Java Programmer – Módulo I

Gabarito

TechnoEdition

Exercícios de fixação

Capítulo 2	Capítulo 4	Capítulo 6	Capítulo 7
1 - A 2 - C 3 - A 4 - E 5 - E	1 - C 2 - B 3 - A 4 - C 5 - B	1 - B 2 - C 3 - D	1 - B 2 - E 3 - B 4 - D 5 - C
Capítulo 8	**Capítulo 9**	**Capítulo 14**	**Capítulo 15**
1 - C 2 - a) V, b) V, c) V, d) F, e) F, f) F, g) V, h) F, i) F, j) V	1 - A 2 - C 3 - C 4 – a) V, b) V, c) F, d) F, e) V, f) V, g) F, h) V, i) F, j) V, k) V, l) V.	1 - B 2 - E 3 - A 4 - D 5 - E	1 – D 2 - D 3 - E 4 - D 5 - B

Exercícios complementares

Capítulo 1
1. As frentes de programação para Java são: Desenvolvimento de aplicativos, desenvolvimento de applets (programas a serem inseridos em páginas WEB) e desenvolvimento de softwares para sistemas embutidos (rodam em automóveis, celulares, fornos microondas, etc.).
2. Java é uma linguagem de programação que possui as seguintes características: orientada a objetos, deve ser compilada e interpretada, é multiplataforma e multithreaded, é robusta e confiável, oferece segurança e apresenta portabilidade.
3. A função do Garbage Collector é fazer o gerenciamento automático da memória, ou seja, tem como responsabilidade alocar memória, anotar quantas referências existem para cada objeto, e limpar os objetos que não possuem referências.
4. Multithreaded é um aplicativo com várias linhas de execução (múltiplos eventos) rodando simultaneamente.
5. JDK contém as ferramentas necessárias para o desenvolvimento de aplicações em Java, enquanto JRE é o ambiente para a execução das aplicações. JDK possui JRE, porém, JRE não possui JDK.

Capítulo 3
1. A diferença é que o operador de atribuição (=) concede um valor a uma variável e o operador relacional de igualdade (==) compara dois valores.
2. Estes operadores têm a função de aumentar ou diminuir exatamente o valor 1 em uma variável.
3. A diferença é que o operador not bit a bit é usado para a inversão de bits, e o operador not lógico é usado para mudar a lógica de uma expressão e só pode ser usado em expressões booleanas.
4. O operador instanceof verifica, em tempo de execução, se um objeto pertence a uma classe.
5. O operador ternário retorna um valor, caso a condição especificada seja avaliada como verdadeira, e outro valor, se for avaliada como falsa.

Capítulo 4

1. Utilizamos comandos de decisão, ou instruções de decisão (if/else e switch/case), para decidirmos qual ação deverá ser efetuada em determinada parte de um programa.

2. Usamos a cláusula else para executar comandos caso a condição testada seja falsa. A cláusula else deverá ser inserida após as instruções da condição verdadeira.

3. As estruturas de repetição disponíveis na linguagem Java são os loops for, while e do/while.

4. O loop for repete a execução de instruções um número específico de vezes. O laço while permite que as instruções sejam executadas até que uma condição seja verdadeira. O comando do/while executa as instruções pelo menos uma vez.

5. A instrução switch tem por objetivo avaliar uma relação de igualdade, e pode substituir a utilização de várias instruções if. Apenas os tipos byte, char, int e short podem ser utilizados na instrução switch.

Capítulo 5

1. A essência de uma linguagem orientada a objetos é a possibilidade de alterar ou substituir partes de um sistema, sem que haja riscos de introdução de erros.

2. As bases para uma linguagem orientada a objetos são: abstração, encapsulamento, herança e polimorfismo.

3. Abstração é um processo por meio do qual filtram-se os fatos relevantes, separando-os dos detalhes menos importantes. Em se tratando do desenvolvimento de sistemas, abstração significa focalizar o objetivo em dois fatores essenciais: o que um objeto é, e o que ele faz.

4. Genericamente, objetos são elementos da natureza aos quais é possível atribuir comportamentos e características. Em uma linguagem orientada a objetos, os objetos são elementos capazes de representar uma entidade que esteja no domínio de interesse do problema analisado, e podem ser concretos ou abstratos.

5. Os atributos referem-se às características dos objetos, já os métodos referem-se às ações dos objetos.

Capítulo 6
1. Classe é uma abstração de um conjunto de objetos, os quais são agrupados por possuírem similaridades em termos de comportamento e características.
2. A vantagem de um código encapsulado está no fato deste ser facilmente acessível, podendo ser usado de maneira segura, sem que haja a necessidade do conhecimento dos detalhes de programação envolvidos.
3. Instanciação é um processo por meio do qual se realiza a cópia de um objeto (classe) existente. Qualquer classe pode ser instanciada, exceto a classe abstract.
4. Caso os atributos de uma classe não possuam valores de inicialização, tais valores serão determinados conforme o tipo de classe no momento de sua instanciação.
5. Acessar uma classe refere-se à capacidade que uma determinada classe deve ter de criar uma instância, estender e acessar certos métodos e variáveis de outra classe.

Capítulo 7
1. Métodos são a implementação de uma operação para uma classe. Podem ser aplicados a várias classes diferentes, mas todos os objetos que estão em uma mesma classe compartilham os métodos aplicados a eles.
2. Quando o argumento é passado por referência, o parâmetro receberá uma referência do argumento original, e não uma cópia do valor, como ocorre quando o argumento é passado por valor. Em Java, a passagem de parâmetros é sempre por valor.
3. Os moderadores de acesso em Java são: public, protected, private, private protected e friendly.
4. Quando precisarmos restringir o acesso a um método.
5. O modificador de método especifica as propriedades de um método, e pode ser abstract, final, native, static ou synchronized.

Capítulo 8
1. O construtor é a estrutura responsável por inicializar o estado interno de um objeto assim que este acabou de ser criado. O método construtor tem o mesmo nome da classe na qual ele está localizado.
2. Quando um construtor não for definido explicitamente para uma dada classe, o compilador do Java se encarrega de criar automaticamente um método construtor padrão.
3. Diante de uma superclasse que não tenha um construtor sem argumentos, devemos inserir um construtor na classe (subclasse).

Capítulo 9
1. Herança é um recurso que possibilita que as classes compartilhem seus atributos e métodos entre si.
2. A superclasse concede suas características a uma outra classe, enquanto a subclasse herda essas características.
3. Com o uso da herança, um objeto pode tornar-se uma instância específica de um caso mais geral, ou seja, um objeto pode herdar os atributos gerais da classe mãe. Ele precisa apenas definir as características que o tornam único na classe à qual pertence.
4. A palavra-chave extends define herança, fazendo com que uma subclasse herde as características de uma superclasse.
5. É-UM é um relacionamento baseado na herança, e o utilizamos para classificar um item, por exemplo Y É-UM animal .

Capítulo 10
1. Arrays são objetos que contêm diversos valores de um mesmo tipo. Esses valores são os elementos desse array e são identificados ou referenciados por um índice, que é um valor inteiro que determina a posição desses elementos dentro do array.
2. Para que um array seja construído, é preciso, primeiramente, criar uma instância utilizando o operador new.
3. A referência a um vetor é feita por meio do índice, que é a posição do elemento dentro do array. Exemplo: *Curso[0] = "Java";*

Capítulo 10
4. O fato de um array poder ser passado e recebido como parâmetro por um método permite que o conteúdo do array possa ser alterado dentro do escopo desse método. Ou seja, os elementos pertencentes ao array, passado como argumento, podem ter seus valores alterados após a chamada ao método.
5. Os arrays podem ser criados com tipos primitivos e com tipos construídos.

Capítulo 11
1. O polimorfismo é definido como um princípio a partir do qual as classes que derivam de uma única superclasse são capazes de invocar os métodos. Estes, embora apresentem a mesma assinatura, comportam-se de forma específica para cada uma das classes derivadas.
2. Dynamic binding, ou run-time binding, é um mecanismo de ligação que deve ser suportado por uma linguagem de programação orientada a objetos, a fim de que o polimorfismo possa ser utilizado. Este mecanismo determina que o método chamado só é definido durante a execução do programa.
3. A redefinição de métodos declarados como private e final não é possível, portanto, seus descendentes são incapazes de invocá-los de maneira polimórfica.
4. O conceito de ligação tardia indica que somente no decorrer da execução do programa, o método, que será indicado, é definido.
5. É um mecanismo utilizado nas situações em que se define o método que será invocado no decorrer do processo de compilação do programa.

Capítulo 12
1. Interfaces são formas de comunicação entre objetos desprovidas de qualquer ação. Pode-se dizer também que as interfaces definem certas funcionalidades.
2. As interfaces são formadas pela declaração de um ou mais métodos, os quais, obrigatoriamente, não possuem corpo.

Capítulo 12
3. Em uma mesma classe podem ser implementadas uma ou mais interfaces, sendo que elas devem estar separadas por vírgulas. Se uma classe implementa uma determinada interface, todos os métodos declarados nessa interface implementada devem ser definidos na respectiva classe. Caso isso não ocorra, será gerado um erro de compilação.
4. As variáveis de referência a objetos podem ser criadas utilizando-se, como tipo, uma interface. Quando criamos uma variável dessa maneira, podemos armazenar nela uma instância de qualquer classe que implemente a interface.
5. Em uma interface, podem existir variáveis inicializadas que funcionam como constantes, mas não podem existir variáveis de instância.

Capítulo 13
1. Os pacotes são utilizados em Java quando temos a necessidade de reutilizar as classes que foram escritas ou definidas em pacotes distintos. Dessa forma, caso seja necessário utilizar recursos de outras classes, estas não precisarão ser novamente construídas, mas apenas importadas para os programas.
2. Por convenção, os nomes dos pacotes devem ser escritos com a letra inicial em caixa baixa.
3. O uso de public para membros de classes que utilizam o comando package permite que esses membros possam ser referenciados por outras classes que não estejam no mesmo pacote.
4. O comando package é utilizado para especificar o pacote ao qual a classe que está sendo definida pertencerá.
5. O comando import permite importar uma classe para um programa, a fim de que essa classe possa ser referenciada fora de seu pacote.

Capítulo 14
1. Visando a utilização da memória de forma eficiente, a Java Virtual Machine mantém uma área reservada, chamada de pool constante de strings, que é um local verificado pelo compilador com a finalidade de identificar se há coincidências de Strings literais. Dessa forma, não será criado um novo objeto String e sim será direcionada uma referência ao novo valor literal para a String já existente.
2. É uma das características importantes de objetos String, que garante que esses objetos jamais serão alterados. Porém, sua variável de referência poderá ser alterada sem problemas.
3. Uma classe String é marcada como final para que os métodos do objeto String não sejam modificados de forma alguma, assegurando, assim, sua inalterabilidade.
4. A classe Math é utilizada com a finalidade de efetuar operações matemáticas comuns. Tendo em vista que todos os métodos dessa classe são definidos como static, estes não precisam ser instanciados a fim de que possam ser utilizados.
5. As classes Wrapper da linguagem Java são responsáveis por fornecer, essencialmente, um conjunto de funções que são utilitárias aos tipos primitivos, e também funcionam como um mecanismo capaz de agrupar os valores primitivos em um objeto.

Capítulo 15
1. O pacote Swing é utilizado para construir as interfaces gráficas de usuário(GUIs) na linguagem Java. A maioria das classes utilizadas nesse pacote é escrita na própria linguagem Java.
2. O pacote Swing permite que os componentes gráficos sejam exibidos mais rapidamente, os botões e os rótulos podem ter imagens adicionadas a eles e, ainda, possui novos componentes gráficos com relação ao AWT.
3. Os pacotes que devem ser importados são import Java.awt.; e import Java.awt.event.*;.*
4. Os contêineres referem-se ao local em que serão adicionados os componentes visualizados na tela.
5. Os gerenciadores de layout FlowLayout, BorderLayout e GridLayout são responsáveis pela disposição e pelo tamanho dos elementos na tela ou dentro dos contêineres.

Capítulo 16
1. Os Applets são pequenos programas ou aplicações executados em browsers.
2. Para visualizar um Applet, é preciso criar um arquivo do tipo HTML ou HTM logo depois que o Applet for compilado. Em seguida, devemos criar uma tag dentro do arquivo recém-criado para que o Applet seja chamado. Feito isso, usamos o browser para abrir o arquivo, que carregará automaticamente o Applet.
3. Um Applet contém quatro métodos principais: init(), start(), stop() e destroy(). O uso desses quatro métodos dentro de um Applet não é obrigatório.
4. Uma das características dos Applets é o suporte para som e imagem. Os Applets aceitam a implementação de imagens do tipo JPEG ou GIF (que também pode ser animado) e arquivos de som do tipo MIDI, WAV, AU e AIFF.
5. O Java Plug-in precisa estar instalado na máquina cliente para que seja possível usar componentes Swing em applets. Para isso, usamos o pacote AWT com a finalidade de construir componentes gráficos na tela dos browsers.

Exercícios de laboratório

Capítulo 2

Laboratório 1

O erro está na linha 4: faltou o ponto-e-vírgula.
Solução: String cidade, estado;

Laboratório 2

A linha 8 está concatenando as variáveis valor1 e valor2, ao invés de somá-las.
Solução: System.out.println("Resultado da soma = "+ (valor1 + valor2));

Capítulo 3

Laboratório 1

a) verdadeiro
b) verdadeiro
c) falso
d) verdadeiro
e) falso
f) verdadeiro

Laboratório 2

```
public class ExercicioCap3_3_Resp {
   public static void main(String args[]) {
         int A = 3, B = 7, C = 2, D = 5;
         System.out.println(A >= B);
         System.out.println((A * B == C) || (A > D));
         System.out.println((B < C) && (B > D));
         System.out.println((A > D) || (A * B == C));
         System.out.println((A + B < D) && (D > C));
         System.out.println(D % B < 2);
   }
}
```

Laboratório 4

Alternativa correta: A

Capítulo 3

Laboratório 5

```
class ExercicioCap3_5
{
      public static void main(String args[])
      {
         int VlrPag = 2000;
         System.out.println (VlrPag >= 0 && VlrPag <=
   1000
         ? VlrPag :
         VlrPag > 1000 && VlrPag <= 3000 ?
         VlrPag * 0.95 :
         VlrPag * 0.90);
      }
}
```

Capítulo 4

Laboratório 1

Resposta: 676,72

```
public class ExercicioCap4_2 {
   public static void main(String args[]) {

   double Salario = 4000, IR = 0;
         if (Salario >= 0 && Salario <= 1058)

   IR = 0;
         else if (Salario > 1058 && Salario < 2115)
               IR = Salario * 0.15 - 158.70;
         else if (Salario >= 2115)

   IR = Salario * 0.275 - 423.08;
         System.out.println("O IR eh: " + IR);
   }
}
```

Capítulo 4

Laboratório 2

```
public class ExercicioCap4_3 {
  public static void main(String args[]) {
        for (int x = 1; x < 11; x++)
              for (int y = 1; y < 11; y++)
                    System.out.println(x + " x " + y + "
= " + x * y);
  }
}
```

Laboratório 3

```
public class ExercicioCap4_4 {
  public static void main(String args[]) {
        for (int x = 0; x < 100; ++x) {
              if (x == 56) {

  System.out.println("\tAgora x vale.........." + x);
                    continue;
              }
              if ((x % 2) == 0) {
                    System.out.println("\tValor de x eh
par = " + x);
              }
        }
  }
}
```

Capítulo 6

Laboratório 1

```
public class Cliente {

   // atributos
   String nomeCliente;
   String telefone;
   String endereco;

   // métodos
   void impressao() {
         System.out.println("Nome Cliente: " +
nomeCliente);
         System.out.println("Telefone: " + telefone);
         System.out.println("Endereco: " + endereco);
         System.out.
println("----------------------------");

   }
}
public class Cadastrar {
   public static void main(String args[]) {

   Cliente vanessa = new Cliente();
         vanessa.nomeCliente = "Vanessa";
         vanessa.telefone = "(011) 3282-5555";
         vanessa.endereco = "Av. Paulista, 2000";
         vanessa.impressao();
         Cliente marcia = new Cliente();
         marcia.nomeCliente = "Marcia";
         marcia.telefone = "(011) 3366-7070";
         marcia.endereco = "Rua Maranhao, 328";
         marcia.impressao();
         Cliente ernesto = new Cliente();
         ernesto.nomeCliente = "Ernesto";
         ernesto.telefone = "(011) 5555-6677";
         ernesto.endereco = "Rua Alemanha, 727";

   ernesto.impressao();
   }
}
```

Capítulo 7

Laboratório 1

```
public class Calculadora {
   static double somar(double n1, double n2) {
        return (n1 + n2);
   }

   static double subtrair(double n1, double n2) {
        return (n1 - n2);
   }

   static double multiplicar(double n1, double n2) {
        return (n1 * n2);
   }

   static double dividir(double n1, double n2) {

   return (n1 / n2);
   }
}


public class UsaCalculadora {
   public static void main(String args[]) {
        double x = 80;
        double y = 20;
        System.out.println("Somar " + x + " e " + y);

   System.out.println(Calculadora.somar(x, y));

   System.out.println("Subtrair " + y + " de " + x);

   System.out.println(Calculadora.subtrair(x, y));
        System.out.println("Multiplicar " + x + " por "
+ y);
        System.out.println(Calculadora.multiplicar(x,
y));
        System.out.println("Dividir " + x + " por " +
y);
        System.out.println(Calculadora.dividir(x, y));
   }
}
```

Capítulo 10

Laboratório 1

```
public class Notas {
   static public void main(String args[]) {
         double nota[][] = new double[30][5];
         int i;

   int j;
         nota[0][0] = 7.5;
         nota[0][1] = 10;
         nota[0][2] = 8;
         nota[1][0] = 5;
         nota[1][1] = 7.8;
         nota[1][2] = 9;
         nota[2][0] = 10;
         nota[2][1] = 9.5;
         nota[2][2] = 8;
         nota[3][0] = 3;
         nota[3][1] = 4.5;
         nota[3][2] = 6;
         nota[4][0] = 10;
         nota[4][1] = 8.8;
         nota[4][2] = 6.6;
         for (i = 0; i < 5; i++)
               for (j = 0; j < 3; j++)
                     System.out.println("aluno " + i + "
nota " + j + " - "
                                       + nota[i][j]);
   }
}
```

Capítulo 10

Laboratório 2

```
public class Bola {

   double raio;
   String tipo;

}
public class Lab2 {
   public static void main(String args[]) {

   Bola bolas[] = new Bola[2];
         bolas[0] = new Bola();
         bolas[0].tipo = "futebol";
         bolas[0].raio = 7.5;
         bolas[1] = new Bola();
         bolas[1].tipo = "ping-pong";
         bolas[1].raio = 1.5;
         System.out.println(bolas[0].tipo);
         System.out.println(bolas[0].raio);
         System.out.println();
         System.out.println(bolas[1].tipo);
         System.out.println(bolas[1].raio);
   }
}
```

Capítulo 12

Laboratório 1A

```
public interface Treinamento {
   static final String ESCOLA = "Impacta Tecnologia";
   static final String FONE = "11 3285.5566";

   public int getPreco(int modulo);
}
```

Capítulo 12

Laboratório 1B

```
public class Office implements Treinamento {
   public int getPreco(int modulo) {
         if (modulo == 1)
               return (300);
         else
               return (500);
   }
}

public class DesenvolvimentoWeb implements Treinamento
{
   public int getPreco(int modulo) {
         if (modulo == 1)
               return (400);
         else
               return (700);
   }
}
```

Capítulo 12

Laboratório 1C

```
public class Cap12Lab1 {
  static Office excel;
  static DesenvolvimentoWeb flash;

  public static void main(String arg[]) {

  inicio();

  treinaInfo(excel, 1);

  treinaInfo(flash, 1);
  }

  public static void inicio() {

  excel = new Office();

  flash = new DesenvolvimentoWeb();
  }

  public static void treinaInfo(Treinamento item, int
tipo) {

  System.out.println();

  System.out.println("Escola: " + item.ESCOLA + "\
nFone: " + item.FONE);

  System.out.println("Este Treinamento custa: " +
item.getPreco(tipo));
  }
}
```

Capítulo 13

Laboratório 1

```
package estoque;

public class Caneta {

   public void mostra() {
         System.out.println("Canetas estão em Estoque");
   }
}

import estoque.Caneta;

public class Cap13_Lab {
   public static void main(String arg[]) {

   Caneta c = new Caneta();
         c.mostra();
   }
}
```

Capítulo 14

Laboratório 1

```
public class ExercicioMetodosEncadeados {
   public static void main(String[] args) {

   String frase = "      java é divertido", resultado =
(frase.toUpperCase())

   .trim();

   System.out.println(resultado + "!!!");
   }
}
```