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

Exercícios

1) Faça os dois [exercícios do slide 28 de GOMES, s.d]().

1. Implemente o padrão Decorator no exemplo Starbuzz Cofee dos slides anteriores (diagrama do slide no. 21)

```java
public abstract class Beverage {
   String description = "Unknown Beverage";

   public String getDescription() {
       return description;
   }

   public abstract double cost();
}

public abstract class CondimentDecorator extends Beverage {
   public abstract String getDescription();
}

public class Espresso extends Beverage {
   public Espresso() {
       description = "Espresso";
   }

   public double cost() {
       return 1.99;
   }
}

public class HouseBlend extends Beverage {
   public HouseBlend() {
       description = "House Blend Coffee";
   }

   public double cost() {
       return 0.89;
```

```
    }
}

public class DarkRoast extends Beverage {
    public DarkRoast() {
        description = "Dark Roast";
    }

    public double cost() {
        return 0.99;
    }
}

public class Mocha extends CondimentDecorator {
    Beverage beverage;

    public Mocha(Beverage beverage) {
        this.beverage = beverage;
    }

    public String getDescription() {
        return beverage.getDescription() + ", Mocha";
    }

    public double cost() {
        return 0.20 + beverage.cost();
    }
}

public class Whip extends CondimentDecorator {
    Beverage beverage;

    public Whip(Beverage beverage) {
        this.beverage = beverage;
    }

    public String getDescription() {
        return beverage.getDescription() + ", Whip";
    }

    public double cost() {
        return 0.10 + beverage.cost();
    }
```

```java
}

public class Soy extends CondimentDecorator {
   Beverage beverage;

   public Soy(Beverage beverage) {
       this.beverage = beverage;
   }

   public String getDescription() {
       return beverage.getDescription() + ", Soy";
   }

   public double cost() {
       return 0.15 + beverage.cost();
   }
}

public class StarbuzzCoffee {
   public static void main(String args[]) {
       Beverage beverage = new Espresso();
              System.out.println(beverage.getDescription() + " $" +
beverage.cost());

       Beverage beverage2 = new DarkRoast();
       beverage2 = new Mocha(beverage2);
       beverage2 = new Mocha(beverage2);
       beverage2 = new Whip(beverage2);
              System.out.println(beverage2.getDescription() + " $" +
beverage2.cost());

       Beverage beverage3 = new HouseBlend();
       beverage3 = new Soy(beverage3);
       beverage3 = new Mocha(beverage3);
       beverage3 = new Whip(beverage3);
              System.out.println(beverage3.getDescription() + " $" +
beverage3.cost());
   }
}
```

2. Altere a implementação para prover as seguintes funcionalidades:

“StarBuzz introduziu diferentes tamanhos de bebida em seu cardápio. Agora é possível pedir café em tamanhos pequeno, médio e grande. A classe Beverage deve implementar métodos setSize() e getSize(). Também será necessário alterar o preço de complementos de acordo com o tamanho: por exemplo, leite de soja (Soy) deve custar 10, 15 e 20 centavos para tamanho P, M e G, respectivamente.”

```java
import java.util.EnumSet;

enum Size {
   SMALL(0), MEDIUM(1), LARGE(2);

   private final int value;

   Size(int value) {
       this.value = value;
   }

   public int getValue() {
       return value;
   }
}

abstract class Beverage {
   protected Size size;
   protected String description;
   protected int dosesOfCoffee;

   public Beverage() {
       this.size = Size.SMALL;
       this.description = "Unknown Beverage";
       this.dosesOfCoffee = 1;
   }

   public void setSize(Size size) {
       this.size = size;
   }

   public Size getSize() {
       return size;
   }
```

```
    public void setDosesOfCoffee(int doses) {
        this.dosesOfCoffee = doses;
    }

    public int getDosesOfCoffee() {
        return dosesOfCoffee;
    }

    public String getDescription() {
        return description;
    }

    public abstract double cost();
}

class Espresso extends Beverage {
    public Espresso() {
        description = "Espresso";
    }

    @Override
    public double cost() {
        return 1.99;
    }
}

class HouseBlend extends Beverage {
    public HouseBlend() {
        description = "House Blend Coffee";
    }

    @Override
    public double cost() {
        return 0.89;
    }
}

abstract class CondimentDecorator extends Beverage {
    public abstract String getDescription();
}

class Milk extends CondimentDecorator {
    private Beverage beverage;
```

```java
    public Milk(Beverage beverage) {
        this.beverage = beverage;
    }

    @Override
    public double cost() {
        double cost = beverage.cost();
        switch (beverage.getSize()) {
            case SMALL:
                cost += 0.10;
                break;
            case MEDIUM:
                cost += 0.15;
                break;
            case LARGE:
                cost += 0.20;
                break;
        }
        return cost;
    }

    @Override
    public String getDescription() {
        return beverage.getDescription() + ", Milk";
    }
}

class Soy extends CondimentDecorator {
    private Beverage beverage;

    public Soy(Beverage beverage) {
        this.beverage = beverage;
    }

    @Override
    public double cost() {
        double cost = beverage.cost();
        switch (beverage.getSize()) {
            case SMALL:
                cost += 0.10;
                break;
            case MEDIUM:
                cost += 0.15;
                break;
            case LARGE:
                cost += 0.20;
                break;
        }
        return cost;
    }

    @Override
    public String getDescription() {
        return beverage.getDescription() + ", Soy";
    }
}

class Mocha extends CondimentDecorator {
    private Beverage beverage;
    private int doses;

    public Mocha(Beverage beverage, int doses) {
        this.beverage = beverage;
        this.doses = doses;
    }

    @Override
    public double cost() {
        return 0.20 * doses + beverage.cost();
    }

    @Override
    public String getDescription() {
          return beverage.getDescription() + String.format(", Mocha (%d
doses)", doses);
    }
}

class Whip extends CondimentDecorator {
    private Beverage beverage;

    public Whip(Beverage beverage) {
        this.beverage = beverage;
    }
```

```java
   @Override
   public double cost() {
       double cost = beverage.cost();
       switch (beverage.getSize()) {
           case SMALL:
               cost += 0.10;
               break;
           case MEDIUM:
               cost += 0.15;
               break;
           case LARGE:
               cost += 0.20;
               break;
       }
       return cost;
   }

   @Override
   public String getDescription() {
       return beverage.getDescription() + ", Whip";
   }
}

public class CoffeeShop {
   public static void main(String[] args) {
       Beverage beverage = new Espresso();
       beverage.setSize(Size.SMALL);
       beverage.setDosesOfCoffee(2);
             System.out.println(beverage.getDescription()  +  "  $  "  +
beverage.cost());

       Beverage beverage2 = new HouseBlend();
       beverage2.setSize(Size.MEDIUM);
       beverage2.setDosesOfCoffee(3);
       beverage2 = new Milk(beverage2);
       beverage2 = new Mocha(beverage2, 2);
       beverage2 = new Whip(beverage2);
             System.out.println(beverage2.getDescription()  +  "  $  "  +
beverage2.cost());
   }
}
```