



INTRODUÇÃO À LINGUAGEM JAVA

Acesso nos computadores


Caso seu usuário ainda não esteja disponível

Usuário: **t-317112**

Senha: gJgYUSS1


Caso seja necessário, poderá entrar em contato com a unidade por meio da seção: Formulário de Contato

Importante: Caso você seja do programa Empreenda Rápido Super MEI ou de turmas exclusivas de empresas, desconsidere esta mensagem.

 **Rede Educacional** X

Para utilizar os computadores de laboratórios e salas de aulas do SENAC você deve utilizar o login e a senha é a mesma que você utilizou para acessar a área exclusiva (Portal do Aluno ou Portal do Professor).

Blackboard Mobile Learn



Vinicius Ledesma

Trabalho com tecnologia desde 2000,
atualmente atuo como Tech lead na
Santander F1RST Tecnologia.



- Nome
- Experiencia com desenvolvimento
- Ocupação atual



- Duração do curso: Início 14/01/2023 a 25/03/2023
- Carga horária: 40 horas
- Horário: 8h às 12h
- Frequência: 75% (limite de faltas: 10 horas)
- Intervalos de 15 min
- Celulares: silencioso, caso precise atender, saia rapidamente da sala
- Não é permitido se alimentar na sala de aula / laboratório

O que vamos ver

- Introdução: o que é Java, edições da plataforma, histórico, onde usar, mercado e perfil do profissional, referências e documentação;
- Compilação e execução de programa Java, JVM;
- JDK, ambiente de desenvolvimento, configurações;
- Regras (sintaxe), convenções e boas práticas;
- Conceitos iniciais e prática para programação orientada a objetos: definição, princípios, onde é usada, histórico, classes e objetos, atributos e métodos;
- Depuração de programa (debug);
- Lógica de programação em Java: tipos de dados, estruturas de controle, operadores, casting;

O que vamos ver

- Classes, métodos e construtor;
- Strings e arrays: manipulação;
- Encapsulamento, modificadores, getters e setters;
- Herança, superclasse, subclasse, acoplamento, anotação `@override`, palavras reservadas `this` e `super` (análise: herança X composição), polimorfismo;
- Pacotes, API Date/Time (Java 8);
- Interface e threads;
- Collections Framework;
- Erros e exceções.

Mercado

- <https://www.vagas.com.br/mapa-de-carreiras/cargos/desenvolvedor-java/0>
- <https://insights.stackoverflow.com/survey/2021#overview>
- <https://www.apinfo2.com/apinfo/informacao/p12sal-br.cfm>

Senac

Java x Javascript

- JavaScript é uma outra linguagem, orientada a modelos e o java é orientado a objetos, e que tem um nome similar simplesmente porque compartilha os modelos de dados e objetos de Java.
- JavaScript é utilizado para acrescentar processamento local às páginas HTML, e interage diretamente com as diretivas da página. Pode ser considerado como uma espécie de HTML procedural. Convém ressaltar que JavaScript é muito poderosa e pode substituir Java em muitas aplicações.

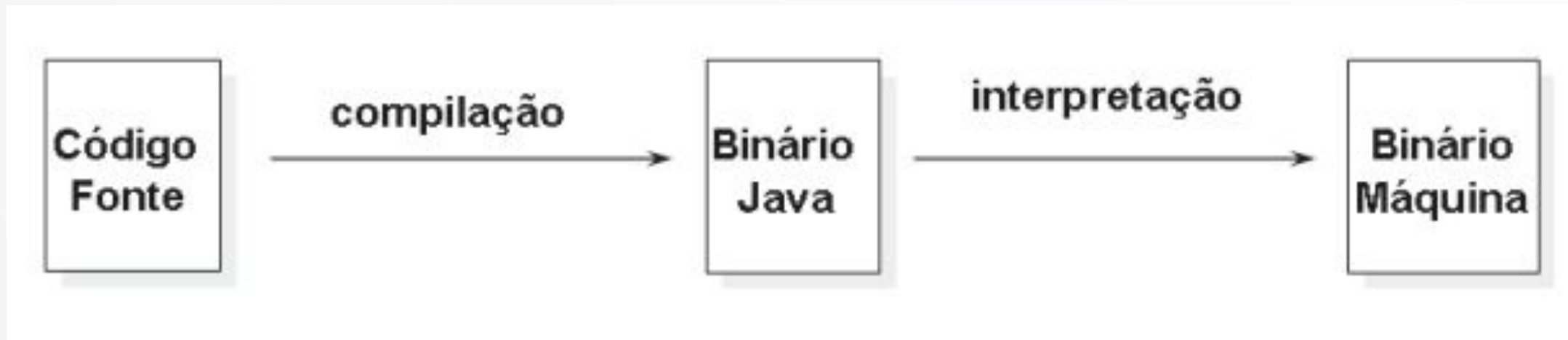
O que é Java

Criada pela Sun Microsystems (posteriormente adquirida pela Oracle) em 1995, Java é uma plataforma de desenvolvimento que usa a linguagem de mesmo nome para a programação de aplicações

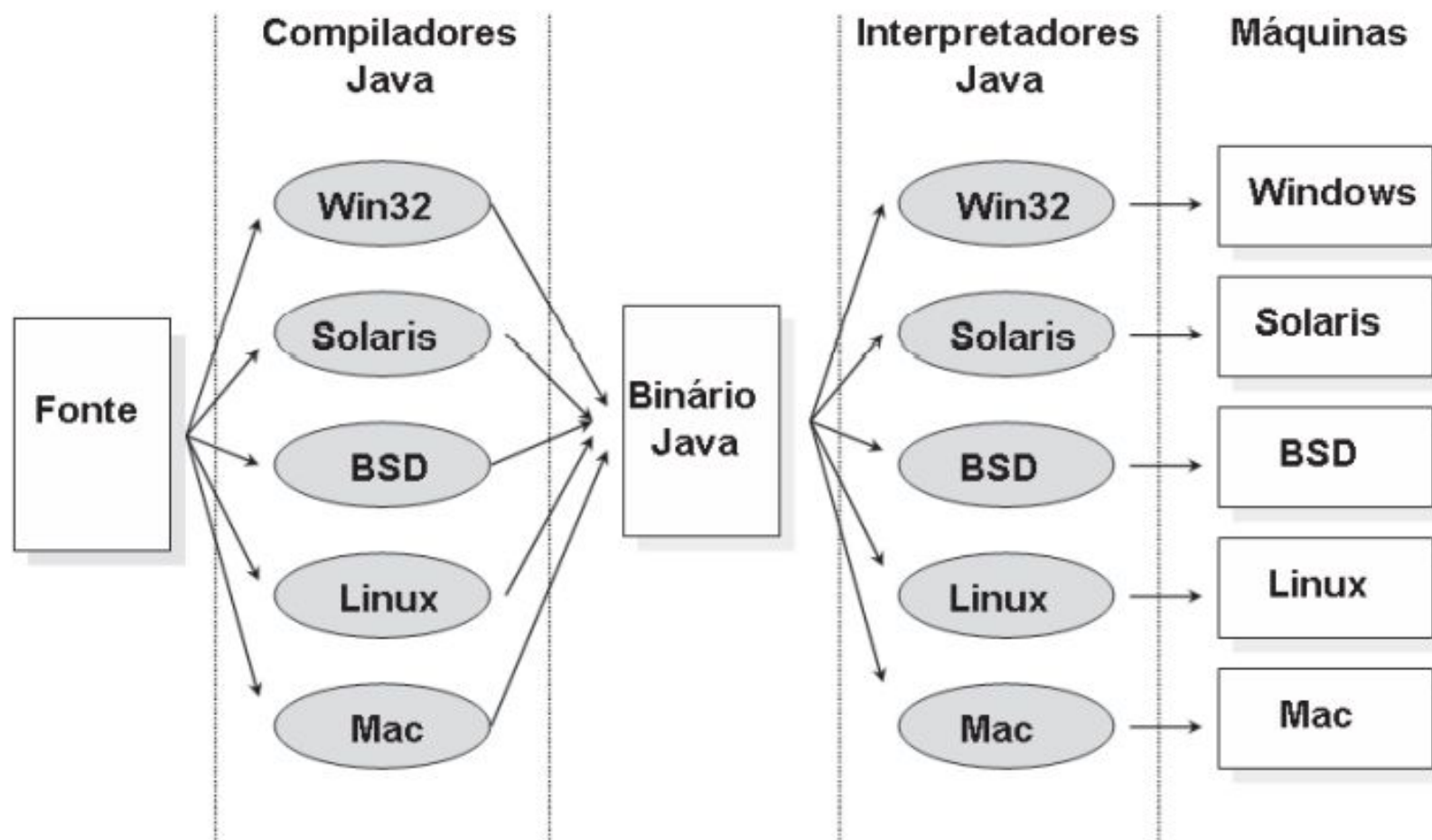
Senac

JVM

- A JVM, ou Java Virtual Machine, é a denominação dada ao interpretador Java rodando sob o sistema operacional (SO) de qualquer máquina. O código fonte é compilado para código binário Java, que é interpretado e executado em diferentes plataformas



Compiladores e Interpretadores



Versões

- JSE (Java Standard Edition): pode ser considerada o core (a base principal) da linguagem, projetada para execução em máquinas simples e estações de trabalho.
- JEE (Java Enterprise Edition): voltada para o desenvolvimento de aplicações baseadas em servidor, como páginas JSP (JavaServer Pages), Servlets, XML (Extensible Markup Language) etc.
- JME (Java Micro Edition): projetada para dispositivos com menor poder de processamento e memória, tais como dispositivos móveis, celulares etc.

Vantagens

- Multiplataforma
- Compatibilidade
- Portabilidade
- Verificação da segurança do código
- Toda máquina pode ser Java Virtual por software

Senac

Desvantagens

- Interpretação - queda de desempenho
- Duas ferramentas: compilador e interpretador
- Código nativo - plataforma específica

Senac

Características da linguagem

- Totalmente orientada a objetos
- Multithread (paralelização de processos)
- Sintaticamente e morfolologicamente quase idêntica ao C++
- Ausência de ponteiros • redução de bugs • aumento da segurança
- Independe da plataforma
- Orientada a aplicações de rede (especialmente Web)
- Quatro S's : small, simple, safe and secure

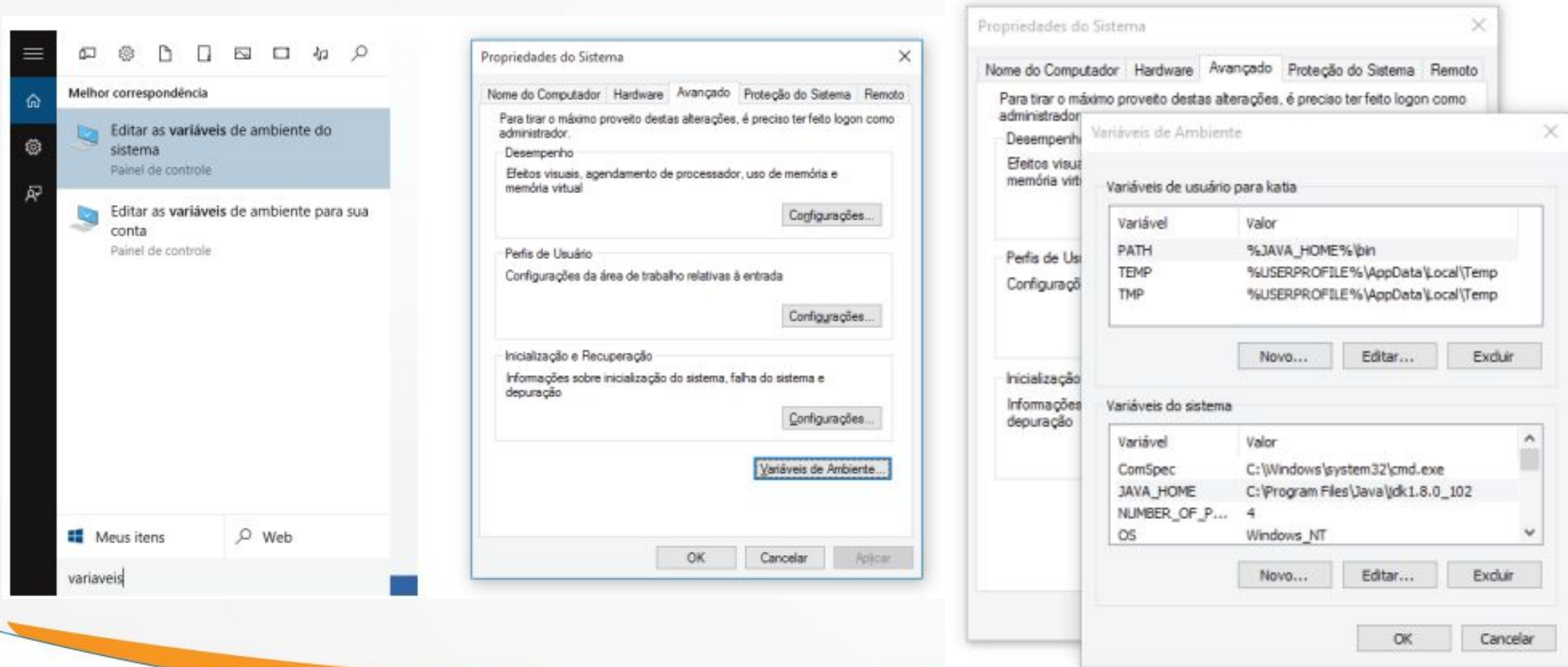
Como iniciar o desenvolvimento

- Fazer o download da JDK no site da Oracle:
<https://www.oracle.com/java/technologies/downloads/>
- Instalar no computador (necessário acesso para instalação)

Senac

Como iniciar o desenvolvimento

- Incluir variável de ambiente de ambiente



Como iniciar o desenvolvimento

- Baixar uma IDE de desenvolvimento:
 - <https://www.eclipse.org/downloads/>
 - <https://www.jetbrains.com/pt-br/idea/download> (versão ultimate necessita de licença)

Senac

Hello World

```
public class Hello{  
    public static void main(String[] args){  
        System.out.println("Hello World");  
    }  
}
```

Convenções de Nomenclatura

- Variáveis:
 - Iniciadas em letra minúscula
 - Se for mais de uma palavra utilizar **camelCase**
- Constantes:
 - letras maiúsculas
 - separação **snake_case**
- Classes
 - Substantivos
 - Iniciadas em letra maiúscula
- Métodos:
 - Verbos
 - Se for mais de uma palavra utilizar **camelCase**

Tipos de dados

Primitivos			
	Tipo	Tamanho	Faixa de Valores
Inteiros	byte	8 bits	-128 a 127
	short	16 bits	-32.768 a 32.767
	int	32 bits	-2.147.483.648 a 2.147.483.647
	long	64 bits	-9.223.372.036.854.775.808 a 9.223.372.036.854.775.807
	boolean	8 bits	true / false
flutuante	float	32 bits	-3,4E-38 a 3,4E_38
	double	64 bits	-1.7E-308 a 1.7E+308
	char	16 bits	Caracteres: "A" / Unicode: "\u0041" / inteiros de 0 a 65535

Conversões

- `Int x = 10;`
- `Float y = (int) 10;`
- `int idade = Integer.parseInt("10")`
- `String idade = String.valueOf(idade);`

Palavras reservadas

abstract	boolean	break	byte	case
catch	char	class	continue	default
do	double	else	enum	extends
final	finally	float	for	if
implements	import	instanceof	int	interface
long	new	package	private	protected
public	return	short	static	super
switch	this	throw	throws	try
void	while			

Lista completa:

https://www.w3schools.com/java/java_ref_keywords.asp

Operadores

Operador	Significado
&&	AND / E
	OR / OU
!	NOT
==	IGUAL A
!=	DIFERENTE
>	MAIOR QUE
<	MENOR QUE
<=	MENOR OU IGUAL A
>=	MAIOR OU IGUAL A

Estruturas condicionais

```
if (condição) {  
    // bloco de código que só executa se for verdadeiro  
} else {  
    // bloco de código que só executa se for falso  
}
```

Ternário:

condição ? bloco de código verdadeiro : bloco de código falso

Estruturas condicionais

```
switch (expressao){  
    case valor:  
        // bloco de código break;  
        // outros cases  
    default:  
        // bloco de código  
}
```

Estruturas de Repetição

```
while(condição)
{
    // bloco de código
}
```

```
do {
    // bloco de código
} while (condição);
```

Estruturas de Repetição

```
for(int i=0; i<10;i++){  
    System.out.println(i);  
}
```

```
for(int a : arrayValores)  
{ soma += a; }
```


Exercícios

- Desenvolva um programa que imprime na saída padrão o número de argumentos e os argumentos passados para o programa.
- Desenvolva um programa que calcule a tabuada de 1 a 10
- Desenvolva um programa que calcule o fatorial de um número passado como argumento e que teste a integridade do mesmo:
 - verifica se o número é inteiro e se tem um valor suficientemente pequeno para que o resultado seja coerente
 - imprima mensagens para o usuário alertando para possíveis erros no argumento (inclusive avisando, caso o usuário não tenha passado nenhum argumento)
- Desenvolva um programa que converta números decimais em números hexadecimais.

Exercícios

- Fazer método que transforme um array de strings em uma string concatenada
- Faça um método que imprima uma string invertida (por exemplo: carro => orrac)
- Faça um método que verifique se uma string contém um texto de busca (exemplo> procurar a palavra aluno em uma frase)

Exercícios

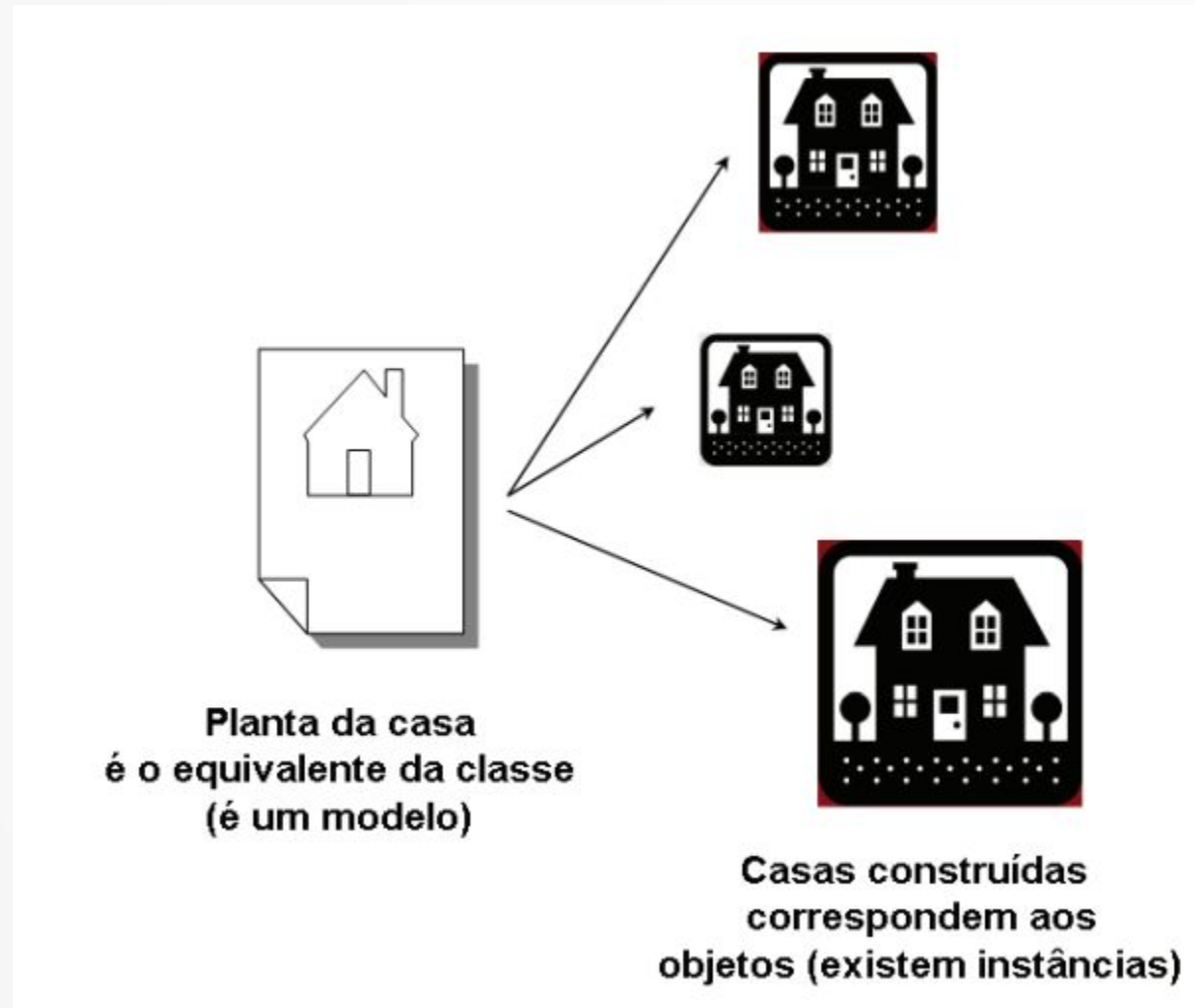
- Faça uma aplicação de reserva de passagens, considerando os seguintes requisitos:
 - O avião terá 6 assentos por fileira e 28 fileiras
 - Utilizar o método Random para definir alguns lugares já ocupados
 - Primeiro deverá ser exibido para o cliente a lista de assentos ocupados
 - Deverá ser solicitado ao cliente digitar primeiro a fileira e depois o assento
 - Caso ele escolha um lugar já ocupado, exibir uma mensagem de que o assento está ocupado. Caso contrário, reserve o assento.
 - Ao final mostrar uma mensagem perguntando se deseja reservar mais um assento, caso positivo, voltar ao início, caso contrário, finalizar a aplicação

Orientação à objetos

O que é:

- Objetos são modelados como objetos do mundo real, ou seja, possuem existência “física” dentro de um programa, ocupando espaço de memória e armazenando as informações pertinentes.
- Classes são definições genéricas de objetos (moldes de objetos). Podem existir, portanto, vários objetos de uma única classe.
- Classes são definidas para que a partir delas possam ser gerados os objetos que assumem aquelas propriedades e executam aquela funcionalidade definida na classe.

Orientação à objetos



Orientação à objetos

Pra que serve:

- Metodologia eficiente.
- Programas grandes e desenvolvidos por equipes.
- Grande índice de reaproveitamento do código

Pacotes

Dentro de um projeto nossas classes ficam organizadas em pacotes (packages)

O nome completo de uma classe compreende o nome do pacote ao qual ela pertence junto com o nome com o qual ela foi declarada. A classe Date, do Java, está no pacote java.util, portanto seu nome completo é java.util.Date. Num arquivo fonte qualquer, para se utilizar a classe Date, ou importamos o pacote no início do fonte ou a referenciamos pelo seu nome completo;

Pacotes

- Define o nome completo da classe (“fully qualified name class”).
- Uso da palavra reservada package.
- Devem ser declarados nas primeiras linhas do arquivo Java.
- Um outro arquivo fonte pode referenciar uma classe pelo seu nome completo ou com o uso de import.
- Nomes de pacotes devem estar todo em minúsculo, apenas letras
- É padrão usar o nome do domínio da empresa pra iniciar: (ex: com.sun.***.br.edu.senac.***)

Classes

Uma classe contém uma definição de um conjunto de variáveis e funções que são encapsuladas conjuntamente e compartilham o mesmo escopo de definições. Assim, uma classe é definida por seus atributos (variáveis) e comportamento (métodos ou funções)

Regras gerais

- As classes devem ser claras e significativas com respeito às propriedades que representam.
- Seus métodos devem abranger, por definição, “todo” o comportamento desejado da classe, inclusive aqueles ainda não implementados, ou que devem ser implementados por subclasses.

Exemplo

```
package com.senac.escola;
```

```
public class Endereco {  
    private String nomeRua;  
    private String cidade;  
    private String numero;  
    private String complemento;  
    private Estado estado;
```

```
    //código omitido
```

```
}
```

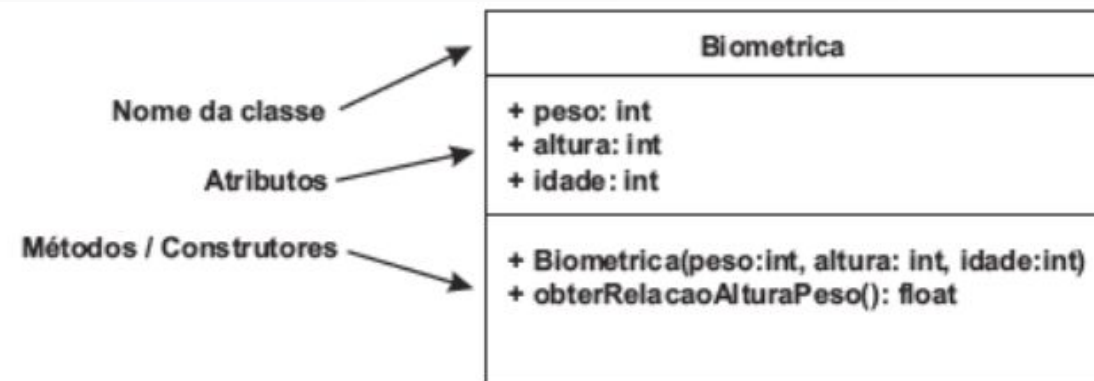
Diagrama de Classes

A notação para especificar uma classe é um retângulo que possui três regiões: uma com o nome da classe, outra com os seus atributos e uma terceira com os seus métodos ou construtores.

Os atributos são especificados com seus tipos e sua acessibilidade:

- + para acesso público e
- para atributos privados

Os métodos são especificados com seus tipos de retorno e lista de argumentos, além da acessibilidade, como os atributos.



Construtores

- Método que inicializa o objeto. É o primeiro método chamado quando o objeto é criado (instanciado) e somente pode ser chamado nesta situação.
- Usado para inicializar atributos do objeto e executar os métodos de inicialização (abrir um arquivo, por exemplo).
- Deve ter o mesmo nome da classe e não possui valor de retorno. Pode existir em qualquer número para uma mesma classe (difere pelo número de argumentos).

Construtores

```
public class Endereco {  
    ...código omitido  
    public Endereco() {  
    }  
}
```

```
public class Endereco {  
    ..código omitido  
    public Endereco(String nomeRua, String cidade, String numero, String complemento, Estado estado) {  
        this.nomeRua = nomeRua;  
        this.cidade = cidade;  
        this.numero = numero;  
        this.complemento = complemento;  
        this.estado = estado;  
    }  
}
```

Metódos

- Um método é uma função definida dentro de uma classe e que implementa um comportamento desta classe. Quando não possuir nenhum tipo de retorno deve ser utilizada a palavra chave void. Sempre que possuir um tipo de retorno (não for void) deve incluir um return

```
public void incrementarIdade( ) {  
    this.idade++;  
}
```

```
public String getNome( ) {  
    return this.nome;  
}
```

Metódos

- Os argumentos de um método devem ser definidos pelos seus tipos.

```
public void acrescentarAltura(int aumento) {  
    this.altura += aumento;  
}
```

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


Qualificadores

- **public:** o método é visível por qualquer classe. É o qualificador mais aberto no sentido de que qualquer classe pode usar esse método;
- **private:** o método é visível apenas pela própria classe. É o qualificador mais restritivo;
- **protected:** o método é visível pela própria classe, por suas subclasses e pelas classes do mesmo pacote.

Sobrecarga de métodos / Construtores

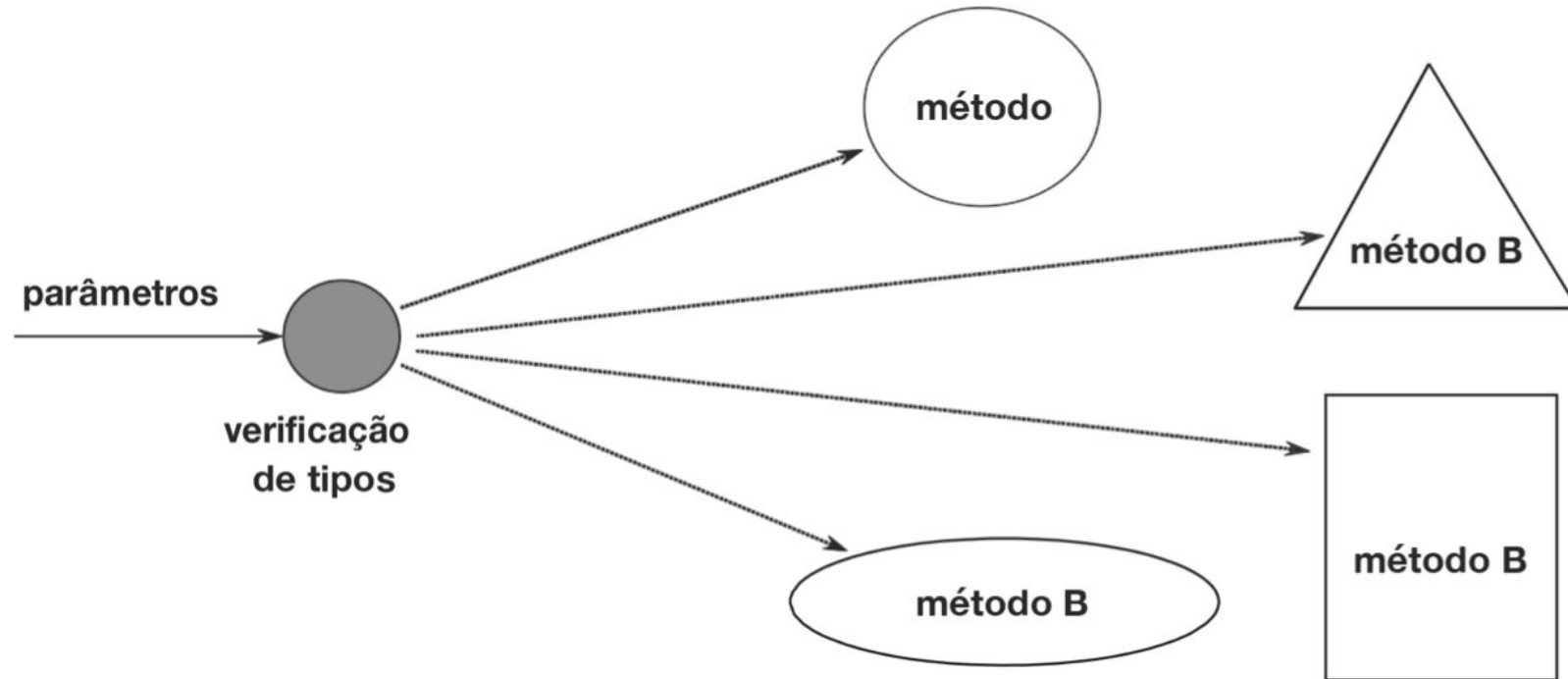
- Polimorfismo de métodos ou sobrecarga de métodos **permite que métodos com o mesmo nome sejam implementados de forma diferente**. Isto se torna muito útil quando empregado para prover o “mesmo” comportamento para um conjunto de parâmetros diferentes.
- Da mesma forma que métodos, os **construtores também podem ser sobrecarregados**. A diferenciação ocorre da mesma forma, pela lista de argumentos.

Sobrecarga de construtores

```
public class Endereco {  
    ...código omitido  
    public Endereco() {  
    }  
}
```

```
public Endereco(String nomeRua, String cidade, String numero, String complemento, Estado estado) {  
    this.nomeRua = nomeRua;  
    this.cidade = cidade;  
    this.numero = numero;  
    this.complemento = complemento;  
    this.estado = estado;  
}  
}
```

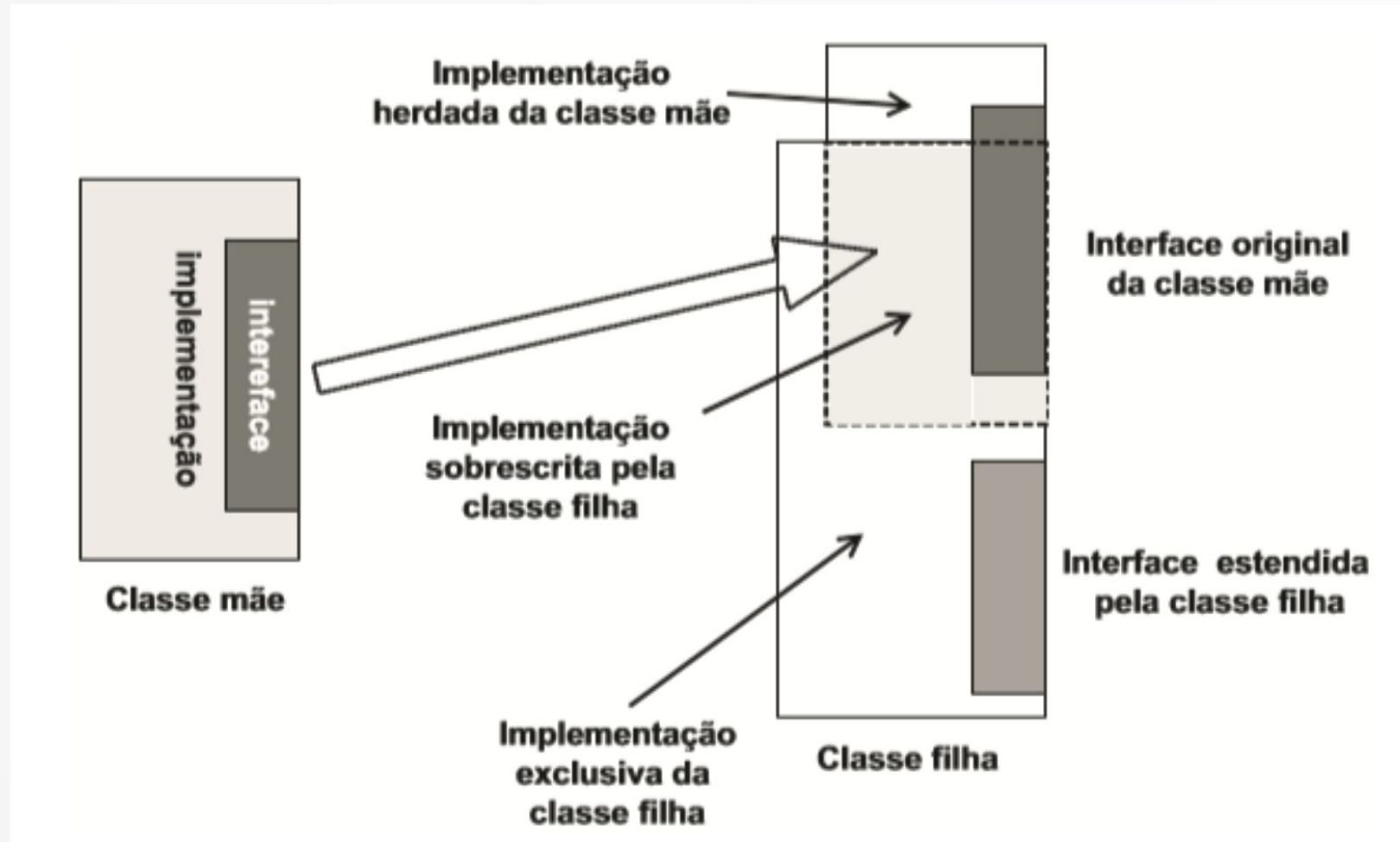
Sobrecarga de métodos / Construtores



Atributos estáticos

- Atributos e métodos estáticos são também chamados de atributos ou métodos de classe, em contraposição aos atributos e métodos de instância, que assumem um valor para cada instância (objeto).
- São alocados uma única vez em memória, e permanecem alocados enquanto a classe estiver carregada.
- Um atributo estático é compartilhado entre todos os objetos da classe. Se o atributo for público, outras classes e objetos também utilizam a sua única alocação em memória.

Herança



Herança

- Usando herança podemos definir uma nova classe a partir de outra, preservando sua interface e acrescentando ou alterando sua implementação:
- É uma forma pela qual novas classes são definidas de forma incremental.
- Definição de uma nova classe especificando uma classe mãe e um conjunto adicional de atributos e/ou métodos.
- A classe definida dessa forma herda todos os atributos e métodos da classe mãe.
- A acessibilidade da interface da classe mãe não pode ser redefinida na classe herdada.
- A linguagem Java não permite herança múltipla com classes, mas apenas com interfaces.

Herança

```
public class Gerente extends Funcionario {  
    ...código omitido  
    public double obterCusto() {  
        return this.salario * encargo + this.bonus * encargoBonus;  
    }  
}
```


Classes abstratas

- Métodos abstratos são aqueles que não possuem implementação.
- Uma classe com apenas um método abstrato não pode ser instanciada e é também chamada de classe abstrata.
- **Classes abstratas não podem ser instanciadas**, uma vez que “falta” parte de sua implementação. Uma classe abstrata serve para estabelecer uma interface cujas subclasses, se precisarem ser instanciadas, necessariamente terão de ser implementadas.
- Cria-se uma classe abstrata para evitar que ela possa ser implementada, **devendo ser apenas base de herança**

Interfaces

- Interfaces são o conjunto de atributos estáticos (variável única compartilhada por todos os objetos de uma mesma classe) e/ou métodos abstratos (métodos com assinatura definida, mas corpo não definido) que estabelecem uma série de comportamentos **que se espera da classe que a implementa**
- Classes e interfaces são equivalentes, exceto pelo fato de que **interfaces não podem ser instanciadas**.
- Ao contrário da classe abstrata **a interface não possui nenhum método implementado**; é apenas um "contrato"

Interfaces

- Herança múltipla em Java: cada classe pode ter apenas UMA superclasse e implementar VÁRIAS interfaces.
- Vantagem: Duas classes distintas que implementam a mesma interface respondem a um conjunto idêntico de funções (embora não necessariamente da mesma forma).

Senac

Interface

```
public interface Colaborador { public double obtercusto( ); }
```

```
public class Consultor implements Colaborador {  
    public double valorhora;  
    public int totalhoras;  
    public double obterCusto( ) {  
        return this.valorhora * this.totalhoras;  
    }  
}
```

Composição

- Um objeto pode ser composto de outros objetos, por exemplo, um funcionário que pertence a um departamento:

```
public class Departamento{  
    ...código omitido  
}
```

```
public class Funcionario{  
    ...código omitido  
    Departamento departamento;  
}
```

Collections

- A linguagem Java oferece uma série de classes que estruturam objetos de uma forma mais prática do que arrays: Listas, Listas ligadas, Hashtables.
- No pacote `java.util` existe uma série de classes que implementam a interface `Collection`. Todas as classes concretas que implementam essa interface são estruturas para armazenar um conjunto de objetos com as mais variadas finalidades

- Map<String, Aluno> mapa = new TreeMap<String, Aluno>();
Aluno aluno = new Aluno("Maria", "1231231", "123")
mapa.put(aluno.getMatricula(), aluno);
mapa.get(aluno.getMatricula());
- List<String> list = new ArrayList<>();
list.add("Maria");

Exercícios

- Construa uma classe que represente um círculo:
 - Deverá receber em seu construtor o valor do raio do círculo e/ou a cor
 - Construir métodos para retornar a cor, a área e o perímetro do círculo

Exercícios

Construa uma aplicação para calcular a folha de pagamento de um empresa que possui diversos cargos e departamentos, obedecendo às seguintes regras:

- São armazenados as seguintes informações de todos os funcionários: nome, CPF, data de nascimento, endereço, estado civil, formação.
- Para o cadastro de gerentes deve ser adicionado, centro de custo;
- Todo departamento deve possuir um gerente e uma lista dos funcionários;
- Todos os funcionários, exceto estagiários recebem comissão de vendas de 15%; gerentes recebem comissão de 20%;
- Todos os funcionários devem pertencer a um departamento;
- Construir método para calcular salário do funcionário;
- Imprima a lista de funcionários de um departamento

Exercícios

Construa as classes para uma escola, definindo:

- Aluno: nome, cpf, matrícula, endereço, email, data de nascimento
- Turma: id, nome da disciplina, professor, média mínima, descrição
- Professor: nome, cpf, endereço, email, data de nascimento, matrícula
funcionário

Exercícios

Faça uma classe Banco que contenha um grupo de contas (que devem ser alocados em um array de Contas).

Faça métodos que efetuem cadastramento de contas, movimentações financeiras (débitos e créditos em contas especificadas pelo seu número).

Faça um método que informe o ativo do banco (soma de todos os saldos).

Faça um método que debite um valor fixo (taxa de administração de R\$ 3,50) de todas as contas.

Exercícios

Construa uma aplicação para calcular o valor a ser cobrado por serviço por uma oficina mecânica:

- São armazenadas as seguintes informações de todos os clientes: nome, cpf, data de nascimento, endereço.
- São armazenados as seguintes informações de todos os veículos: placa, ano, combustível, importado/nacional.
- São armazenados as seguintes informações de todos os mecânicos: nome, cpf, data de nascimento, endereço, valor por hora;
- São armazenados as seguintes informações de todas as notas fiscais: placa do veículo, cpf do cliente, cpf do mecânico, horas trabalhadas, peças adicionais, descrição do serviço realizado, data de início e data de término do serviço, prazo inicial;
- Construir método para calcular o valor da nota fiscal;

Datas

- Date() => Instancia um objeto Date que representa a data e hora que foi criado

```
SimpleDateFormat dateFormat = new SimpleDateFormat("dd/MM/yyyy");
Date resultado;
try{
    resultado = dateFormat.parse("25/10/2021");
} catch (ParseException e) {
    resultado = new Date();
}
```

Exceções

- O tratamento de exceções e erros em Java é uma forma de tornar o código mais previsível e controlável.
- As exceções e erros são modelados por objetos herdados da classe básica Throwable. Este modelamento em objetos permite estabelecer uma hierarquia de erros, muito útil na definição de tratamentos específicos e/ou genéricos. Diante de um método que pode gerar alguma condição de exceção ou erro, o programador pode tomar três atitudes:

- Não capturar, declarar a passagem da exceção/erro para uma classe específica e não implementar o seu tratamento:

```
public void metodoQuePreveExc() throws umaExcecao {  
    ...  
    // chamada a métodos que podem gerar uma exceção  
    // do tipo umaExcecao ...  
}
```

- Capturar e passar a condição para o nível superior de seu código:

```
public void metodoQuePreveExc() throws MinhaExcecao{  
    try {  
        // algum método que pode gerar uma excecao  
        // do tipo MinhaExcecao  
    }  
    catch (MinhaExcecao e) {  
        throw e;  
        // ...para o nível externo tratar  
    }  
}
```


- Capturar e tratar a condição “in loco”:

```
public void metodoQuePreveExc() {  
    ...  
    try {  
        // algum método que pode gerar uma exceção  
        // do tipo umaExcecao  
    } catch (umaExcecao e) {  
        // tratamento da exceção  
    }  
    ...  
}
```

Threads

- Multithread é a capacidade de um programa de rodar diferentes trechos (ou diferentes instâncias do mesmo trecho) de forma independente. Permite, por exemplo, que o programa realize tarefas em background enquanto mantém “viva” a interface com o usuário.
- Implementação
 - Classe deve implementar interface Runnable.
 - Classe deve possuir uma variável que contenha um tipo Thread.
 - Método start() deve ser invocado para disparar uma nova thread.
 - O método run() deve conter o código que roda dentro da thread.

Threads

```
package com.senac.threads;

import java.util.Date;

public class RelogioMultiThread implements Runnable {

    Date data;

    public void run() {

        while (true) {
            data = new Date( );
            System.out.println(data);
            try {
                Thread.sleep(1000);
            }
            catch (InterruptedException e) { }
        }

    }

}

package com.senac.threads;

public class Programa {

    public static void main(String[] args) {
        RelogioMultiThread relógio = new RelogioMultiThread();
        Thread t = new Thread(relógio);
        t.start();
    }

}
```

Generics

```
• public class Generics<Tipo1,Tipo2> {  
    Tipo1 e;  
    Tipo2 t;  
  
    Generics(Tipo1 e, Tipo2 t) {  
        this.e = e;  
        this.t = t;  
    }  
    public String toString()  
    {  
        return e.toString() + t.toString();  
    }  
}
```

Generics

- `List<Generics> list = new ArrayList<>();`

```
public void preencheList() {  
    this.list.add(new Generics<Integer, String>(1, "Maria"));  
    this.list.add(new Generics<Boolean, String>(true, "Maria"));  
}
```

Exercícios

- Um banco oferece aos seus clientes a possibilidade de abertura e movimentação de 2 tipos de contas, Poupança e Corrente. Ambas possuem agência, conta e saldo, além da possibilidade de debitar e creditar valores.
A principal diferença entre elas é que a conta poupança não pode ter saldo negativo e a conta corrente pode ter limite de conta (Cheque especial)
- Faça uma classe Banco que contenha um grupo de contas (que devem ser alocados em uma lista de Contas `ex:(Map<String, Conta> contas = new HashMap<>();)`).
- Faça métodos que efetuem cadastramento de contas, movimentações financeiras (débitos e créditos em contas especificadas pelo seu número).
- Faça um método que debite um valor fixo (taxa de administração de R\$ 3,50) de todas as contas; `ex: contas.values().forEach()`

Onde Praticar

- <https://www.beecrowd.com.br> (em português)
- <https://www.w3schools.com/java/exercise.asp>
- <https://leetcode.com/problemset/all/>
- <https://www.hackerrank.com/>

Atalhos Eclipse

Atalho	Windows	Mac
Busca rápida	Ctrl + 3	⌘ + 3
Abrir arquivo	Ctrl + Shift + R	⌘ + Shift + R
Encontrar todas as referências a uma classe	selecionar nome da classe + Ctrl + Shift + G	selecionar nome da classe + ⌘ + Shift + G
Mover linha	Alt + ↑ ou ↓	Alt + ↑ ou ↓
Sugestões de classes	Ctrl + espaço	⌘ + espaço
Inserir comentário	Ctrl + /	⌘ + /
Opções avançadas (gerar construtor / getter / setter)	Alt + Shift + S	Opt + ⌘ + S
Renomear variável	Ctrl + 2, R	-

Referências

- <https://docs.oracle.com/en/java/index.html> (inglês)
- [Learn Java | Codecademy](#) (inglês)
- <https://www.devmedia.com.br/artigos/java>
- Apostila JAVA 2017 - SENAC
- FURGERI, S. Java Ensino Didático. São Paulo: Editora Érica, 2018. E-book.

SENAC