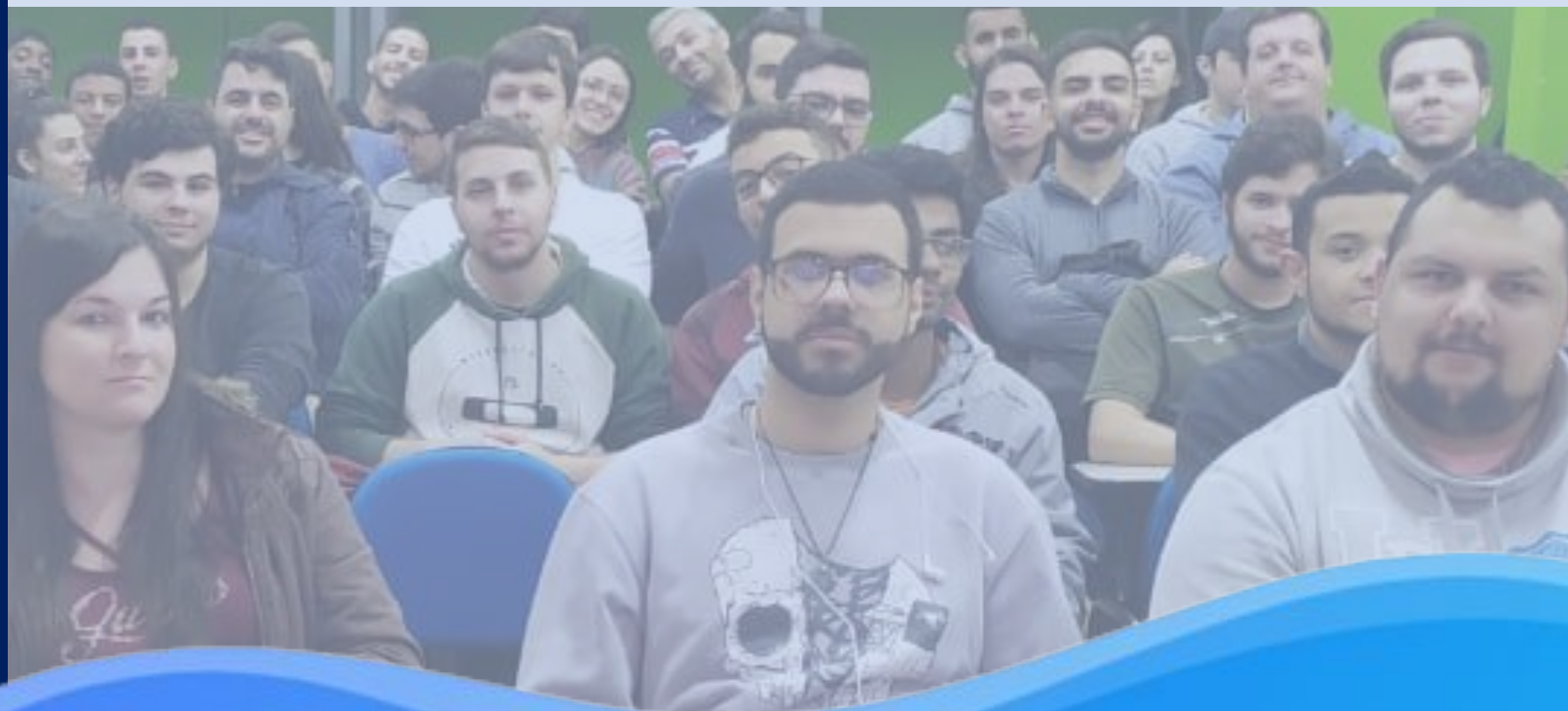


53 45 52 45 49 20 46 49 45 4c 20
41 4f 53 20 50 52 45 43 45 49 54
4f 53 20 44 41 20 48 4f 4e 52 41
20 45 20 44 41 20 43 49 c3 8a 4e
43 49 41 2c 20 50 52 4f 4d 4f 56
45 4e 44 4f 20 4f 20 55 53 4f 20
45 20 4f 20 44 45 53 45 4e 56 4f
4c 56 49 4d 45 4e 54 4f 20 44 41
20 49 4e 46 4f 52 4d c3 81 54 49
43 41 20 45 4d 20 42 45 4e 45 46
c3 8d 43 49 4f 20 44 4f 20 43 49
44 41 44 c3 83 4f 20 45 20 44 41
20 53 4f 43 49 45 44 41 44 45 2e

RESIDÊNCIA DE SOFTWARE

**CAPACITAR
TREINAR
EMPREGAR**

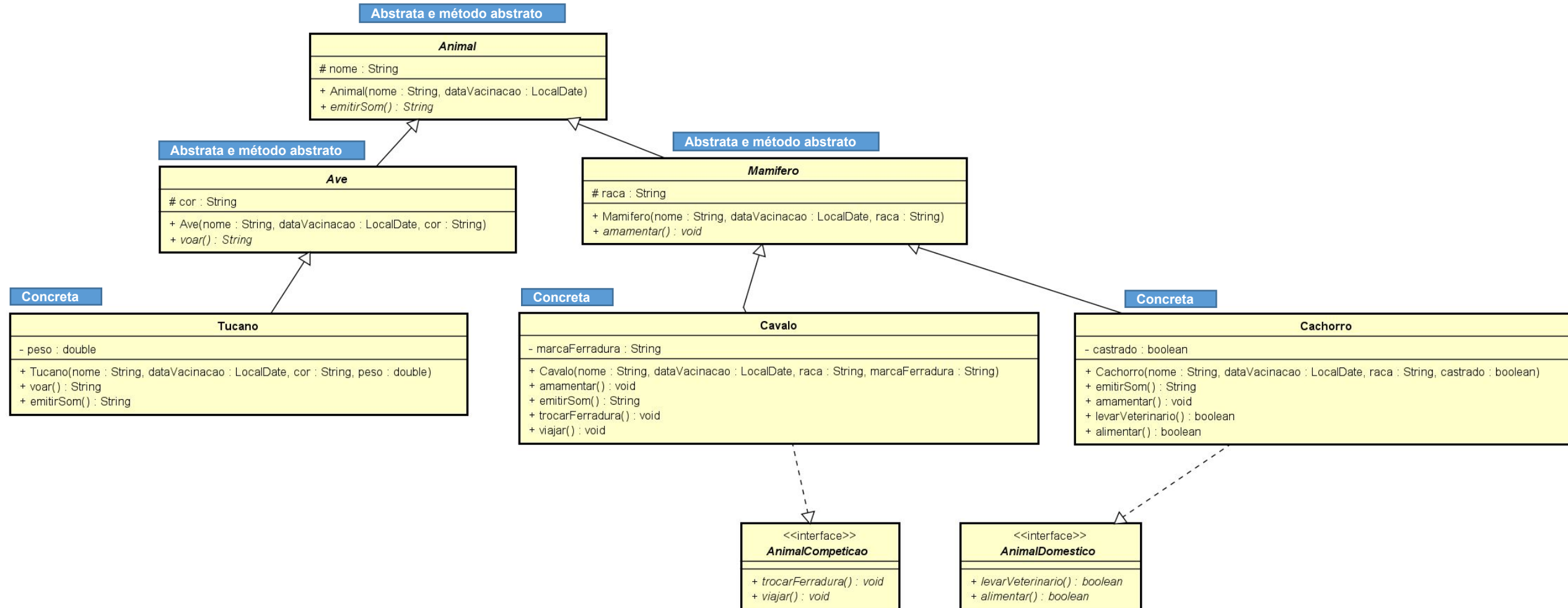
TRANSFORMAR



JAVA I
Enumeração e “final”
28/07/2020

REVISÃO CONCEITOS – HERANÇA, CLASSE E MÉTODOS ABSTRATOS E INTERFACE

Criar as classes abaixo de acordo com o diagrama



CLASSES COM FINAL

Classes finais não podem ser utilizadas como classes-pai, impedindo com que classes-filhas sejam criadas a partir delas. O final faz com a classe fique **imutável** não podendo utilizar o **extends** nas classes filhas. A classe Math por exemplo é final.

Classe com final. Erro na tentativa de herdar da super classe

```
Policial.java ✕  
  
package aula;  
  
public final class Policial {  
    protected String cpf;  
    protected String nome;  
}
```

```
PolicialFederal.java ✕  
  
package aula;  
  
public class PolicialFederal extends Policial {  
    private String matricula;  
}
```

Métodos e atributos com final

O **final** também pode ser aplicado em atributos e métodos de uma classe. Um método com **final** não pode ser **sobrescrito** pelas classes filhas. Um atributo com **final** não pode ser modificado sendo definido valores **constantes**.

MÉTODOS COM FINAL

Vamos fazer um teste retirando o final da classe **Policial** e vamos criar um método com **final**.

```
Policial.java ✕  
  
package aula;  
  
public class Policial {  
    protected String cpf;  
    protected String nome;  
  
    public final void mostrarValores() {  
        System.out.println(cpf + "-" + nome);  
    }  
}
```

```
PolicialFederal.java ✕  
  
package aula;  
  
public class PolicialFederal extends Policial {  
    private String matricula;  
  
    public final void mostrarValores() {  
        System.out.println(cpf + "-" + nome + "-" + matricula);  
    }  
}
```

Retorna erro pois o método da super classe não pode ser sobrescrito.

```
*Policial.java X
package aula;

public class Policial {
    protected String cpf;
    protected String nome;
    protected final String lotacao = "Brasília";

    public String getCpf() {
        return cpf;
    }

    public void setCpf(String cpf) {
        this.cpf = cpf;
    }

    public String getNome() {
        return nome;
    }

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

    public String getLotacao() {
        return lotacao;
    }

    public final void mostrarValores() {
        System.out.println(cpf + "-" + nome);
    }
}
```

Adicione o atributo lotacao e os getters e setters para a classe Policial

Execute a aplicação e veja que todo policial será lotado em Brasília.
Definimos um atributo constante na classe Policial

Atributo final e construtor

O valor para atributo constante pode ser definido em um construtor.

Retire o conteúdo do atributo **lotacao** da classe **Policial** e adicione o construtor abaixo (ALT+SHIFT+s)

```
protected final String lotacao;  
  
public Policial(String lotacao) {  
    this.lotacao = lotacao;  
}
```

```
TestaPolicial.java X  
package aula;  
  
public class TestaPolicial {  
    public static void main(String[] args) {  
        Policial p = new Policial();  
        p.setCpf("124.898.800-78");  
        p.setNome("Jorge");  
        System.out.println(p.getNome() + "-" + p.getLotacao());  
    }  
}
```

Insira o conteúdo na construção do objeto na classe **TestaPolicial**

```
Policial p = new Policial("Brasília");
```

EXERCÍCIO

Criar uma classe com o nome **FuncionarioPublico**. Atributos: nome, salario e anoConcurso;
O atributo anoConcurso deve ser final e seu valor inicializado no construtor.
Criar alguns objetos do tipo FuncionarioPublico e mostrar seus atributos.

RESOLUÇÃO

```
package aula;

public class Main {

    public static void main(String[] args) {
        FuncionarioPublico func1 = new FuncionarioPublico("Joao", 1000, "Concurso 2010");
        FuncionarioPublico func2 = new FuncionarioPublico("Jose", 5000, "Concurso 1974");
        FuncionarioPublico func3 = new FuncionarioPublico("MAria", 1200, "Concurso 2000");

        System.out.println(func1.getNome() + ", " + func1.getSalario() + ", " + func1.getANO_CONCURSO());
        System.out.println(func2.getNome() + ", " + func2.getSalario() + ", " + func2.getANO_CONCURSO());
        System.out.println(func3.getNome() + ", " + func3.getSalario() + ", " + func3.getANO_CONCURSO());
    }
}
```

```
package aula;

public class FuncionarioPublico {
    private String nome;
    private double salario;
    protected final String ANO_CONCURSO;

    public FuncionarioPublico(String nome, double salario, String anoConcurso) {
        super();
        this.nome = nome;
        this.salario = salario;
        this.ANO_CONCURSO = anoConcurso;
    }

    public String getNome() {
        return nome;
    }

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

    public double getSalario() {
        return salario;
    }

    public void setSalario(double salario) {
        this.salario = salario;
    }

    public String getANO_CONCURSO() {
        return ANO_CONCURSO;
    }
}
```


ENUM

Um enum é uma estrutura enumerada em conjuntos de constantes organizados em ordem de declaração. O enum contém uma lista de valores pré-definidos. A funcionalidade principal de enum é agrupar valores com o mesmo sentido dentro de uma única estrutura, como por exemplo, meses, dias da semana, cores, marcas e etc. Um enum também pode limitar os valores que podem ser usados na programação.

Vamos inserir o enum clicando no eclipse em File-new-Enum

```
*MarcaTV.java ✕  
package aula;  
  
public enum MarcaTV {  
    SAMSUNG, PANASONIC, SONY, LG;  
}
```

Estruturas enum não são instanciáveis.

No exemplo abaixo temos uma classe denominada TV, e que nossa TV deverá ser de uma das marcas pré definidas.

```
*TV.java ✕  
.  
.  
public class TV {  
    private int tamanho;  
    private String modelo;  
    private MarcaTV marca;  
  
    public TV(int tamanho, String modelo, MarcaTV marca) {  
        this.tamanho = tamanho;  
        this.modelo = modelo;  
        this.marca = marca;  
    }  
  
    public int getTamanho() {  
        return tamanho;  
    }  
  
    public String getModelo() {  
        return modelo;  
    }  
  
    public MarcaTV getMarca() {  
        return marca;  
    }  
}
```

ENUM

Inserir a classe de teste **TestaTV**.

```
TestaTV.java ✖  
package aula;  
  
public class TestaTV {  
    public static void main(String[] args) {  
        TV tv = new TV(32, "SN2502 Full HD", MarcaTV.SAMSUNG);  
  
        System.out.println("Minha TV é: " + tv.getMarca() + " " + tv.getModelo()  
            + " " + tv.getTamanho());  
    }  
}
```

Acesso direto ao valor do enum
MarcaTV. As constantes
São acessadas estaticamente.

ENUM

Em estruturas **enum** podemos atribuir mais valores. Por exemplo, podemos fazer uma estrutura **enum** de períodos de curso contendo os dias, carga horária e valor dos cursos. Para fazer isso, no final da enumeração, temos que declarar as constantes que foram usadas.

```
*PeriodoCurso.java
package aula;

public enum PeriodoCurso {
    INTEGRAL("terça e quinta", 40, 2400.), NOTURNO("sexta", 20, 1000.), MANHA(
        "segunda e quarta", 30, 2800.);

    private final String diasSemana;
    private final int cargaHoraria;
    private final double valor;

    private PeriodoCurso(String diasSemana, int cargaHoraria, double valor) {
        this.diasSemana = diasSemana;
        this.cargaHoraria = cargaHoraria;
        this.valor = valor;
    }

    public String getDiasSemana() {
        return diasSemana;
    }

    public int getCargaHoraria() {
        return cargaHoraria;
    }

    public double getValor() {
        return valor;
    }
}
```

Construtor do enum: Essas definições se parecem muito com os construtores de classes, mas sua função é apenas indicar o que é cada uma das informações contidas em cada constante. Lembrando que um **enum** não pode ser instanciado.

Inserindo a classe turma

```
Turma.java ✕  
package aula;  
  
public class Turma {  
    private String curso;  
    private PeriodoCurso periodoCurso;  
  
    public Turma(String curso, PeriodoCurso periodoCurso) {  
        this.curso = curso;  
        this.periodoCurso = periodoCurso;  
    }  
  
    public String getCurso() {  
        return curso;  
    }  
  
    public PeriodoCurso getPeriodoCurso() {  
        return periodoCurso;  
    }  
}
```

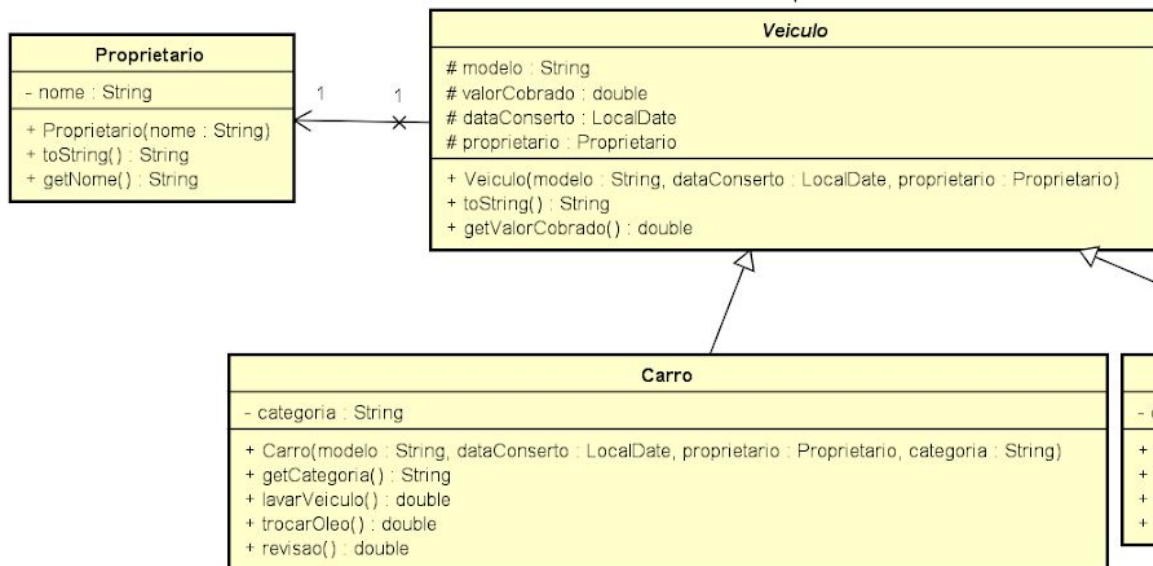
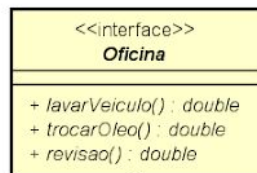
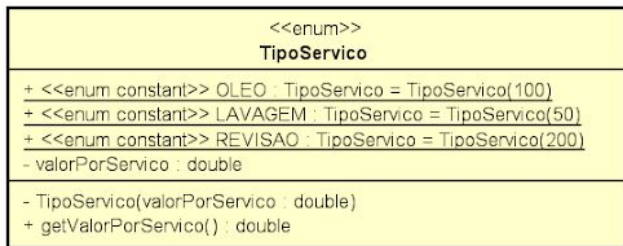

Inserindo a classe de teste

TestaTurma.java

```
package aula;

public class TestaTurma {
    public static void main(String[] args) {
        Turma turma = new Turma("Java", PeriodoCurso.INTEGRAL);

        System.out.println("Curso:" + turma.getCurso());
        System.out.println("Dias da Semana:" + turma.getPeriodoCurso().getDiasSemana());
        System.out.println("Carga Horária:" + turma.getPeriodoCurso().getCargaHoraria());
        System.out.println("Valor do Curso:" + turma.getPeriodoCurso().getValor());
    }
}
```



Montar o diagrama de classes abaixo da seguinte forma:

- Criar a interface **Oficina** com as assinaturas de métodos do diagrama.
- Criar a classe **Proprietario** com o atributo e métodos do diagrama.
- Criar a classe **Veiculo** sendo abstrata e implementando a interface **Oficina**.
- Criar a classe **Carro** herdando de **Veiculo**.
- Criar a classe **Moto** herdando de **Veiculo**.
- Criar o Enum **TipoServico** que armazenará os preços dos serviços prestados pela Oficina:

Troca de Óleo = 100

Lavar = 50

Revisao = 200

Métodos:

Carro: O atributo **valorCobrado** deverá acumular todos os serviços prestados.

TrocarOleo – Aos sábado o cliente terá desconto de 50,00 na troca de óleo.

Revisao – No mês de agosto o cliente terá 10% de desconto na revisão.

LavarVeiculo – O Valor cobrado será o definido no Enum TipoServico.

Métodos:

Moto:

TrocarOleo – O Valor cobrado será o definido no Enum TipoServico.

Revisao – O Valor cobrado será o definido no Enum TipoServico.

LavarVeiculo – O Valor cobrado será o definido no Enum TipoServico.

Classe com o main: Criar uma classe com o nome **TesteOficina** e realizar as seguintes operações:

Criar um objeto do tipo carro

Trocar o óleo e fazer a revisão

Exibir o nome do proprietário, os dados carro e o valor cobrado pelo serviço.

```
public interface Oficina {  
    public double lavarVeiculo();  
    public double trocarOleo();  
    public double revisao();  
}
```

```
public class Proprietario {  
    private String nome;  
  
    public Proprietario(String nome) {  
        super();  
        this.nome = nome;  
    }  
  
    @Override  
    public String toString() {  
        return "Proprietario [nome=" + nome + "]";  
    }  
  
    public String getNome() {  
        return nome;  
    }  
}
```

```

import java.time.LocalDate;

public abstract class Veiculo implements Oficina {
    protected String modelo;
    protected double valorCobrado;
    protected LocalDate dataConserto;
    protected Proprietario proprietario;

    public Veiculo(String modelo, LocalDate dataConserto, Proprietario proprietario) {
        super();
        this.modelo = modelo;
        this.dataConserto = dataConserto;
        this.proprietario = proprietario;
    }

    @Override
    public String toString() {
        return "Veiculo [modelo=" + modelo + ", valorCobrado=" + valorCobrado + "]";
    }

    public double getValorCobrado() {
        return valorCobrado;
    }
}

```

```

public final class Carro extends Veiculo {
    private String categoria;

    public Carro(String modelo, LocalDate dataConserto, Proprietario proprietario, String categoria) {
        super(modelo, dataConserto, proprietario);
        this.categoria = categoria;
    }

    public String getCategoria() {
        return categoria;
    }

    @Override
    public double lavarVeiculo() {
        return valorCobrado = valorCobrado + TipoServico.LAVAGEM.getValorPorServico();
    }

    @Override
    public double trocarOleo() {
        if (dataConserto.getDayOfWeek().name().equals("SATURDAY")) {
            valorCobrado = valorCobrado + TipoServico.OLEO.getValorPorServico() - 50;

        } else {
            valorCobrado = valorCobrado + TipoServico.OLEO.getValorPorServico();
        }
        return valorCobrado;
    }

    @Override
    public double revisao() {
        if (dataConserto.getDayOfMonth() == 8) {
            valorCobrado = valorCobrado + TipoServico.REVISAO.getValorPorServico() * 0.90;
        } else {
            valorCobrado = valorCobrado + TipoServico.REVISAO.getValorPorServico();
        }
        return valorCobrado;
    }
}

```



```
import java.time.LocalDate;

public final class Moto extends Veiculo{
    private int cilindradas;

    public Moto(String modelo, LocalDate dataConserto, Proprietario proprietario, int cilindradas) {
        super(modelo, dataConserto, proprietario);
        this.cilindradas = cilindradas;
    }

    @Override
    public double lavarVeiculo() {
        return valorCobrado = TipoServico.LAVAGEM.getValorPorServico();
    }

    @Override
    public double trocarOleo() {
        return valorCobrado = TipoServico.OLEO.getValorPorServico();
    }

    @Override
    public double revisao() {
        return valorCobrado = TipoServico.REVISAO.getValorPorServico();
    }
}
```

```

public enum TipoServico {
    OLEO(100), LAVAGEM(50), REVISAO(200);

    private TipoServico(double valorPorServico) {
        this.valorPorServico = valorPorServico;
    }

    private double valorPorServico;

    public double getValorPorServico() {
        return valorPorServico;
    }
}

```

```

public class TesteOficina {
    public static void main(String[] args) {
        Proprietario proprietario1 = new Proprietario("Joaquim");
        Veiculo veiculo1= new Carro("Sander", LocalDate.of(2020, 8, 8), proprietario1, "Hatch");

        veiculo1.trocarOleo();
        veiculo1.revisao();

        System.out.println("Proprietário:" + proprietario1.getNome());
        System.out.println("Veículo:" + veiculo1.toString());
    }
}

```