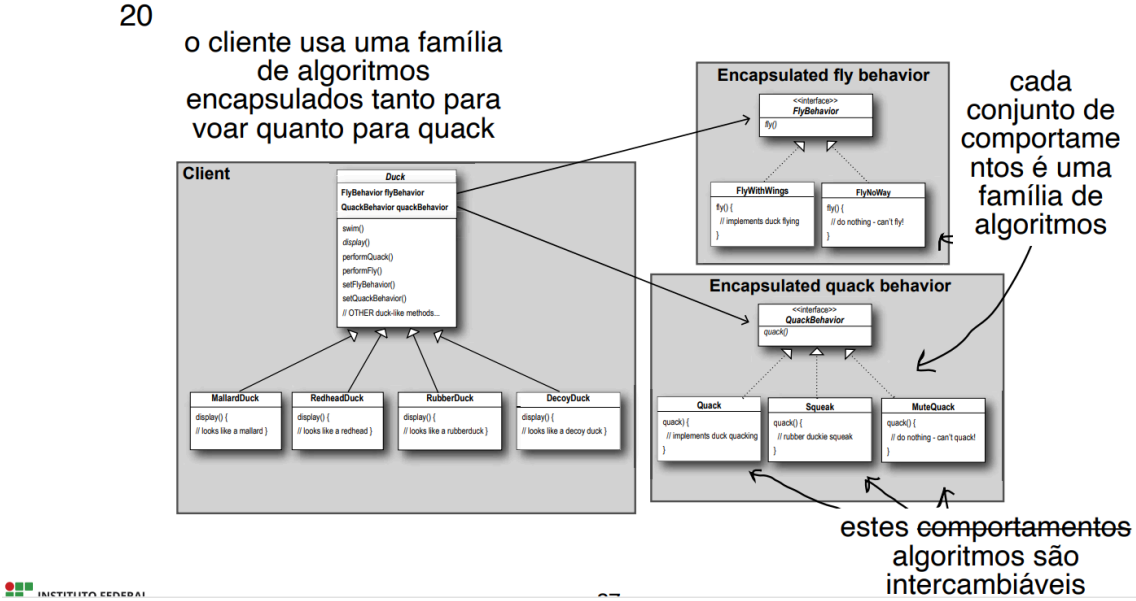
Instituto Federal Catarinense *(Campus Blumenau)*
Professor: Ricardo de La Rocha Ladeira
Matéria: Padrões de Projeto
Nomes: Gabrielli Danker
Turma: BCC 2025.1
Data de entrega: 20 de Março de 2025

Exercício

1) Resolva os três exercícios dos slides 27-29 de Gomes (s.d.).

   Exercício:

   - Implemente o exemplo do simulador de lagoa de patos mostrado no slide 20



   RESPOSTA:

```java
// Interface para comportamento de voo
interface FlyBehavior {
    void fly();
}

// Implementações de voo
class FlyWithWings implements FlyBehavior {
    public void fly() {
        System.out.println("Voando com asas!");
    }
}

class FlyNoWay implements FlyBehavior {
    public void fly() {
        System.out.println("Não posso voar.");
    }
}

// Interface para comportamento de grasnido
interface QuackBehavior {
    void quack();
}

// Implementações de grasnido
class Quack implements QuackBehavior {
    public void quack() {
        System.out.println("Quack! Quack!");
    }
}
```

```
class Squeak implements QuackBehavior {
    public void quack() {
        System.out.println("Som de brinquedo de borracha!");
    }
}

class MuteQuack implements QuackBehavior {
    public void quack() {
        System.out.println("...");
    }
}

// Classe abstrata Duck
abstract class Duck {
    FlyBehavior flyBehavior;
    QuackBehavior quackBehavior;

    public void performFly() {
        flyBehavior.fly();
    }

    public void performQuack() {
        quackBehavior.quack();
    }

    public void setFlyBehavior(FlyBehavior fb) {
        flyBehavior = fb;
    }

    public void setQuackBehavior(QuackBehavior qb) {
        quackBehavior = qb;
    }

    public void swim() {
        System.out.println("Todos os patos nadam, até os de brinquedo!");
    }

    public abstract void display();
}

// Diferentes tipos de patos
class MallardDuck extends Duck {
    public MallardDuck() {
        flyBehavior = new FlyWithWings();
        quackBehavior = new Quack();
    }
    public void display() {
        System.out.println("Eu sou um pato Mallard!");
    }
}

class RedheadDuck extends Duck {
    public RedheadDuck() {
        flyBehavior = new FlyWithWings();
        quackBehavior = new Quack();
    }
    public void display() {
        System.out.println("Eu sou um pato de cabeça vermelha!");
    }
```

```java
}

class RubberDuck extends Duck {
    public RubberDuck() {
        flyBehavior = new FlyNoWay();
        quackBehavior = new Squeak();
    }
    public void display() {
        System.out.println("Eu sou um pato de borracha!");
    }
}

class DecoyDuck extends Duck {
    public DecoyDuck() {
        flyBehavior = new FlyNoWay();
        quackBehavior = new MuteQuack();
    }
    public void display() {
        System.out.println("Eu sou um pato de madeira!");
    }
}

// Classe principal para testar o simulador
public class DuckSimulator {
    public static void main(String[] args) {
        Duck mallard = new MallardDuck();
        mallard.display();
        mallard.performQuack();
        mallard.performFly();


        System.out.println("\nMudando o comportamento de voo do pato Mallard...");
        mallard.setFlyBehavior(new FlyNoWay());
        mallard.performFly();
    }
}
```

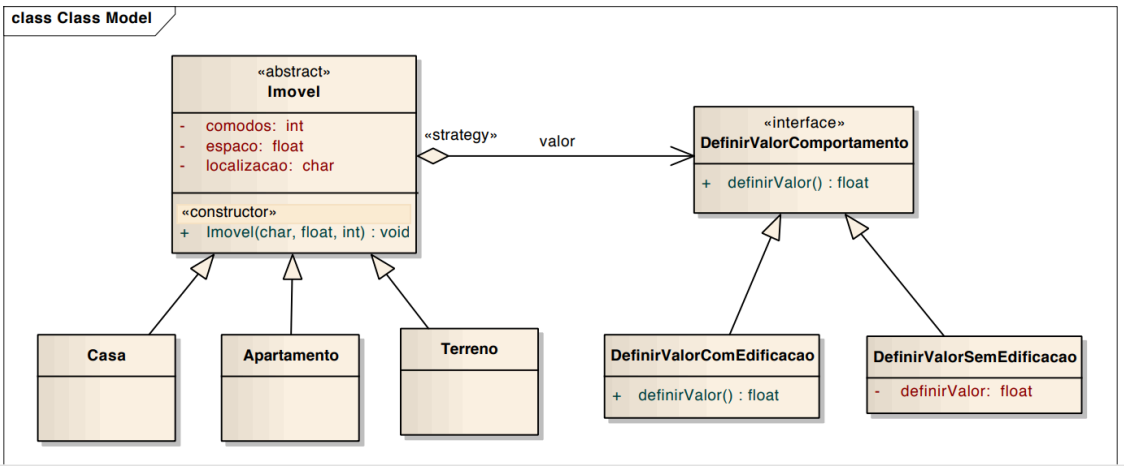
Este código retorna:

```
Eu sou um pato Mallard!
Quack! Quack!
Voando com asas!

Mudando o comportamento de voo do pato Mallard...
Não posso voar.
```

O código está na pasta Strategy -> DuckSimulator.java

Exercício:

- A Prefeitura quer definir o valor dos imóveis da cidade para posteriormente poder atribuir o valor do IPTU. Para isto ela dividiu a cidade em regiões (A, B e C). Casas e Aptos na região A tem um valor de 3000,00 o m2. Na região B o valor é de 1000,00 e na C 500,00. Para os Terrenos os valores definidos foram: região A = 1500,00; B = 750,00 e C = 200,00. Casas e Aptos são considerados imóveis com edificação e é acrescido 1000,00 por cômodo. Considerando o modelo de classes a seguir, que utiliza o padrão Estratégia, crie imóveis e informe o seu valor de venda.



RESPOSTA:

```java
// Interface para definir o comportamento de cálculo do valor do imóvel
interface DefinirValorComportamento {
    float definirValor(float espaco, int comodos, char localizacao);
}

// Implementação para imóveis com edificação
class DefinirValorComEdificacao implements DefinirValorComportamento {
    public float definirValor(float espaco, int comodos, char localizacao) {
        float precoBase = getPrecoPorM2(localizacao) * espaco;
        return precoBase + (comodos * 1000);
    }

    private float getPrecoPorM2(char localizacao) {
        return switch (localizacao) {
            case 'A' -> 3000;
            case 'B' -> 1000;
            case 'C' -> 500;
            default -> 0;
        };
    }
}

// Implementação para terrenos sem edificação
class DefinirValorSemEdificacao implements DefinirValorComportamento {
    public float definirValor(float espaco, int comodos, char localizacao) {
        return getPrecoPorM2(localizacao) * espaco;
    }

```

```
    private float getPrecoPorM2(char localizacao) {
        return switch (localizacao) {
            case 'A' -> 1500;
            case 'B' -> 750;
            case 'C' -> 200;
            default -> 0;
        };
    }
}

// Classe abstrata Imóvel
abstract class Imovel {
    protected int comodos;
    protected float espaco;
    protected char localizacao;
    protected DefinirValorComportamento estrategiaValor;

    public Imovel(char localizacao, float espaco, int comodos) {
        this.localizacao = localizacao;
        this.espaco = espaco;
        this.comodos = comodos;
    }

    public void setEstrategiaValor(DefinirValorComportamento estrategia) {
        this.estrategiaValor = estrategia;
    }

    public float calcularValor() {
        return estrategiaValor.definirValor(espaco, comodos, localizacao);
    }
}
```

```

// Classes concretas Casa, Apartamento e Terreno
class Casa extends Imovel {
    public Casa(char localizacao, float espaco, int comodos) {
        super(localizacao, espaco, comodos);
        this.estrategiaValor = new DefinirValorComEdificacao();
    }
}

class Apartamento extends Imovel {
    public Apartamento(char localizacao, float espaco, int comodos) {
        super(localizacao, espaco, comodos);
        this.estrategiaValor = new DefinirValorComEdificacao();
    }
}

class Terreno extends Imovel {
    public Terreno(char localizacao, float espaco) {
        super(localizacao, espaco, 0);
        this.estrategiaValor = new DefinirValorSemEdificacao();
    }
}

// Classe principal para testar o cálculo do valor dos imóveis
public class CalculadoraImoveis {
    public static void main(String[] args) {
        Imovel casa = new Casa('A', 100, 3);
        System.out.println("Valor da casa: R$ " + casa.calcularValor());

        Imovel apartamento = new Apartamento('B', 80, 2);
        System.out.println("Valor do apartamento: R$ " + apartamento.calcularValor());
```

Este código retorna:

```
Valor da casa: R$ 303000.0
Valor do apartamento: R$ 82000.0
Valor do terreno: R$ 40000.0
```

O código está na pasta Strategy -> CalculadoraImoveis.java

Exercício:

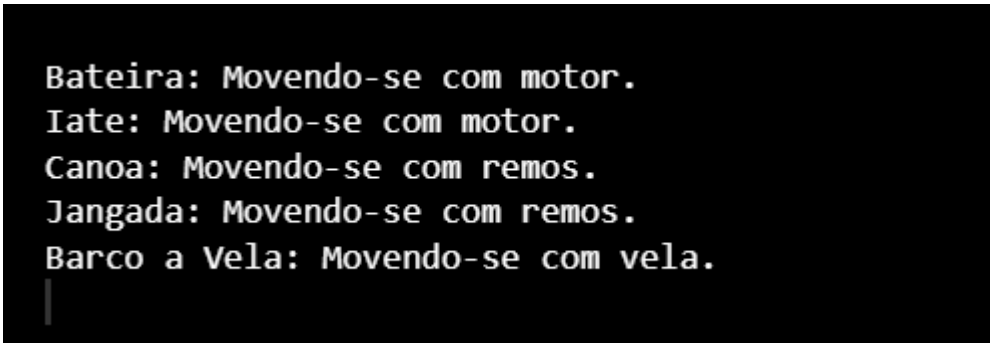
Exercícios Propostos

- Uma empresa de locação de barcos trabalha com os seguintes modelos: bateira, iate, canoa, jangada e barco a vela. Os dois primeiros modelos usam motor, os dois seguintes são movimentados a remos e o último à vela. Usando o padrão estratégia crie e implemente um modelo de classes que, dado um determinado barco, informe como ele está se movimentando.

RESPOSTA:

```java
// Interface para a estratégia de movimentação
interface Movimentacao {
    void mover();
}

// Implementações das estratégias
class MovimentacaoMotor implements Movimentacao {
    public void mover() {
        System.out.println("Movendo-se com motor.");
    }
}

class MovimentacaoRemos implements Movimentacao {
    public void mover() {
        System.out.println("Movendo-se com remos.");
    }
}

class MovimentacaoVela implements Movimentacao {
    public void mover() {
        System.out.println("Movendo-se com vela.");
    }
}

// Classe Barco
class Barco {
    private String nome;
    private Movimentacao movimentacao;

    public Barco(String nome, Movimentacao movimentacao) {
        this.nome = nome;
        this.movimentacao = movimentacao;
    }

    public void exibirMovimento() {
        System.out.print(nome + ": ");
        movimentacao.mover();
    }
}

// Classe principal para teste
public class LocacaoBarcos {
    public static void main(String[] args) {
        Barco bateira = new Barco("Bateira", new MovimentacaoMotor());
        Barco iate = new Barco("Iate", new MovimentacaoMotor());
        Barco canoa = new Barco("Canoa", new MovimentacaoRemos());
        Barco jangada = new Barco("Jangada", new MovimentacaoRemos());
        Barco barcoAVela = new Barco("Barco a Vela", new MovimentacaoVela());

        bateira.exibirMovimento();
        iate.exibirMovimento();
        canoa.exibirMovimento();
        jangada.exibirMovimento();
        barcoAVela.exibirMovimento();
    }
}
```

Retorna isto:



O código está na pasta Strategy -> LocacaoBarcos.java