INSTITUTO FEDERAL DE MINAS GERAIS
CAMPUS SÃO JOÃO EVANGELISTA
SISTEMAS DE INFORMAÇÃO

VALDIR DE SOUZA CARVALHO NETO

Linguagem de Programação I - SI 241

Aula 2 - Exercícios de Aprendizagem

São João Evangelista
2025

SUMÁRIO

LISTA DE FIGURAS

1. **Exercício 1**

Explique o processo híbrido de tradução de código-fonte feito pelo Java, evidenciando a fase de compilação, interpretação e o papel da JRE neste processo.

Resposta: O código-fonte, escrito em Java (com a extensão *.java*), é compilado pelo Javac (o compilador Java). O resultado dessa compilação não é um código de máquina, mas sim um código intermediário chamado Bytecode, com extensão *.class*. Esse Bytecode não é específico para um sistema operacional, o que permite que o mesmo código seja executado em diferentes plataformas. O Bytecode é então interpretado. Para isso, a JRE, que contém a JVM, entra em ação. A JVM traduz o Bytecode para o código de máquina do sistema operacional específico onde o programa está sendo executado. A JRE, com a sua JVM, é a responsável por essa etapa de interpretação e execução do Bytecode em cada sistema operacional. Graças a essa arquitetura, o Java segue o princípio "Write Once, Run Anywhere!".

2. **Exercício 2**

Qual a principal vantagem do Java em relação a linguagens puramente compiladas? E a(s) desvantagem(ns)?

Resposta: A principal vantagem do Java em relação a linguagens puramente compiladas, como o C ou o C++, é a sua portabilidade. Como o Java gera um Bytecode intermediário, ele pode ser executado em qualquer sistema operacional que tenha uma JRE instalada, sem a necessidade de recompilar o código para cada plataforma. Isso contrasta com as linguagens puramente compiladas, que geram um executável nativo específico para cada sistema, exigindo a compilação em cada plataforma. Uma desvantagem é que o Java, por ter essa etapa de interpretação, pode apresentar um desempenho um pouco inferior em comparação com linguagens puramente compiladas.

3. **Exercício 3**

Explique como compilar e executar um programa em Java via linha de comando?

Resposta: Para compilar um programa, o primeiro passo é abrir o terminal e ir até a pasta onde está o arquivo *.java,* ou ir no gerenciador de arquivos e clicar com o botão direito do mouse e escolher a opção de abrir a pasta no terminal. Depois, use o comando *javac* seguido do nome do arquivo, por exemplo: *javac NomeArquivo.java*. Se correr tudo certo, o compilador vai gerar um arquivo *.class*. Para executar, use o comando *java* seguido do nome da classe sem incluir o tipo de arquivo, por exemplo: *java NomeArquivo*. Assim, a JRE irá carregar e executar o Bytecode.

4. **Exercício 4**

Defina o que são classes e objetos em Java. Explique a diferença entre eles e exemplifique com uma situação do mundo real.

Resposta: Classe é um molde ou um projeto que define as características (atributos) e os comportamentos (métodos) de algo. Ela é um conceito abstrato, uma "planta" que serve de base para a criação de objetos. Já um objeto, é uma instância concreta de uma classe. Ele representa uma entidade real, com valores específicos para os atributos e a capacidade de executar os métodos definidos na classe.

A classe é a ideia, e o objeto é a manifestação dessa ideia. Um bom exemplo seria uma classe *Carro*, em que a classe é a planta que define que um carro tem atributos como *cor* e *modelo*, e métodos como *acelerar()* e *frear()*. Já um objeto,seria o "Meu Carro", uma instância específica da classe *Carro*. Ele pode ter a *cor* "azul", o *modelo* "Sedan" e pode executar o método *acelerar()*.

5. **Exercício 5**

Em Programação Orientada a Objetos (POO), o recurso que permite a uma classe controlar o acesso a seus atributos e métodos, escondendo detalhes internos e expondo apenas o necessário, é chamado de encapsulamento.
Em Java, qual é o papel dos modificadores de visibilidade public e private? Dê exemplos práticos de como eles contribuem para o encapsulamento.

Resposta: O modificador *public*, permite que o atributo ou método seja acessado de qualquer parte do programa. Seu uso é ideal para métodos que precisam ser acessados externamente, como *getNome()* ou *setNome()*, que retorna e lê um nome, respectivamente. Já o modificador *private*, restringe o acesso a um atributo ou método apenas à própria classe onde foi declarado. Seu uso nos atributos de uma classe, como *nome* e *preço* no *Produto*, é fundamental para o encapsulamento, pois impede que outras classes os alterem diretamente, garantindo a integridade dos dados.

Um exemplo prático seria uma classe *Produto* com os atributos *nome* e *preço*, que ao serem declarados como privados, impede que outra classe simplesmente altere o preço para um valor negativo, evitando isso através de uma validação no método *setPreco()*, que é *public*.

6. **Exercício 6**

Em Java, quando os atributos de uma classe são declarados como private, eles não podem ser acessados diretamente por outras classes.
Para permitir o acesso e a modificação desses valores de forma controlada, é necessário criar métodos específicos.

Crie uma classe chamada Produto com os atributos privados nome (String) e preco (double). Depois, implemente métodos que permitam:
− definir o nome de um produto setNome;
− definir o preço de um produto setPreco;
− consultar o nome (getNome) e o preço (getPreco) já armazenados.
Por fim, crie um objeto da classe Produto e utilize esses métodos para alterar e exibir os valores.

Código:
Arquivo Main:

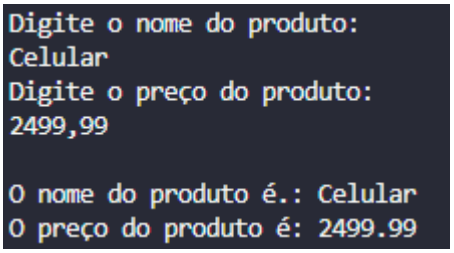
```
import java.util.Scanner;

public class Main2 {
    public static void main(String[] args) {
        Produto produto = new Produto();
        Scanner cin = new Scanner(System.in);
        String nome;
        double preco;

        System.out.println("Digite o nome do produto:");
        nome = cin.nextLine();
        System.out.println("Digite o preço do produto:");
        preco = cin.nextDouble();

        produto.setNome(nome);
        produto.setPreco(preco);

        System.out.println("\nO nome do produto é.: " +
produto.getNome());
        System.out.println("O preço do produto é: " +
produto.getPreco());

        cin.close();
    }
}
```

Arquivo da Classe:

```java
public class Produto {
    private String nome;
    private double preco;

    public void setNome(String nome) {
        this.nome = nome;
    }

    public void setPreco(double preco) {
        this.preco = preco;
    }

    public String getNome() {
        return nome;
    }

    public double getPreco() {
        return preco;
    }
}
```

Imagens:

Figura 1. Captura de tela Atividade 06



Fonte: Captura de tela retirada pelo autor em 07 de outubro de 2025.

7. Exercício 7

Crie uma classe chamada Aluno com os atributos privados nome (String) e três notas (nota1, nota2, nota3, do tipo double).

Implemente métodos que permitam:

− Armazenar as notas dos alunos;

− Calcular a média das notas;

− Verificar, usando uma estrutura de decisão, se o aluno está aprovado (média maior ou igual a 60) ou reprovado.

No método principal, crie um objeto da classe Aluno, cadastre suas notas e exiba a média e a situação final.

Código:

Arquivo Main:

```java
import java.util.Scanner;

public class Main3 {
    public static void main(String[] args) {
        Scanner cin = new Scanner(System.in);
        Aluno aluno = new Aluno();
        String nome;
        double nota1, nota2, nota3;

        System.out.println("Digite o nome do aluno:");
        nome = cin.nextLine();
        System.out.println("Digite a 1ª nota do aluno:");
        nota1 = cin.nextDouble();
        System.out.println("Digite a 2ª nota do aluno:");
        nota2 = cin.nextDouble();
        System.out.println("Digite a 3ª nota do aluno:");
        nota3 = cin.nextDouble();

        aluno.setNome(nome);
        aluno.setNota1(nota1);
        aluno.setNota2(nota2);
        aluno.setNota3(nota3);

        System.out.println("Nome do aluno.: " + aluno.getNome());
        System.out.println("Média do aluno: " + aluno.getMedia());
        if (aluno.getSituacaoFinal()) {
            System.out.println("Situação final: Aprovado!");
        } else {
            System.out.println("Situação final: Reprovado!");
        }
        cin.close();
    }
}
```

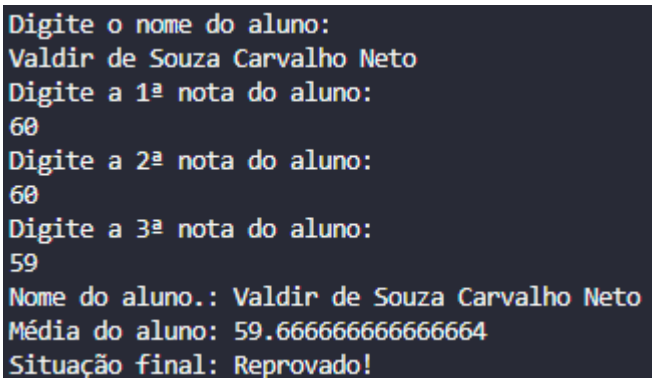
Arquivo da Classe:

```java
public class Aluno {
```

```
    private String nome;
    private double nota1, nota2, nota3;

    public void setNome(String nome) {
        this.nome = nome;
    }

    public void setNota1(double nota1) {
        this.nota1 = nota1;
    }

    public void setNota2(double nota2) {
        this.nota2 = nota2;
    }

    public void setNota3(double nota3) {
        this.nota3 = nota3;
    }

    public String getNome() {
        return nome;
    }

    public double getMedia() {
        double media = (nota1 + nota2 + nota3) / 3;
        return media;
    }

    public boolean getSituacaoFinal() {
        double media = (nota1 + nota2 + nota3) / 3;

        if (media >= 60) {
            return true;
        } else {
            return false;
        }
    }
}
```

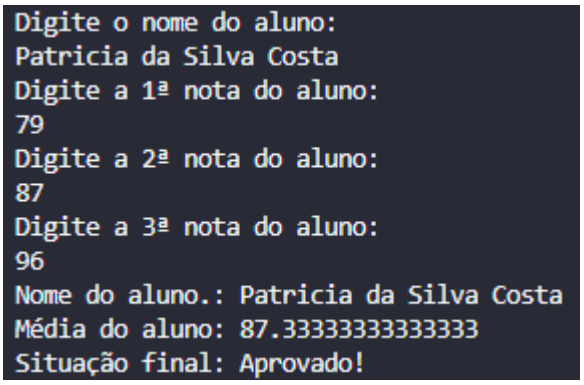
Imagens:

Figura 2. Captura de tela Atividade 07



Fonte: Captura de tela retirada pelo autor em 07 de outubro de 2025.

Figura 3. Captura de tela Atividade 07



Fonte: Captura de tela retirada pelo autor em 07 de outubro de 2025.

8. Exercício 8

Crie uma classe chamada Ponto com os atributos privados x e y (ambos do tipo double), representando um ponto no plano cartesiano.

Implemente métodos que permitam:

− Definir e acessar os valores de x e y;

− Calcular a distância entre dois pontos, utilizando a fórmula da distância entre dois pontos no plano:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

− Exibir as coordenadas de um ponto.

No método principal, crie dois objetos da classe Ponto, atribua valores às suas coordenadas, exiba estas coordenadas e a distância entre eles.

Código:

Arquivo Main:

```java
import java.util.Scanner;

public class Main4 {
    public static void main(String[] args) {
        Scanner cin = new Scanner(System.in);
        Ponto ponto1 = new Ponto();
        Ponto ponto2 = new Ponto();
        double x1, y1, x2, y2;

        System.out.println("Digite o valor de x1:");
        x1 = cin.nextDouble();
        System.out.println("Digite o valor de y1:");
        y1 = cin.nextDouble();
        System.out.println("Digite o valor de x2:");
        x2 = cin.nextDouble();
        System.out.println("Digite o valor de y2:");
        y2 = cin.nextDouble();

        ponto1.setX(x1);
        ponto1.setY(y1);
        ponto2.setX(x2);
        ponto2.setY(y2);

        System.out.println("Ponto 1: x é igual a " + ponto1.getX() + "
e y é igual a " + ponto1.getY());
        System.out.println("Ponto 2: x é igual a " + ponto2.getX() + "
e y é igual a " + ponto2.getY());

        System.out.println("A distância entre os pontos é de: " +
ponto1.calcularDistancia(ponto2));

        cin.close();
    }
}
```

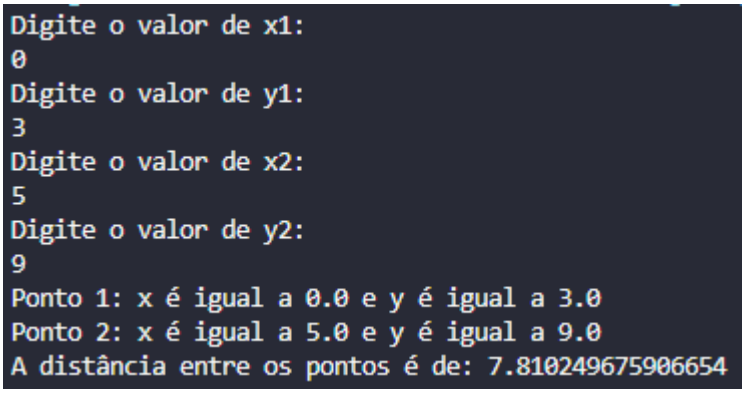
Arquivo da Classe:

```java
public class Ponto {
    private double x, y;

    public void setX(double x) {
        this.x = x;
```

```java
    }

    public void setY(double y) {
        this.y = y;
    }

    public double getX() {
        return x;
    }

    public double getY() {
        return y;
    }

    public double calcularDistancia(Ponto outroPonto) {
        double deltaX = this.x - outroPonto.getX();
        double deltaY = this.y - outroPonto.getY();
        return Math.sqrt(Math.pow(deltaX, 2) + Math.pow(deltaY, 2));
    }
}
```

Imagens:

Figura 4. Captura de tela Atividade 08



Fonte: Captura de tela retirada pelo autor em 07 de outubro de 2025.