Introdução à Linguagem de Programação JAVA

1. O que é um Programa?

É um conjunto de instruções escritas em uma determinada linguagem de programação para descrever como uma tarefa será executada pelo computador. Esse conjunto de instruções é chamado de código fonte, e possibilita a realização de tarefas específicas, tais como cálculos, manipulação de dados, controle de dispositivos, etc.

O ciclo básico de execução de um programa é conhecido como Modelo de "Entrada-Processamento-Saída" ("Input-Process-Output):

- Entrada: fase na qual o programa recebe dados por meio do usuário (via teclado, mouse, etc.), outro sistema, arquivos ou sensores. Exemplo: O usuário digita dois números e uma operação matemática em uma calculadora ("5 + 7").
- Processamento: fase na qual o programa executa as instruções para realizar cálculos ou operações lógicas sobre os dados de entrada. Exemplo: O programa realiza a soma dos números inseridos.
- Saída: fase na qual o programa exibe o resultado do processamento. Exemplo: A calculadora mostra o resultado da soma de "5 + 7", que é "12".

O programa coordena todo o processo, especificando como os dados são coletados, quais operações devem ser realizadas durante o processamento, e como (e onde) o resultado será exibido ou entregue.

2. O que é o Java?

Java é uma linguagem de programação orientada a objetos e também uma plataforma. Portanto, tenha em mente que Java não é somente uma linguagem. Ela é uma plataforma completa, que fornece serviços como segurança, portabilidade para os diversos sistemas operacionais e gerenciamento automático da memória. Além disso, ela possui uma vasta biblioteca de classes.

Principais características do Java:

- Linguagem orientada a objetos com sintaxe amigável.
- É Case-sensitive.
- Possibilita a portabilidade ("*write-once / run-anywhere*") – permitindo que os programas Java possam ser executados em qualquer plataforma que suporte a Java Virtual Machine (JVM), sem a necessidade de recompilação do código.
- Implementa segurança – os programas são executados na JVM, que isola o código Java do sistema subjacente, proporcionando um ambiente mais seguro.
- Implementa Multiprogramação (Multithreading) – suporte nativo para desenvolvimento de programas que executam múltiplas tarefas de maneira simultânea.
- Possui um conjunto de bibliotecas de classes, conhecidas como APIs (*Application Programming Interfaces*) Java.
- Não define apenas uma linguagem, mas também uma plataforma, que inclui a JVM (*Java Virtual Machine* – Máquina Virtual Java) e as APIs Java.

3. Breve Histórico

Em 1991 um grupo de engenheiros da empresa *Sun Microsystems*, liderados por James Gosling (considerado o pai do Java), iniciou um projeto cujo objetivo era criar uma linguagem para um dispositivo portátil com tecnologia *touchscreen* que iria controlar um ambiente doméstico (luzes, tvs, videocassetes, liquidificadores, aparelhos da tv a cabo, etc.). Buscava-se uma linguagem sem associação com a arquitetura de um determinado fabricante e também capaz de rodar em dispositivos com pouca memória. Tais requisitos levaram a criação de uma linguagem que gerava um código intermediário para uma máquina virtual. Esse código intermediário poderia ser executado em qualquer máquina, bastava para isso a utilização de um interpretador correto.

A linguagem foi batizada com o nome *Oak* (carvalho, uma árvore). Entretanto, após verificar que *Oak* era uma marca registrada pela empresa *Oak Technology*, os engenheiros precisaram definir um novo nome, e decidiram por utilizar Java. A busca por compradores para essa tecnologia ocorreu sem sucesso até 1994.

Com o crescimento da Internet, a *Sun Microsystems* decidiu criar um *browser* (navegador) utilizando Java. Esse *browser* tinha a capacidade de executar pequenas aplicações em Java (*Applets*) dentro das páginas Web. Foi o início da popularização do Java.

A partir daí, com as novas versões, defeitos foram corrigidos, novas funcionalidades foram adicionadas e o Java seduziu os programadores com sua sintaxe amigável, as características de orientação a objetos, o gerenciamento de memória e a portabilidade.

Veja a seguir algumas características das versões disponibilizadas:

Versão	**Características**
JDK beta (1995)	- Lançamento do Java.
JDK 1.02 (1996)	- 250 classes. - Lento e cheio de bugs. - Os *Applets* eram a grande sensação.
JDK 1.1 (1997)	- 500 classes. - Um pouco mais rápido. Tornou-se bastante popular.
Java 2 (J2SE 1.2 a J2SE 1.4) (1998 a 2002)	- 2300 classes. - Muito mais rápido. - Disponibilizado em três versões: - *Standard* (J2SE): base da plataforma, incluindo o ambiente de execução e as bibliotecas básicas para o desenvolvimento de software para *desktop* e servidores. É muito utilizado para o aprendizado da linguagem. - *Enterprise* (J2EE): ambiente para o desenvolvimento e execução de aplicações distribuídas para soluções corporativas baseadas na internet e intranet. - *Micro* (J2ME): ambiente para desenvolvimento e execução de aplicativos em dispositivos móveis e embarcados (celulares, impressoras, reprodutores de discos *blu-ray*, *set-top boxes*, dispositivos de mídia digital, etc.). - Tornou-se a linguagem das aplicações web. - 4 milhões de desenvolvedores (2003).
Java J2SE 5.0 (2004)	- 3500 classes. - Mais de 1000 novas classes foram adicionadas. - Características populares de outras linguagens foram incorporadas. - As versões Java foram renomeadas para JSE, JEE e JME. - 2004: Lançamento do IRPF (Imposto de Renda Pessoa Física) em Java.
Java SE 6 (2006)	- 3700 classes. - Melhorias na Performance. - Java torna-se *Open Source*. - 2009: Oracle anuncia compra da Sun Microsystems por US$ 7,4 bilhões.
Java SE 7 (2011)	- 4400 classes. - Melhorias na linguagem. - Novo sistema de arquivos (Java I/O).

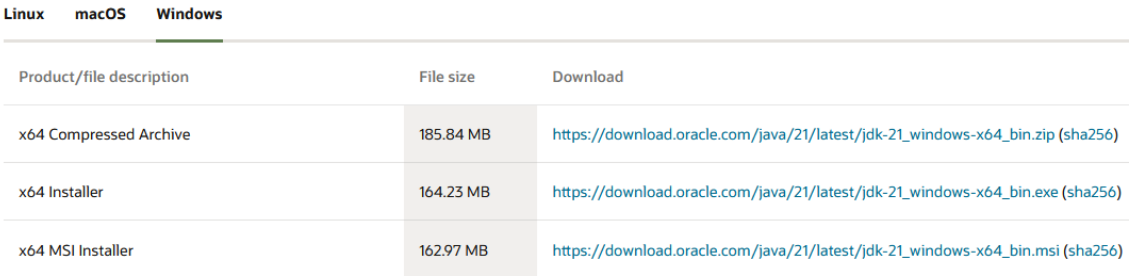
Java SE 8 (LTS) (2014)	- 4900 classes. - Java 8 LTS (*Long Term Support*) - Expressões lambda. - Nova API para manipulação de datas.
Java SE 9 (2017)	- 5300 classes. - 3 anos após Java 8. - Mais de 90 funcionalidades (*features*) implementadas. - Modularização do JDK (java.base é o módulo principal – root).
Java SE 10 a Java SE 15 (2018 a 2020)	- 5600 classes. - Novas versões lançadas semestralmente (março e setembro). - Java 10: novas APIs para Coleções. - Java 11 LTS (*Long Term Support*): código Java EE é separado do repositório e se transforma no projeto Jakarta EE. - 2020: 25º aniversário do Java • 9,1 milhão de desenvolvedores. • 45 bilhões de JVM ativas.
Java SE 16 (2021)	- 5800 classes. - Lançamento em março de 2021. - Após 25 anos, Java continua: • Flexível (se adapta a um cenário tecnológico em constante mudança, mantendo-se independente de plataforma). • Confiável (mantem compatibilidade com versões anteriores).
Java SE 17 (LTS) (2021)	- 5900 classes. - Java 17 LTS (*Long Term Support*) lançado em setembro de 2021. - Implementa 14 funcionalidades (*features*). - Oracle ajustou os termos de licença, permitindo que empresas utilizem a versão por 3 anos sem custos. - 2645 problemas foram corrigidos.
Java SE 18 (2022)	- 6000 classes. - UTF-8 como padrão para APIs que dependem do charset padrão. - Inclusão de Simple Web Server, um servidor HTTP para testes e desenvolvimento local.
Java SE 19 (2022)	- 6050 classes. - Melhoria na execução de threads virtuais. - Lançamento de APIs para gerenciamento mais robusto e eficiente de threads em paralelo e melhoria na manipulação de dados imutáveis.
Java SE 20 (2023)	- 6100 classes - Possibilidade de armazenamento imutável de valores compartilhados por várias threads. - Refinamentos no Coletor de Lixo. - Melhorias no desempenho do JDK.
Java SE 21 (LTS) (2023)	- 6200 classes. - Java 21 LTS lançado em setembro de 2023. - Virtual Threads (GA): forma mais eficiente de gerenciar milhares de threads com menos overhead. - Sequenced Collections: novas interfaces para coleções ordenadas, aperfeiçoando o tratamento de sequências em listas, conjuntos e mapas. - Correções e melhorias na API de coleções, serialização e desserialização de objetos.

4. Instalação do JDK (Java Devolpment Kit) no Windows

Passo 1: Baixar o JDK 21

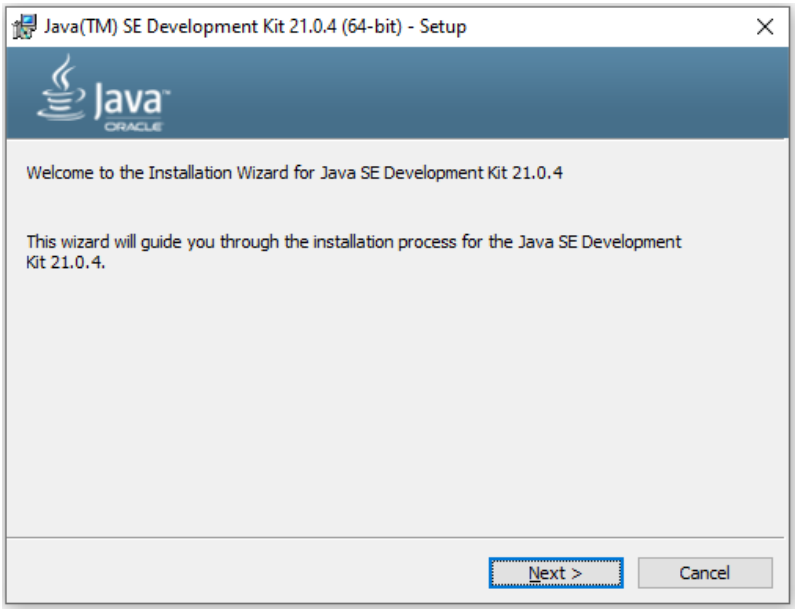
1) Acesse o site oficial da Oracle: https://www.oracle.com/java/technologies/downloads/#jdk21-windows
2) Selecione a aba da versão do JDK 21.

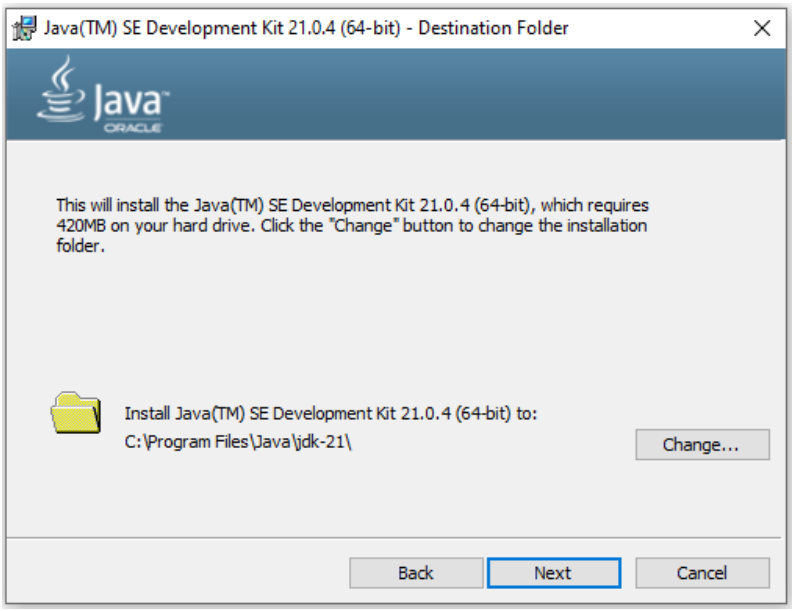
3) Procure pela versão do JDK 21 compatível com o seu sistema operacional Windows Clique no link de download (você pode optar pelo instalador .exe ou .msi) e salve o arquivo de instalação em seu computador.



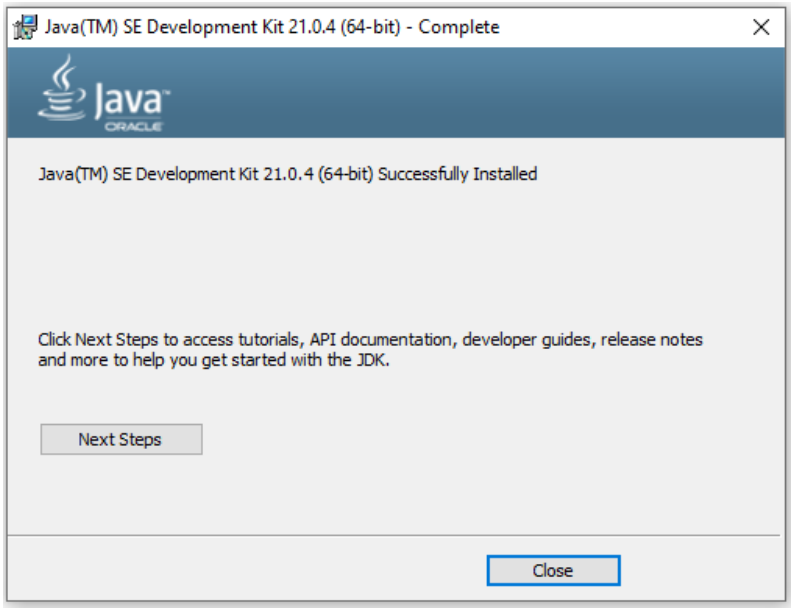
Passo 2: Instalar o JDK 21

1) Abra o instalador baixado (jdk-21_windows-x64_bin.exe).
2) Siga os passos indicados pelo assistente de instalação. O caminho de instalação padrão será algo semelhante a C:\Program Files\Java\jdk-21.





3) Conclua a instalação.



É possível acessar tutoriais, documentação e guias de desenvolvimento por meio do link: https://docs.oracle.com/en/java/javase/21/index.html

Passo 3: Verificar a Instalação

1) Abra o Prompt de Comando (CMD).
2) Digite o seguinte comando para verificar a versão do JDK instalado: java -version

```
C:\Users\Edilson>java -version
java version "21.0.4" 2024-07-16 LTS
Java(TM) SE Runtime Environment (build 21.0.4+8-LTS-274)
Java HotSpot(TM) 64-Bit Server VM (build 21.0.4+8-LTS-274, mixed mode, sharing)
```

Se a versão não for exibida, será preciso configurar as Variáveis de Ambiente. Abra o Explorador de Arquivos e clique com o botão direito em "Este Computador" (ou "Meu Computador"), depois clique em Propriedades.

1) Clique em Configurações Avançadas do Sistema no painel esquerdo.

2) Na aba Avançado, clique em Variáveis de Ambiente.
3) Na seção Variáveis do Sistema, localize a variável chamada Path e clique em Editar.
4) Clique em Novo e adicione o caminho para o diretório bin do JDK 21 (algo semelhante a C:\Program Files\Java\jdk-21\bin).
5) Clique em OK para salvar e fechar as janelas.
6) Verifique a instalação novamente.

5. Como o Java funciona?

1) Escreva o código-fonte em um arquivo com extensão .java.

Exemplo: PrimeiroPrograma.java. O código-fonte é escrito utilizando a sintaxe da linguagem Java.

```
public class PrimeiroPrograma {
  public static void main(String[] args) {
    System.out.println("Primeiro programa em Java!");
  }
}
```

2) Compile o código-fonte utilizando o compilador Java (javac).

```
javac PrimeiroPrograma.java
```

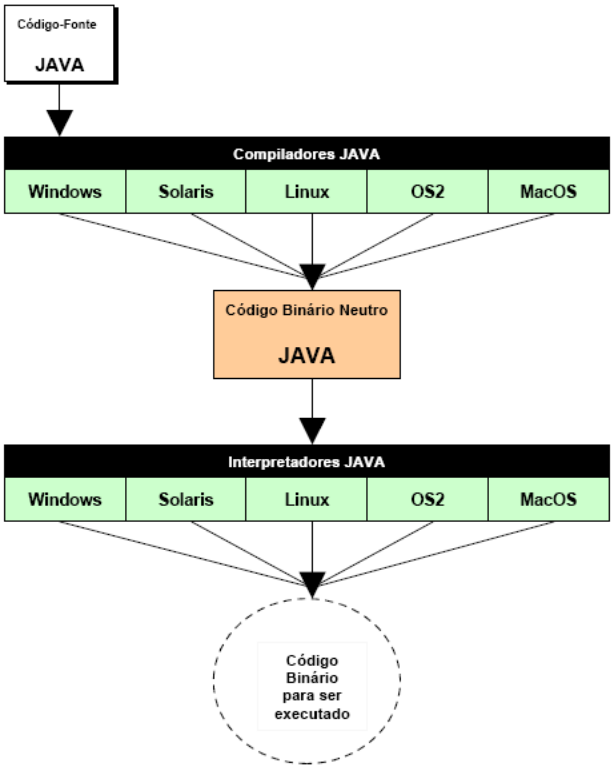
O compilador verifica se há erros no código. Se não houver erros, ele gera um arquivo com a extensão .class (Exemplo: Programa.class), que contém o bytecode Java.

- **bytecode** é um conjunto de instruções intermediárias que são independentes de plataforma, ou seja, podem ser executadas em qualquer sistema operacional, desde que ele possua uma Máquina Virtual Java (JVM) instalada. Ele é mais eficiente do que o código-fonte original, mas ainda precisa ser interpretado pela JVM.

3) Execute o programa utilizando a Máquina Virtual Java (JVM).

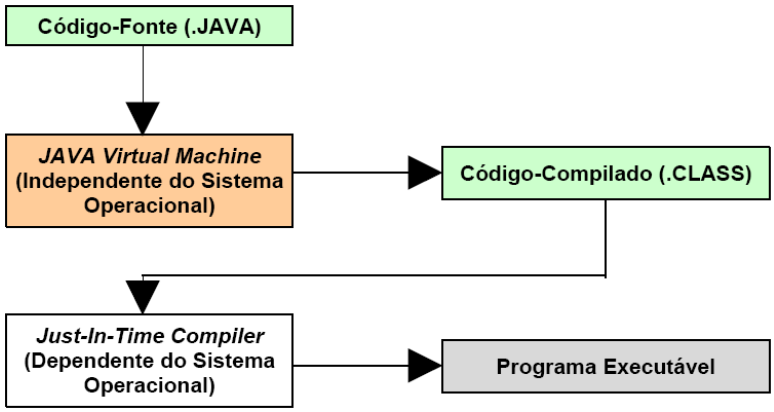
```
java PrimeiroPrograma.java
```

A JVM interpreta e executa o bytecode contido no arquivo com extensão “class”. Ela traduz o bytecode em instruções que o sistema operacional do computador consegue entender, permitindo que o programa seja executado.



6. O que é a Java Virtual Machine?

A JVM (*Java Virtual Machine* – Máquina Virtual Java) é um software que carrega e executa os aplicativos Java, convertendo os *bytecodes* em código de máquina. A JVM também é responsável pelo gerenciamento dos aplicativos que estão sendo executados (segurança, gerenciamento de memória, etc).



7. Comentários detalhados sobre o código do primeiro programa em Java

O primeiro exemplo de programa em Java exibe uma linha de texto. Ele ilustra a estrutura básica de um programa com vários recursos da linguagem Java.

```
1   /*
2    Programa que exibe a mensagem "Primeiro programa em Java!"
3   */
4
5   public class PrimeiroPrograma {
6     // Método principal, que inicia a execução de um programa em Java
7     public static void main(String[] args) {
8       System.out.println("Primeiro programa em Java!");
9     }
10  }
```

As linhas 1, 2 e 3 formam um bloco de linhas de comentário, que inicia com /* e se encerra com */. Você pode inserir comentários para documentar os programas e facilitar a leitura e a manutenção dos mesmos. O compilador Java ignora os comentários. Outra opção é inserir o comentário de uma única linha, tal como apresentado na linha 6, que inicia com // e inclui uma descrição sobre o método main.

A linha 4 está em branco. Linhas em branco também são ignoradas pelo compilador e são utilizadas para aprimorar a legibilidade do programa.

As palavras-chave, ou palavras reservadas, são restritas para uso da linguagem Java e são sempre escritas com todas as letras em minúsculo.

A linha 5 inicia uma declaração de classe para a classe PrimeiroPrograma. Todo programa em Java possui pelo menos uma classe definida pelo programador. A palavra-chave class é utilizada para declarar uma nova classe e é imediatamente seguida pelo nome da classe (PrimeiroPrograma). Por convenção, os nomes de classes iniciam com uma letra maiúscula, tal como PrimeiroPrograma. O nome de uma classe é um identificador.

Importante: Os identificadores podem começar com uma letra, um sublinhado (_) ou um cifrão ($). Eles não podem começar com um número e não tem espaços.

Importante: Java faz distinção entre letras maiúsculas e minúsculas, ou seja, as letras maiúsculas e letras minúsculas são diferentes. Isso é chamado de sensitive case. O uso incorreto de letras maiúsculas e minúsculas para um identificador leva a um erro de compilação.

Além disso, a declaração da classe inicia com a palavra-chave public. Por enquanto, utilizaremos apenas classes public.

Importante: uma classe public deve ser inserida em um arquivo com o mesmo nome da classe (tanto na ortografia quanto no uso de maiúsculas e minúsculas) seguido pela extensão .java. O não seguimento desta regra leva a um erro de compilação. Neste nosso primeiro programa, a classe PrimeiroPrograma deve ser armazenada em um arquivo chamado PrimeiroPrograma.java.

No final da linha 5, uma chave esquerda { inicia o corpo da declaração de classe. Uma chave direita } correspondente, presente na linha 10, é utilizada para terminar a declaração de classe.

Dica: Ao digitar uma chave de abertura de bloco (chave esquerda, {), imediatamente digite a chave de fechamento (chave direita, }). Essa prática ajuda a evitar erros causados pela ausência das chaves.

É importante observar o recuo do texto das linhas 6 a 9. Esse recuo é uma das convenções de espaçamento da linguagem Java e é uma boa prática de programação. Essa prática enfatiza a estrutura da declaração e facilita a leitura do código-fonte.

A linha 7 é o ponto de partida da execução do programa Java. Os parênteses depois do identificador main indicam um bloco de construção de programa chamado método. Em uma aplicação Java, um dos métodos deve ser chamado main e deve ter a sintaxe como apresentado na linha 7. Nas próximas aulas serão fornecidos mais detalhes sobre os conceitos de método e de classe. Por enquanto, basta que você utilize a sintaxe deste método.

A chave esquerda do final da linha 7 é utilizada para indicar o início do corpo de método. Uma chave direita correspondente é utilizada para terminar o corpo do método.

A linha 8 instrui o computador a realizar uma ação, ou seja, exibir uma string (cadeia de caracteres) contida entre as aspas duplas, mas sem exibir as aspas. Os caracteres de espaço em branco em uma string não são ignorados pelo compilador, isto é, são exibidos ao usuário.

O objeto System.out é chamado de objeto de saída padrão (o dispositivo de saída padrão é o monitor do computador). Este objeto permite que aplicativos Java exibam strings na janela de comando a partir da qual o aplicativo Java executa.

O método System.out.println exibe ou imprime uma linha de texto na janela de comando. A string entre parênteses na linha 8 é o argumento para o método.

Após digitar o texto, salve o arquivo com a extensão “java”. Certifique-se de que o nome do arquivo é idêntico ao nome que aparece após o texto “public class”, ou seja, “PrimeiroPrograma”. Isso é necessário porque em Java o nome do arquivo deve ser idêntico ao nome da classe principal.

8. Exercícios

8.1. Elabore uma pesquisa sobre os seguintes conceitos: JRE, JDK e JVM. Indique o significado de cada sigla, suas características e possibilidades de utilização. Adicione as referências utilizadas para pesquisa.

8.2. Escreva um programa que leia uma determinada temperatura em graus Fahrenheit (F), calcule e mostre a conversão para graus Celsius (C). A fórmula de conversão é: C = (F – 32).

8.3. Escreva um programa que efetue a leitura de duas notas, calcule e mostre a média ponderada dessas notas. A primeira nota tem peso 1 e a segunda nota tem peso 2.

8.4. Crie um programa que calcule o fatorial de um número fornecido pelo usuário.

8.5. Crie um programa que gerencie um estacionamento rotativo de carros. O programa deve armazenar a descrição do carro, a placa, o horário de entrada e o horário de saída (despreze os minutos). O estacionamento cobra X reais pela primeira hora de permanência com o automóvel e X/3 pelas demais horas. Além disso, é fornecido um desconto para o pagamento de acordo com a tabela abaixo:

Valor	Desconto (%)
Até R$ 20 (inclusive)	5
Entre R$ 20 e R$ 50 (inclusive)	10
Acima de R$ 50	20

O programa deve exibir um relatório contendo as seguintes informações:

a) Tipo do carro.
b) Placa.
c) Hora da Entrada.
d) Hora da Saída.
e) Valor Pago.

8.6. Implemente um programa que realize a busca binária em um array ordenado.