# Trabalho sistema Pizzaria

Netson Cavina

Classes:

- Principal
- View
- Pizza
- CarrinhoDeCompras

## Pizza

Funções e métodos:

- Pizza( )
- Pizza(String ingrediente)
- void contabilizaIngrediente(String nome)
- List<String> getIngredientes( )
- String imprimeTodosIngredientes( )
- void adicionaIngrediente(String nome)
- int getPreco( )

## Carrinho de Compras

Funções e métodos:

- CarrinhoDeCompras( )
- CarrinhoDeCompras(Pizza pizza)
- addPizza(Pizza pizza)
- imprimeTotal( )

### Refatoração:

Construtor antigo permitia somente um ingrediente de cada vez, sendo necessário a utilização do metodo adicionaIngrediente(String nome) para adicionar outros ingredientes.

```
public Pizza(String ingrediente) {
      adicionaIngrediente(ingrediente);
  }
```

Mudando a função para receber mais argumentos e utilizando um *for* para adição dos ingredientes, podemos melhorar essa função, evitando repetição de código

```
public Pizza(String... ingredientes) {
    for(String ingrediente: ingredientes) {
      adicionaIngrediente(ingrediente);
    }
  }
```

## Testes

### ***Pizza***

Foi declarada uma pizza como objeto global na classe, para evitar repetição de código

```
Pizza p1 = new Pizza("Queijo","Presunto","Cebola");
```

- Teste para verificar se os ingredientes estão sendo adicionados

```
@Test
  void testeAdicionaIngredientes() {
    Pizza p2 = new Pizza();
    p2.adicionaIngrediente("Queijo");
    assertEquals(true,p2.getIngredientes().contains("Queijo"));
  }
```

- Teste para verificar se os ingredientes estão sendo retornados

```
@Test
  void testeGetIngredientes() {
    List<String> ingredientes = new ArrayList<String>();
    ingredientes.add("Queijo");
    ingredientes.add("Presunto");
    ingredientes.add("Cebola");
    assertEquals(ingredientes, p1.getIngredientes());
  }
```

- Teste para verificar se o preço está sendo calculado corretamente

```
@Test
  void testeGetPreco() {
    assertEquals(20, p1.getPreco());
  }
```

***Carrinho de Compras***

Foi declarado uma pizza e um carrinho de compras como objetos globais na classe, para evitar repetição de código.

```
Pizza p1 = new Pizza("Queijo","Presunto","Cebola");
CarrinhoDeCompras carrinho = new CarrinhoDeCompras(p1);
```

- Teste para verificar se as pizzas estão sendo retornadas corretamente

```
@Test
  void testGetPizza() {
    List<Pizza> pizzas = new ArrayList<Pizza>();
    pizzas.add(p1);
    assertEquals(pizzas, carrinho.getPizzas());
  }
```

- Teste para verificar se o valor do carrinho está sendo calculado corretamente

```
@Test
  void testImprimeTotal() {
    assertEquals("20",carrinho.imprimeTotal());
  }
```