Scanner( System.in );
9
10    // Cria uma nova lista de alunos
11    List<Aluno> listaAlunos = new ArrayList<>() ;
12
```

```
13      // Lê dados de alunos
14      boolean continuaLendoAluno = true ;
15      do {
16          // Lê dados do aluno
17          System.out.print("Entre com o nome do aluno: ");
18          String nome = in.next() ;
19
20          System.out.print("Entre com as notas do aluno: ");
21          double nota1 = in.nextDouble() ;
22          double nota2 = in.nextDouble() ;
23
24          // Adiciona aluno na lista
25          Aluno aluno = new Aluno(nome, nota1, nota2) ;
26          listaAlunos.add(aluno) ;
27
28          // Pergunta se continua lendo alunos
29          System.out.print("Continua lendo alunos (S/N)? ");
30          String resposta = in.next() ;
31          continuaLendoAluno = resposta.equals("S") ? true : false ;
32
33      } while (continuaLendoAluno) ;
34
35      // Mostra dados da turma
36      System.out.printf("Número de alunos da turma: %d \n", listaAlunos.size());
37      for(Aluno aluno : listaAlunos) {
38          System.out.println("-----------------------");

39          System.out.printf("Nome: %s \n", aluno.getNome());
40          System.out.printf("Notas: %.2f e %.2f \n", aluno.getNota1(), aluno.getNota2());
41      }
42    }
43  }
```

Tendo como referência esse código, que foi apresentado na solução de um dos exercícios resolvidos da nossa disciplina (com adaptações), assinale a alternativa INCORRETA:

- Para percorrer a lista de alunos usamos um tipo especial de "for", como mostrado na linha 37. Nesse loop a variável "aluno" recebe, em cada iteração, um dos alunos armazenados na lista.

- Os métodos "getNome", "getNota1" e "getNota2", chamados nas linhas 39 e 40, foram definidos na classe "Aluno". Como eles estão sendo chamados de dentro de outra classe(no caso, de dentro da classe "Programa"), usando a variável "aluno", podemos supor que os métodos foram definidos com visibilidade "public".

- Na linha 11 estamos criando uma lista de objetos do tipo "Aluno", usando para isso a classe "ArrayList". Para adicionar um objeto do tipo "Aluno" na lista chamamos o método "add", como mostrado na linha 26. Para recuperar o número de objetos armazenados na lista chamamos o método "size", como mostrado na linha 36.

- Estamos usando, nesse exemplo, uma repetição do tipo "do-while" para fazer a leitura dos dados de alunos. Esse tipo de loop é utilizado quando queremos executar um conjunto de instruções pelo menos uma vez. Nesse exemplo, a repetição para depois que forem lidos os dados de 10 alunos.

- Nesse exemplo usamos a classe "Scanner" para fazer a leitura dos dados de um aluno. Isso é feito nas linhas 18, 21 e 22 do código, com a chamada dos métodos "next" e "nextDouble" da classe "Scanner".