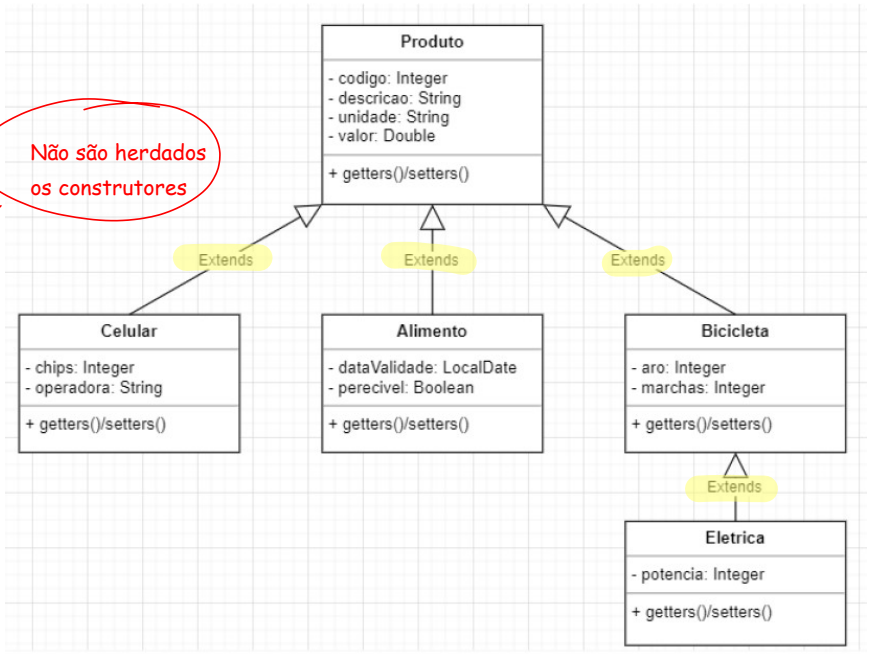
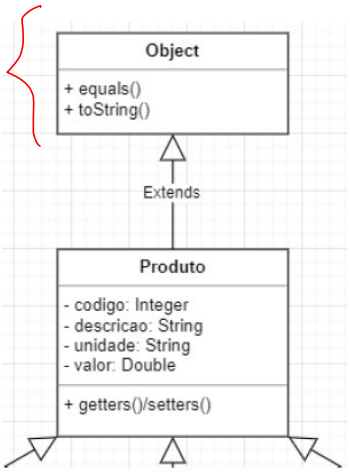
Herança:




```
1:  public class Celular extends Produto {
2:
3:     private Integer chips;
4:     private String operadora;
5:
6:     public Celular(String descricao, Double valor, Integer chips, String operadora) {
7:        super(descricao, valor); construtor da classe Produto
8:        this.chips = chips;
9:        this.operadora = operadora;
10:    }
11:    //getters e setters
12: }
```

Classe Object:



- `toString()`: dá uma representação em forma de texto (String) do objeto;
- `equals()`: permite a comparação entre Objetos do mesmo tipo.

```java
public class TesteProduto02 {

   public static void main(String[] args) {
      Produto celular01 = new Celular("Motorola M16", 1_600.00, 2, "Claro");
      Produto celular02 = new Celular("Motorola M16", 1_600.00, 2, "Claro");

      System.out.printf("Celular1: %s \nCelular2: %s\n", celular01, celular02);

      if (celular01.equals(celular02)) {
         System.out.println("\nIguais");
      } else {
         System.out.println("\nDiferentes");
      }
      /*
       Output:
         Celular1: br.com.kasolution.dominio.Celular@7ac7a4e4
         Celular2: br.com.kasolution.dominio.Celular@6d78f375

         Diferentes
       */

   }

}
```

celular01.toString();
celular02.toString();

Atenção: Anotação @Override

A anotação @Override não é obrigatória, porém, previne erros de sobrescrita de métodos herdados.

```java
   @Override
   public String toString() {
      StringBuilder info = new StringBuilder("Produto { ")
            .append("\ncodigo: ").append(codigo)
            .append(",\ndescricao: ").append(descricao)
            .append(",\nunidade: ").append(unidade)
            .append(",\nvalor: ").append(getCurrencyInstance().format(valor))
            .append("}");
      return info.toString();
   }
```

```java
   @Override
   public boolean equals(Object obj) {
      if (null != obj && obj instanceof Produto) {
         Produto produto = (Produto) obj; //cast para o tipo Produto
         if (this.descricao.equals(produto.descricao) &&
               this.unidade.equals(produto.unidade) &&
               this.valor.equals(produto.valor)) {
            return true;
         }
      }
      return false;
   }
```

celular01.equals(null);
celular01.equals("Farofa!");

1: Object p1 = new Celular();
2: Produto p2 = new Celular();
3: Celular p3 = new Celular();
4: Alimento a1 = new Celular();

Celular é um *** ?

instanceof

5: sysout(p1.getDescricao());
6: sysout(p2.getDescricao());
7: sysout(p3.getDescricao());

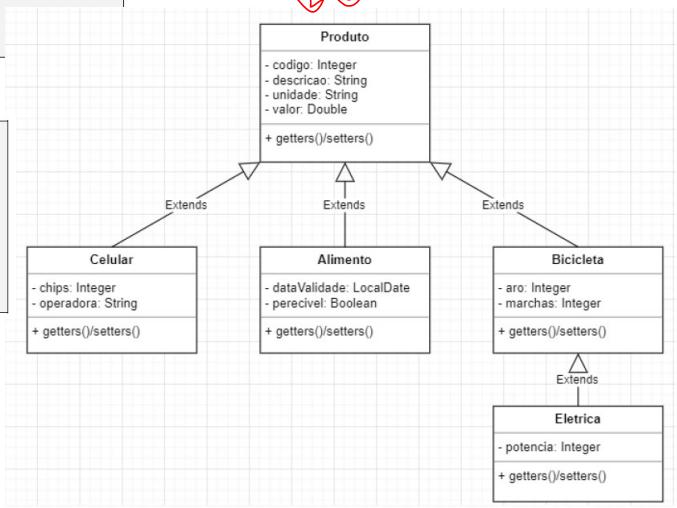
```java
@Override
public String toString() {
    StringBuilder info = new StringBuilder("Produto { ")
            .append("\ncodigo: ").append(codigo)
            .append(",\ndescricao: ").append(descricao)
            .append(",\nunidade: ").append(unidade)
            .append(",\nvalor: ").append(getCurrencyInstance().format(valor))
            .append("}");
    return info.toString();
}
```

public abstract String getTipo();

```java
@Override
public String toString() {
    StringBuilder infoSuper = new StringBuilder(super.toString());
    StringBuilder infoSub = new StringBuilder(",\nchips: ").append(chips)
            .append(",\noperadora: ").append(operadora);

    infoSuper.insert(infoSuper.length()-1, infoSub);

    return infoSuper.toString();
}
```




```java
public abstract class Produto {
    //atributos
    //construtores
    //outros métodos
    public abstract String getTipo();
}
```

Atenção: Classe abstrata

Uma classe abstrata não pode ser instanciada com o **new**. Caso seu modelo necessite de uma instancia da classe que possui métodos abstratos, reveja a modelagem e tente abstrair mais um nível.

2.7 - Atividade 01