Linguagens Comerciais de Programação

Prof. Lucas Guerreiro
guerreiroluc@gmail.com

EMENTA

(tópicos que caracterizam as unidades dos programas de ensino)

1. Recursos computacionais para implementação de sistemas comerciais.
2. Ferramentas computacionais disponíveis
3. Características de uma Linguagem Orientada à Aplicações Comerciais.
4. Implementação de um sistema comercial.

BIBLIOGRAFIA BÁSICA

BARNES, D.J.; KOLLING, M. Programação Orientada a Objetos com Java. 4 ed. Pearson, 2009.

DEITEL, H.M. – Java : Como Programar – 8. ed. – São Paulo: Makron Books, 2010, 1145 p.

MOREIRA NETO, O. – Entendendo e Dominando o Java – 2. Ed. – São Paulo:Digerati Books, 2007, 416 p.

SANTOS, R. - Introdução a Programação Orientada a Objetos usando JAVA – Elsevier, 2003.

REESE, G.- Database programming with JDBC and Java - O'Reilly, 1997, 224 p.

CONTEÚDO PROGRAMÁTICO

(título e discriminação das unidades)

1. Recursos computacionais para implementação de sistemas comerciais: Características, Histórico e Ferramentas Computacionais Disponíveis
2. Estudo detalhado de uma linguagem de programação comercial orientada a objetos.
   - 2.1 A tecnologia Java
   - 2.2. Ambientes de Desenvolvimento de Aplicações em Java
   - 2.3. Conceitos Básicos da Linguagem Java
   - 2.4 Conceitos da programação orientada a objetos : abstração, classe e objeto, herança, polimorfismo e interface
   - 2.5 Conceitos de vetores e coleções
   - 2.6 Manipulação de Strings
   - 2.7 Tratamento de Exceções
   - 2.8. Arquivos e Fluxos
3. Conectividade JDBC
   - 3.1 Conexão com o Banco de Dados
   - 3.2 Manipulação dos dados
   - 3.3 DataSource
4. Componentes Visuais (Swing)
5. Implementação de um sistema comercial em Java

Dinâmica da Disciplina

NF = 0.4 * Prova + 0.4 * Projeto + 0.2 * Media_Trabalhos

```
if (NF < 5){
   float nota = prova_recuperacao();
} else {
   System.out.println("Parabéns!");
}
```

Planejamento das atividades		
Aula	**Data**	**Atividade**
1	29/04	Apresentação da disciplina. Conceitos de Java e de Orientação a Objetos. Exercícios.
2	06/05	Herança. Polimorfismo. Interface. Abstração. Modificadores de Acesso. Pacotes.
3	13/05	Exercícios
4	20/05	Conceitos de Vetores e Coleções
5	27/05	Manipulação de Strings
6*	28/05	Exercícios
7	03/06	Arquivos e Exceções
8	10/06	Trabalho Prático (início)
9	24/06	Conectividade JDBC - Parte 1
10	01/07	Conectividade JDBC - Parte 2. Componentes Visuais - Parte 1
11*	02/07	Exercícios
12	08/07	Componentes Visuais - Parte 2
13*	11/07	Entrega Trabalho. Revisão.
14	15/07	Avaliação
15	22/07	Exame Final

Aula 01

Objetivo: conceitos de sistemas computacionais e linguagens utilizadas. Conceitos de OO. Fundamentos de Java. Práticas em Java.

Sistemas Computacionais

- Automatização de tarefas
- Auxiliar realização de tarefas humanas
- Facilitar tarefas repetitivas
- Componentes fundamentais: hardware, software, usuários, informações, ferramentas, **linguagens**
- Tendência atual: sistemas distribuídos, multiplataforma, informação instantânea

Paradigmas de programação

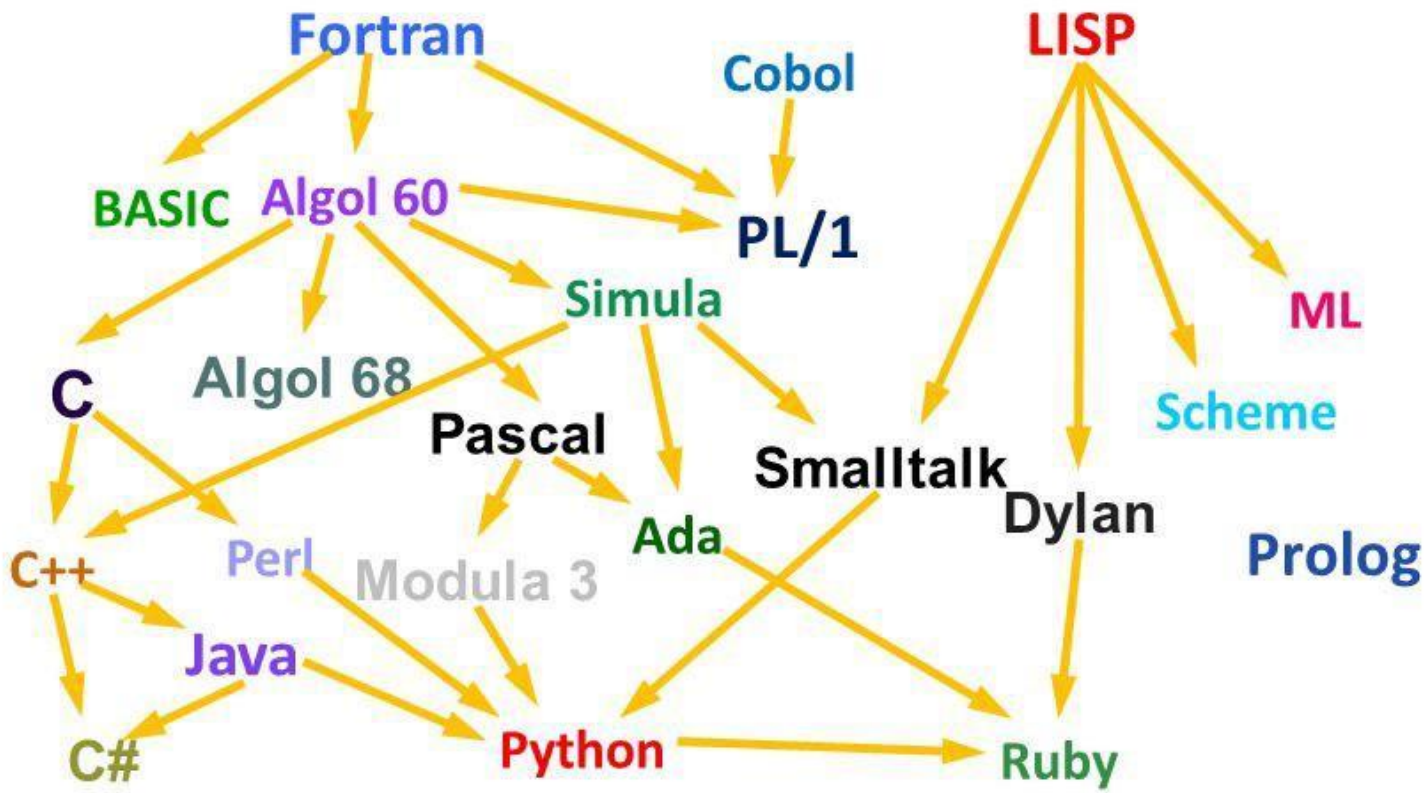
- Paradigma Imperativo (Procedural): passo-a-passo do que será feito de forma estruturada. Exemplos: Pascal e Fortran
- Paradigma Orientado a Eventos: ações decorrentes de eventos dos usuários. Exemplos: Delphi e VB
- Paradigma Funcional: problemas orientados a funções matemáticas. Exemplos: LISP e Haskell
- **Paradigma Orientado a Objetos**: abstrações de problemas, sendo replicável a diferentes aplicações. Exemplos: Python e **Java**

Year: 2021 | Quarter: 4

# Ranking	Programming Language	Percentage (YoY Change)	YoY Trend
1	Python	17.926% (+1.438%)	☐
2	JavaScript	14.058% (-4.714%)	☐
3	Java	12.208% (+0.662%)	
4	TypeScript	8.472% (+1.818%)	☐
5	Go	8.161% (+0.027%)	☐
6	C++	6.670% (-0.331%)	☐
7	Ruby	6.165% (-0.783%)	☐
8	PHP	5.252% (-0.322%)	
9	C#	3.372% (-0.301%)	
10	C	3.150% (+0.023%)	
11	Nix	2.420% (+2.408%)	☐
12	Shell	2.184% (+0.153%)	
13	Scala	2.047% (+0.005%)	☐
14	Kotlin	1.028% (+0.277%)	☐
15	Rust	0.694% (-0.204%)	☐
16	Dart	0.694% (-0.388%)	☐
17	Swift	0.648% (+0.029%)	☐
18	Groovy	0.354% (-0.027%)	☐
19	Lean	0.323% (+0.000%)	☐
20	Elixir	0.311% (-0.035%)	☐

A family tree of languages

Some of the 2400 + programming languages



Java

- Início do projeto em 1991
- Linguagem popular
- Orientada a objetos
- Portátil - “escreva uma vez, execute em qualquer lugar”

Java SE: contém os recursos necessários para desenvolver aplicativos de desktop e servidor.

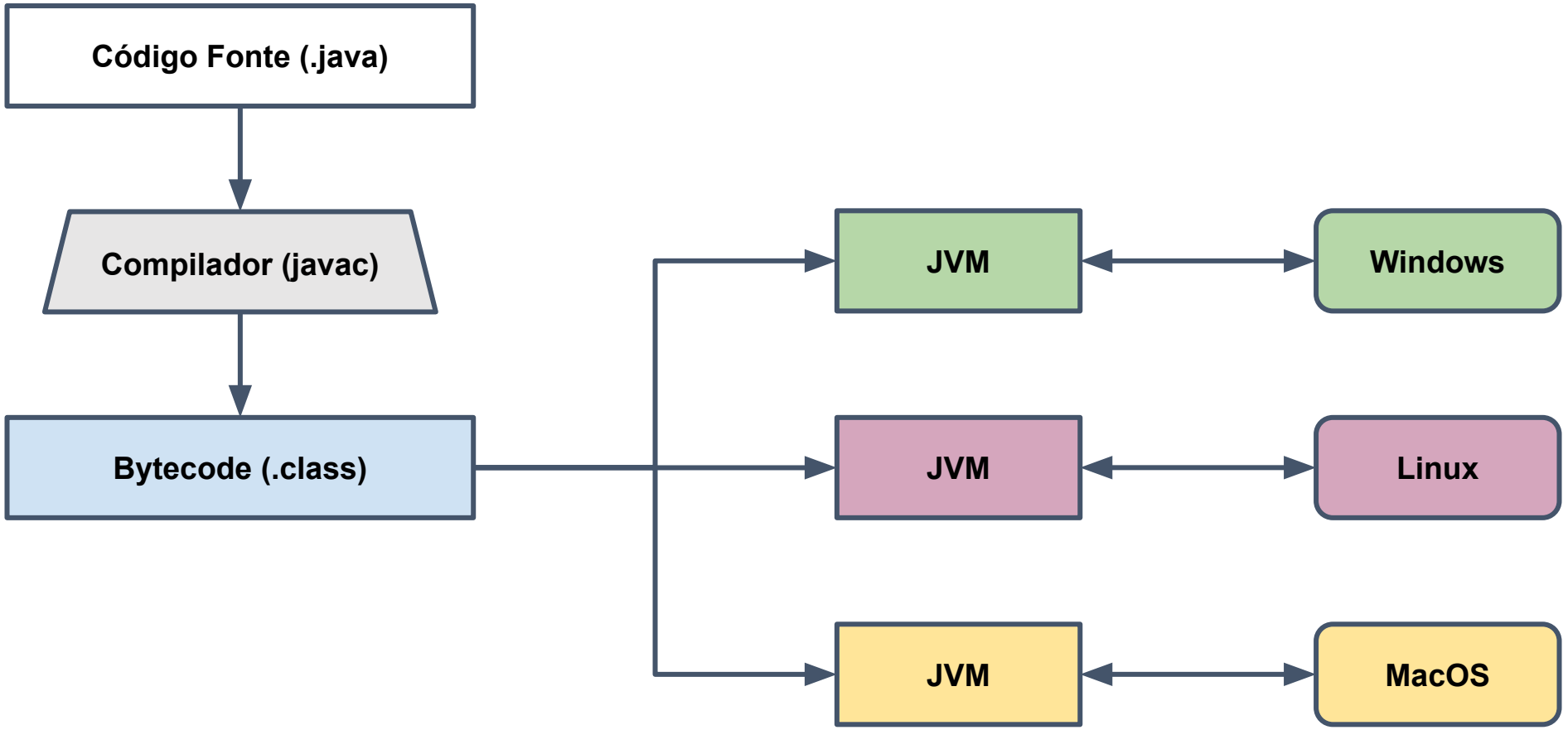
Java EE: desenvolver aplicativos em rede distribuída e em grande escala e também aplicativos baseados na web.

Java ME: desenvolvimento de aplicativos para dispositivos embarcados com recursos limitados

Java

- **JVM (Java Virtual Machine)**: responsável por transformar os bytecodes do programa em linguagem de máquina na arquitetura correspondente. Invocada pelo comando java
- **JDK (Java Development Kit)**: ambiente para desenvolvedores. Possui todas as ferramentas necessárias para compilar, debuggar e executar programas Java
- **JRE (Java Runtime Environment)**: ambiente responsável por executar o programa da JVM. Ambiente necessário para os usuários finais.

Java


Código Fonte (.java)
Compilador (javac)
Bytecode (.class)
JVM
JVM
JVM
Windows
Linux
MacOS

Orientação a Objetos

- ➔ Classe: abstração do problema e onde estão as definições das características e comportamentos gerais das entidades envolvidas
- ➔ Objeto: componente oriundo da classe que representa uma das entidades

“Instanciar um objeto x da classe X”

Orientação a Objetos (OO)

Os 4 pilares da OO:

- Abstração
- Encapsulamento
- Herança
- Polimorfismo

Abstração

- Demais classes/objetos não precisam de detalhes da implementação de métodos
- “Esconder” informações inerentes à classe
- Limitar escopo de métodos sempre que possível
- Uso de Interfaces e Classes abstratas (veremos mais pra frente)

Encapsulamento

- Limitar acesso e visibilidade de atributos e comportamentos
- Apenas aqueles objetos que necessitam terão acesso às características
- Implementação de modificadores de acesso
- Uso de getters/setters

Herança

- Capacidade de passar as características gerais de uma classe para outra, a qual pode alterar algumas delas
- Reuso de código
- Definições genéricas dos comportamentos na Super Classe
- Em Java: “extends”

Polimorfismo

- Utiliza de diferentes tipos de dados (e tamanhos) para a implementação do “mesmo” método
- Ato de sobrescrever um método.
- Ações diferentes a partir da entrada. dentro do mesmo nome de método
- Sobrecarga (mesma classe). Sobreposição (subclasse/super classe)

```
public static void soma(int a, int b) {
        System.out.println("Somando números");
        System.out.println(a + b);
    }

public static void soma(String a, String b) {
    System.out.println("Concatenando Strings");
    System.out.println(a + "," + b);
}

public static void main(String args[]) {
    soma(5,20);
    soma("a","b");
}
```

Conceitos Gerais

- Atributos: "variáveis" da classe
- Métodos: as funções implementadas na classe
- Métodos especiais:
  - Construtor: método executado quando o objeto é instanciado
  - Destrutor: método executado quando o objeto é liberado da memória

Prática

Instalação e Configuração JDK

Baixa e instalar JDK a partir do site da Oracle (https://www.oracle.com/java/technologies/downloads/)

Configurar as variáveis de ambiente PATH, JAVA_HOME e CLASSPATH:

JAVA_HOME-> `"caminho_de_instalacao_da_JDK"`

PATH-> `%JAVA_HOME%/bin`

CLASSPATH-> `%JAVA_HOME%/lib`

Git

Criar conta no GitHub: https://github.com/

Aplicar comandos de acordo com a necessidade:

```
git init
git clone usuario@repositorio_remoto
git add arquivo ou git add *
git commit -m "Mensagem"
git push origin master
```

Fazer em casa:

https://rogerdudler.github.io/git-guide/

https://learngitbranching.js.org/

Programando em Java

Via linha de comando:

salvar arquivo com `arquivo.java`

gerar bytecode com `javac arquivo.java`

executar com `java arquivo`

Via IDE

Executar interativamente dentro da IDE (Eclipse, NetBeans, VS Code, etc)

Olá mundo!

```
/* Comentário longo
* esse programa exibe a frase 'Olá mundo!'
fim de comentário */
// declarando uma classe pública
public class OlaMundo {
   // método principal - público, estático, sem retorno, argumentos como parâmetros
   public static void main(String[] args) {
      System.out.print("Olá, "); // exibindo Olá em tela sem nova linha
      System.out.println("mundo!"); // exibindo o resto da frase, quebrando linha ao final
   }
}
```

Olá *arg*!

```
public class OlaArg {
   public static void main(String[] args) {
      System.out.print("Olá, " + args[0] + "!");
   }
}
```

```
java OlaArg mundo
```

Olá método!

```
public class OlaMetodo {
   static void cumprimentar(){
      System.out.print("Olá, mundo!");
   }
   public static void main(String[] args) {
      cumprimentar();
   }
}
```

Olá parâmetro!

```
public class OlaParametro {
   static void cumprimentar(String nome){
      System.out.print("Olá, " + nome + "!");
   }
   public static void main(String[] args) {
      cumprimentar("mundo");
   }
}
```

Olá Objeto!

```
public class OlaObjeto {
   // por utilizar a chamada a partir de objeto,
   // não precisamos do static
   void cumprimentar(String nome){
      System.out.print("Olá " + nome + "!");
   }
   public static void main(String[] args) {
      OlaObjeto obj = new OlaObjeto();
      obj.cumprimentar("mundo");
   }
}
```

Olá atributo!

```
public class OlaAtributo {
   String palavra = "mundo";
   void cumprimentar(){
      System.out.print("Olá, " + this.palavra + "!");
   }
   public static void main(String[] args) {
      OlaAtributo obj = new OlaAtributo();
      obj.cumprimentar();
   }
}
```

Comandos e sintaxe

Tipos primitivos

- `boolean;`
- `byte;`
- `char;`
- `short;`
- `int;`
- `long;`
- `float;`
- `double.`

`String ->` não é tipo primitivo, mas usamos esta classe para manipular textos.

Comandos e sintaxe

Entrada

instanciar objeto de Scanner

```
Scanner scan = new Scanner(System.in);
String palavra = scan.nextLine();
```

scan.nextInt(), scan.nextFloat, ...

Lembrar de "encerrar" a leitura:
`scan.close()`

Saída

instancia objeto de System.out

```
System.out.print()
```

Comandos e sintaxe

Laços condicionais

- `if (condicao) {comandos;} else {comandos;}`
- `switch (expressao) {case CONST: comando;}`

Laços repetição

- `for (init; condicao; increm) {comandos;}`
- `while (condicao) {comandos;}`
- `do {comandos;} while (condição);`

Exercício 01

Ler o nome e valor de um produto

Exercício 01

```
import java.util.Scanner;
public class Exercicio01 {
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        System.out.print("Digite o nome do produto: ");
        String produto = scan.nextLine();
        scan.close();
        System.out.printf("Digite o valor de compra do produto %s: ",
        produto);
        float valor = scan.nextFloat();
        scan.close();
        System.out.printf("Produto %s custa %f .", produto, valor);
    }
}
```

Exercício 02

Ler o nome e valor de um produto e notificar se este custar mais que 100 e quanto excede.

Exercício 02

```
Scanner scan = new Scanner(System.in);
System.out.print("Digite o nome do produto: ");
String produto = scan.nextLine();
System.out.printf("Digite o valor de compra do produto %s: ", produto);
float valor = scan.nextFloat();
if (valor > 100)
   System.out.printf("Produto %s custa %f , excedendo em %f o limite de 100.", produto, valor, valor - 100);
else
   System.out.printf("Produto %s custa %f.", produto, valor);
```

Exercício 03

Ler a categoria do produto e identificar ações para bebida e comida.

Exercício 03

```
Scanner scan = new Scanner(System.in);
System.out.print("Digite o nome do produto: ");
String produto = scan.nextLine();
System.out.printf("Digite a categoria do produto %s: ", produto);
String categoria = scan.nextLine();
switch (categoria){
    case "BEBIDA": System.out.printf("%s é de se beber ", produto); break;
    case "COMIDA": System.out.printf("%s é de se comer ", produto); break;
    default: System.out.printf("%s sem ações de categoria.", produto);
}
```

Exercício 04

Solicitar valor do produto até que este seja menor que 100. (while)

Exercício 04

```
Scanner scan = new Scanner(System.in);
System.out.print("Digite o nome do produto: ");
String produto = scan.nextLine();
System.out.printf("Digite o valor de compra do produto %s: ", produto);
float valor = scan.nextFloat();
while (valor > 100){
   System.out.printf("Produto excede o valor limite de 100. Digite novo valor: ");
   valor = scan.nextFloat();
}
System.out.printf("Produto %s cadastrado com valor %f .", produto, valor);
```

Exercício 05

Solicitar valor do produto até que este seja menor que 100. (do/while)

Exercício 05

```
Scanner scan = new Scanner(System.in);
System.out.print("Digite o nome do produto: ");
String produto = scan.nextLine();
float valor;
do {
   System.out.printf("Digite o valor de compra do produto (< 100) %s: ", produto);
   valor = scan.nextFloat();
while (valor > 100);
System.out.printf("Produto %s cadastrado com valor %f .", produto, valor);
```

Exercício 06

Receber 5 produtos (sem armazená-los separadamente)

Exercício 06

```
Scanner scan = new Scanner(System.in);
String produto;
float valor;
for (int i = 0; i < 5; i++){
    System.out.print("Digite o nome do produto: ");
    produto = scan.nextLine();
    System.out.printf("Digite o valor de compra do produto %s: ", produto);
    valor = scan.nextFloat();
    scan.nextLine();
    System.out.printf("%s cadastrado com valor %f", produto, valor);
    System.out.println();
}
```

Exercício 07

Calcular valor final do produto, adicionando uma margem de 15%

Exercício 07

```
static float calculaPreco(float valor_bruto){
    return (float) valor_bruto*1.15;
}
public static void main(String[] args){
    System.out.print("Digite o nome do produto: ");
    String produto = scan.nextLine();
    System.out.printf("Digite o valor de compra do produto %s: ", produto);
    float valor = scan.nextFloat();
    float valor_final = calculaPreco(valor);
    System.out.printf("Valor final é de %f.", valor_final);
}
```

Exercício 07 - opção 2

```
String nome;
float valor;
void setNomeProduto(String nome){
    this.nome = nome;
}
void setValor(float valor){
    this.valor = valor;
}
float getValorFinal(){
    return (float) (this.valor * 1.15);
}
```

```
public static void main(String[] args){
        Scanner scan = new Scanner(System.in);
        System.out.print("Digite o nome do produto: ");
        String nome_produto = scan.nextLine();
        System.out.printf("Digite o valor de compra do
produto %s: ", nome_produto);
        float valor = scan.nextFloat();
        Exercicio07b produto = new Exercicio07b();
        produto.setNomeProduto(nome_produto);
        produto.setValor(valor);
        float valor_final = produto.getValorFinal();
        System.out.printf("Valor final é de %f.",
valor_final);
        scan.close();
    }
```

Exercício 08

Cadastro completo do produto e armazenar valor de venda. Margem de 15% para produtos de até 100 reais, de 12% para produtos entre 100 e 200 reais e 10% para produtos acima de 200 reais.

Exercício 08

```
void cadastrarProduto(){
    System.out.print("Digite o nome do produto: ");
    String produto = scan.nextLine();
    System.out.printf("Digite a categoria do produto %s: ", produto);
    String categoria = scan.nextLine();
    System.out.printf("Digite o valor de compra do produto %s: ", produto);
    float valor = scan.nextFloat();
    this.nome = produto;
    this.categoria = categoria;
    this.valor = valor;
}
void definirPreco(){
    if (this.valor < 100)
        this.preco = this.valor * 1.15;
    else if (this.valor < 200)
        this.preco = this.valor * 1.12;
    else
        this.preco = this.valor * 1.1;
}
```

Exercício 08

```
float getPreco(){
    return this.preco;
}
public static void main(String[] args){
    Exercicio08 produto = new Exercicio08();
    produto.cadastrarProduto();
}
```

Exercício 09

Receber itens de mercado (nome, quantidade comprada e valor) até que o produto seja "SAIR" e acumular o valor de compra. Neste exemplo ainda não precisamos acumular produtos ou uma lista deles, apenas o valor final. Quando sair, exibir o valor total da compra.

Exercício 10

Calcular valor a descontar no IRPF a partir do salário mensal com base na tabela. Observação: para cálculo do salário mensal o programa deve receber o valor-hora e a quantidade de horas trabalhadas semanalmente. Considerar o mês com 4 semanas.

Base de cálculo (R$)	Alíquota (%)	Parcela a deduzir do IRPF (R$)
Até 1.903,98	-	-
De 1.903,99 até 2.826,65	7,5	142,80
De 2.826,66 até 3.751,05	15	354,80
De 3.751,06 até 4.664,68	22,5	636,13
Acima de 4.664,68	27,5	869,36