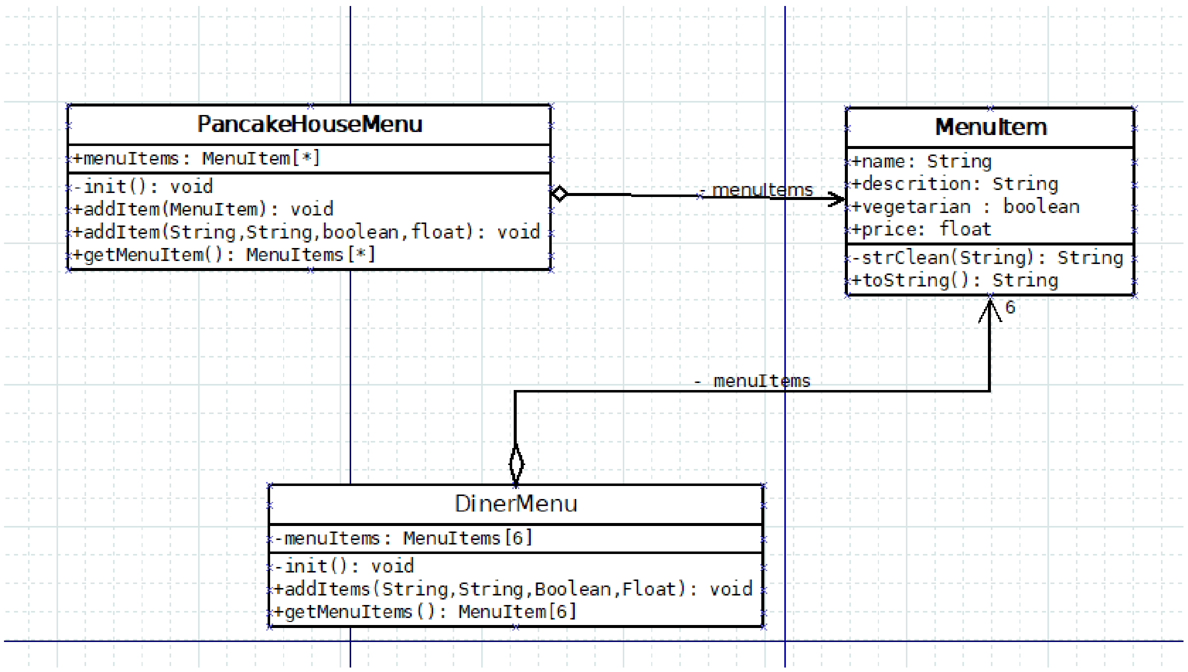
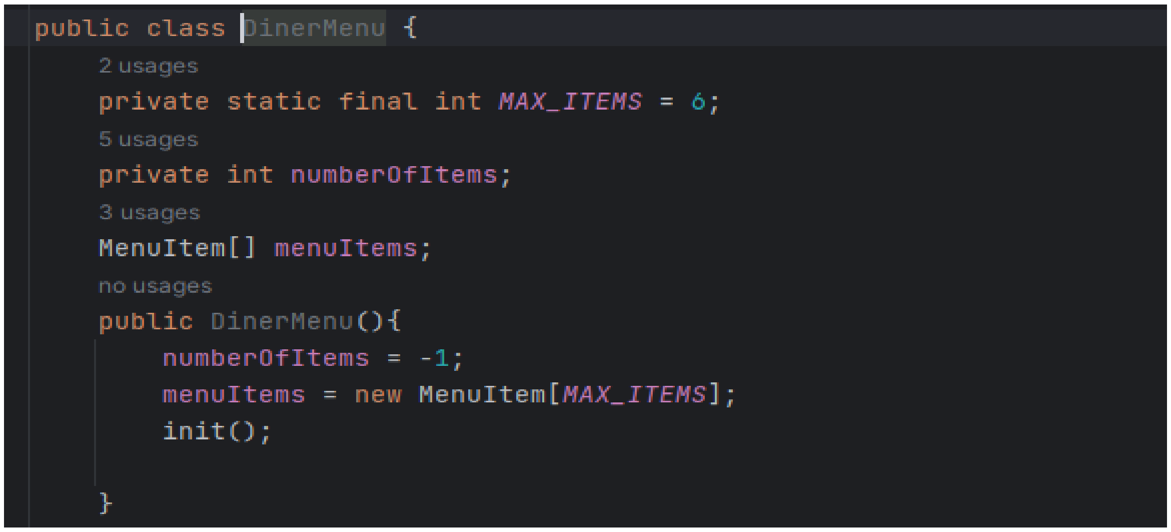
Patron Iterador

Diagrama:

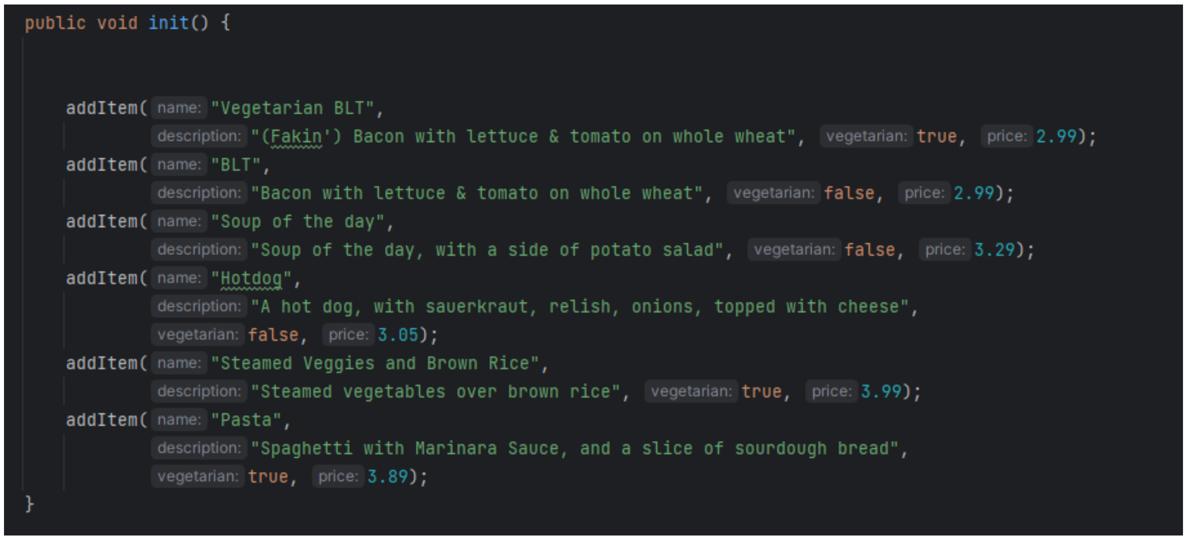


Código:

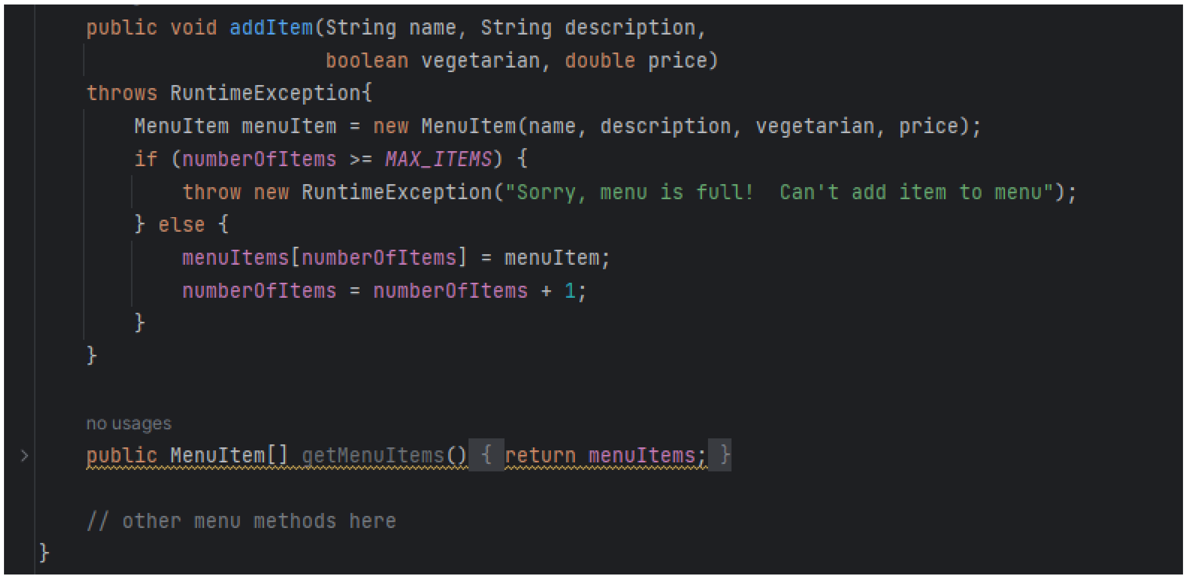
DinerMenu



```
public class DinerMenu {
    2 usages
    private static final int MAX_ITEMS = 6;
    5 usages
    private int numberOfItems;
    3 usages
    MenuItem[] menuItems;
    no usages
    public DinerMenu(){
        numberOfItems = -1;
        menuItems = new MenuItem[MAX_ITEMS];
        init();

    }
```

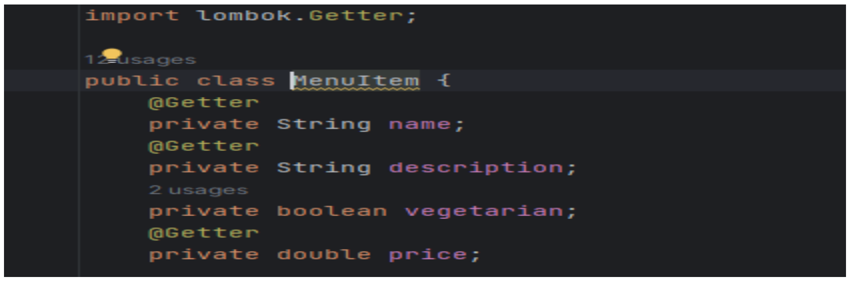


```java
public void init() {


    addItem( name: "Vegetarian BLT",
            description: "(Fakin') Bacon with lettuce & tomato on whole wheat",  vegetarian: true,  price: 2.99);
    addItem( name: "BLT",
            description: "Bacon with lettuce & tomato on whole wheat",  vegetarian: false,  price: 2.99);
    addItem( name: "Soup of the day",
            description: "Soup of the day, with a side of potato salad",  vegetarian: false,  price: 3.29);
    addItem( name: "Hotdog",
            description: "A hot dog, with sauerkraut, relish, onions, topped with cheese",
            vegetarian: false,  price: 3.05);
    addItem( name: "Steamed Veggies and Brown Rice",
            description: "Steamed vegetables over brown rice",  vegetarian: true,  price: 3.99);
    addItem( name: "Pasta",
            description: "Spaghetti with Marinara Sauce, and a slice of sourdough bread",
            vegetarian: true,  price: 3.89);
}
```

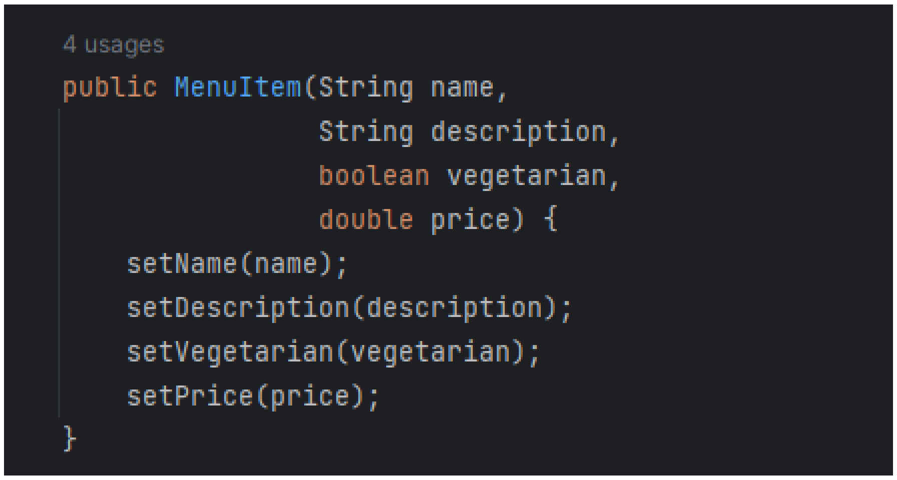


```java
    public void addItem(String name, String description,
                        boolean vegetarian, double price)
    throws RuntimeException{
        MenuItem menuItem = new MenuItem(name, description, vegetarian, price);
        if (numberOfItems >= MAX_ITEMS) {
            throw new RuntimeException("Sorry, menu is full!  Can't add item to menu");
        } else {
            menuItems[numberOfItems] = menuItem;
            numberOfItems = numberOfItems + 1;
        }
    }

    no usages
    public MenuItem[] getMenuItems() { return menuItems; }

    // other menu methods here
}
```

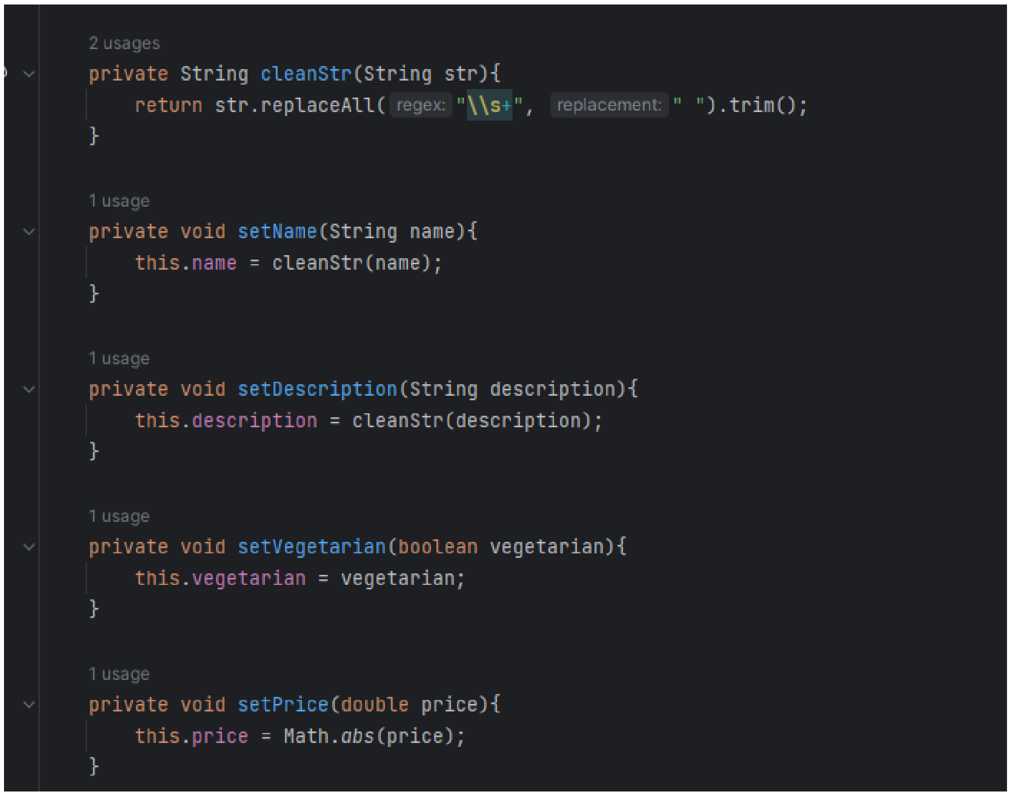
MenuItem



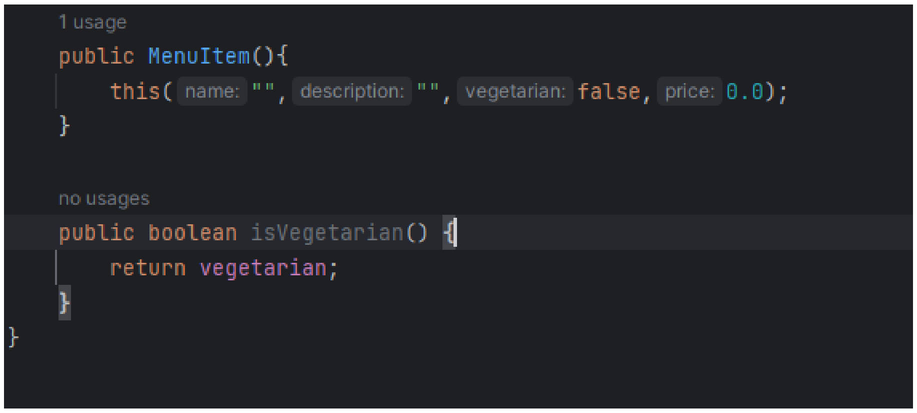
```java
import lombok.Getter;

12 usages
public class MenuItem {
    @Getter
    private String name;
    @Getter
    private String description;
    2 usages
    private boolean vegetarian;
    @Getter
    private double price;
```



```
4 usages
public MenuItem(String name,
                String description,
                boolean vegetarian,
                double price) {
    setName(name);
    setDescription(description);
    setVegetarian(vegetarian);
    setPrice(price);
}
```

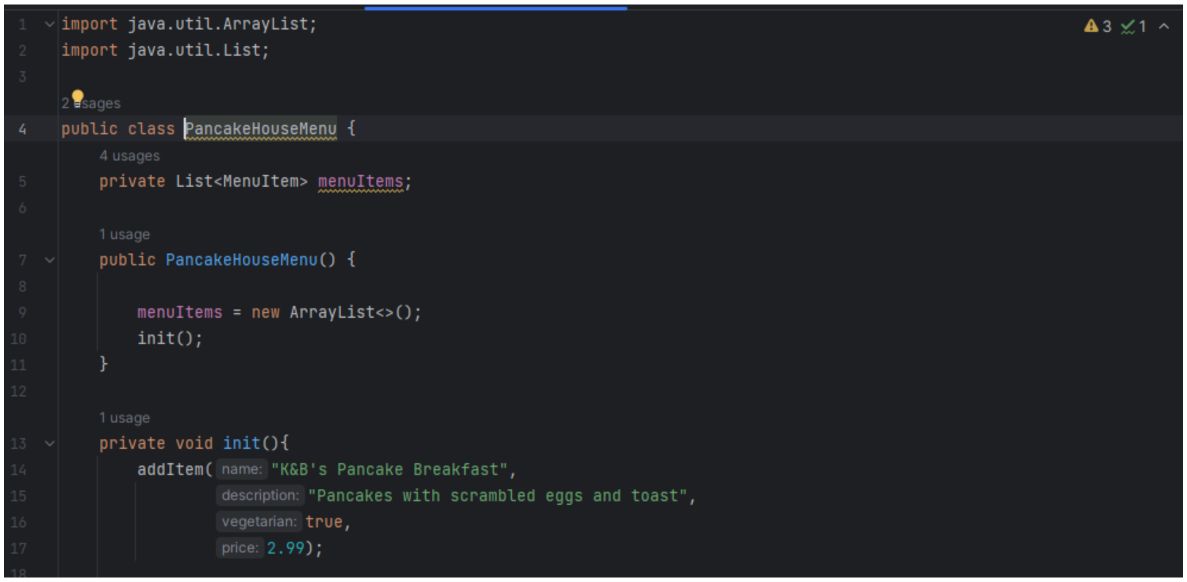


```
2 usages
private String cleanStr(String str){
    return str.replaceAll( regex: "\\s+",  replacement: " ").trim();
}

1 usage
private void setName(String name){
    this.name = cleanStr(name);
}

1 usage
private void setDescription(String description){
    this.description = cleanStr(description);
}

1 usage
private void setVegetarian(boolean vegetarian){
    this.vegetarian = vegetarian;
}

1 usage
private void setPrice(double price){
    this.price = Math.abs(price);
}
```

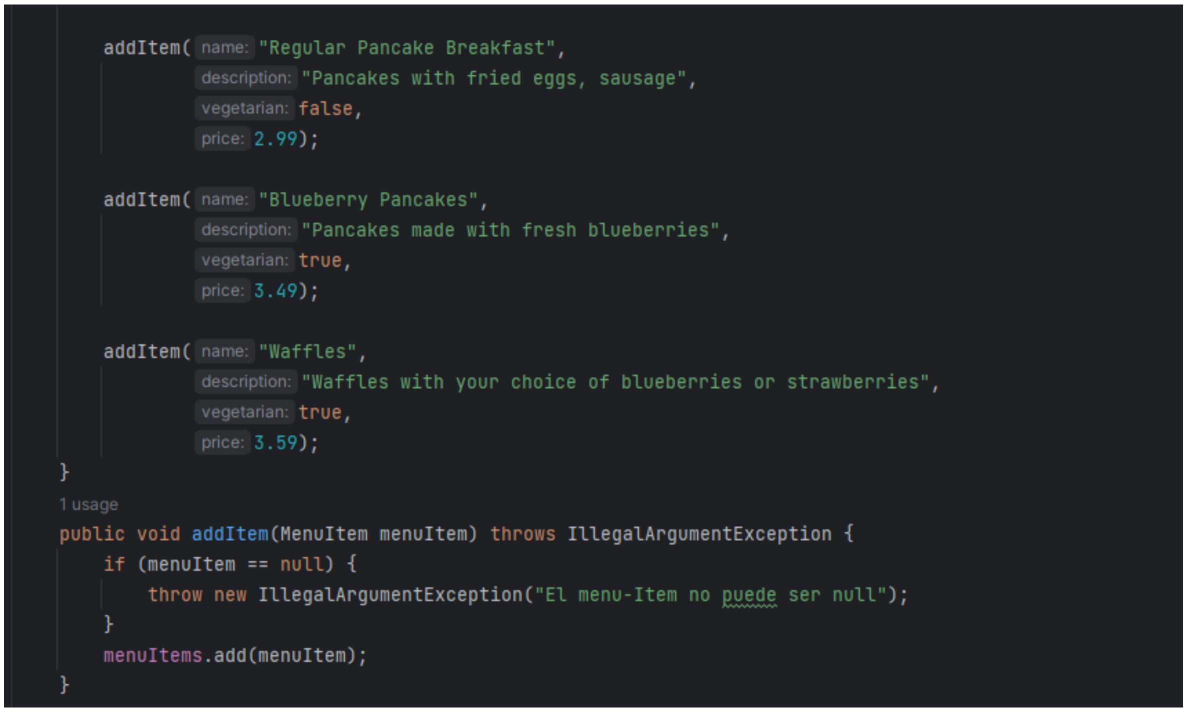


```
    1 usage
    public MenuItem(){
        this( name: "", description: "", vegetarian: false, price: 0.0);
    }

    no usages
    public boolean isVegetarian() {
        return vegetarian;
    }
}
```

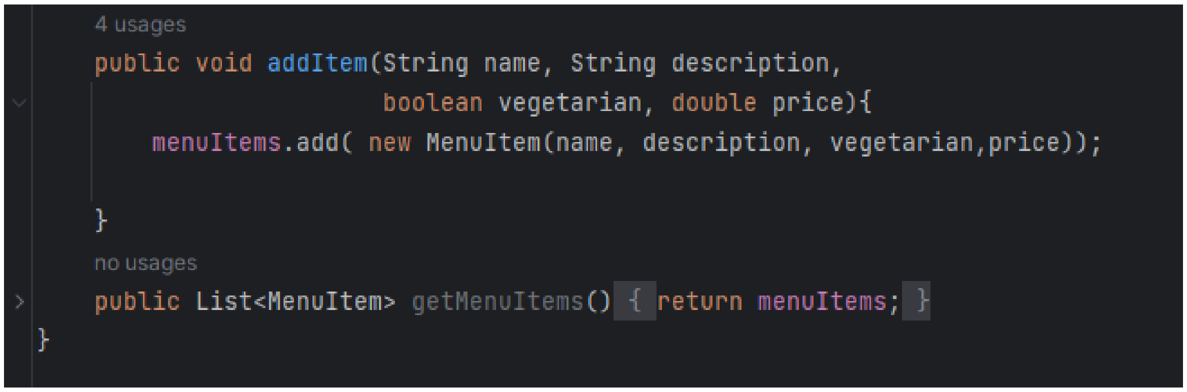
PancakeHouseMenu



```java
import java.util.ArrayList;
import java.util.List;

2 usages
public class PancakeHouseMenu {
    4 usages
    private List<MenuItem> menuItems;

    1 usage
    public PancakeHouseMenu() {

        menuItems = new ArrayList<>();
        init();
    }

    1 usage
    private void init(){
        addItem( name: "K&B's Pancake Breakfast",
                description: "Pancakes with scrambled eggs and toast",
                vegetarian: true,
                price: 2.99);
```

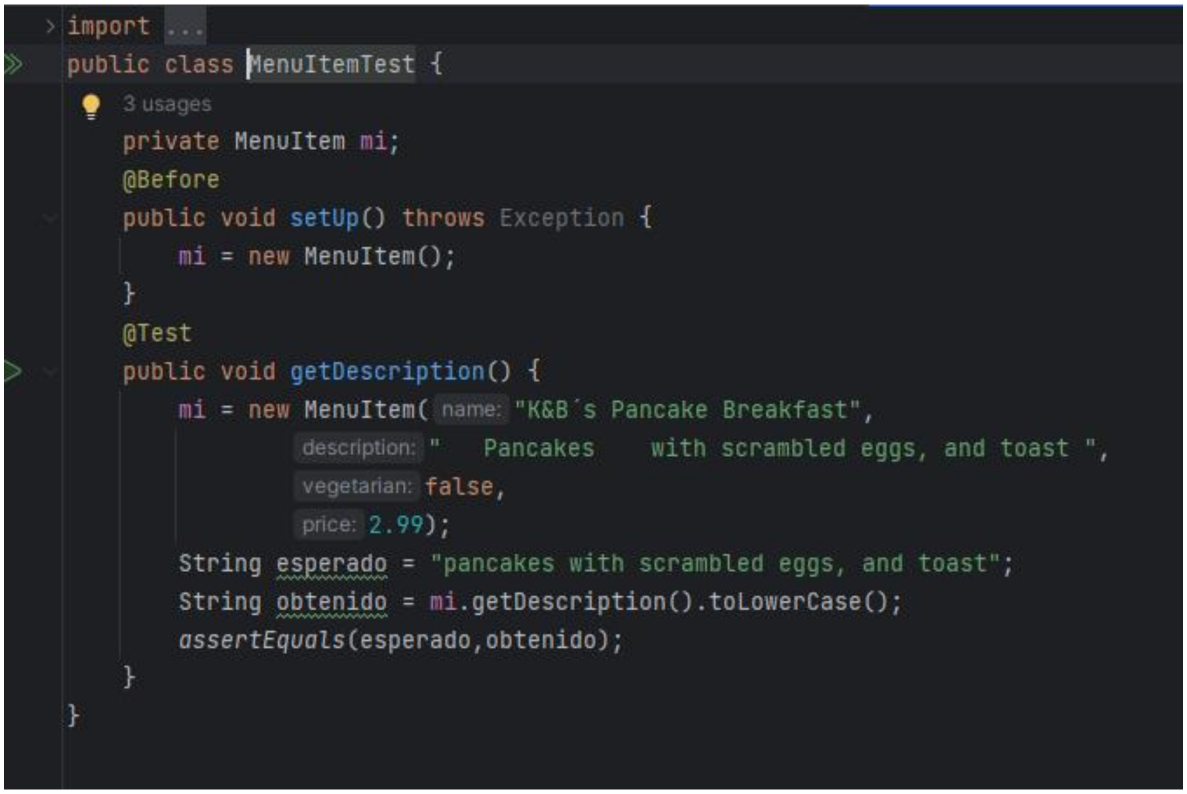


```java
        addItem( name: "Regular Pancake Breakfast",
                description: "Pancakes with fried eggs, sausage",
                vegetarian: false,
                price: 2.99);

        addItem( name: "Blueberry Pancakes",
                description: "Pancakes made with fresh blueberries",
                vegetarian: true,
                price: 3.49);

        addItem( name: "Waffles",
                description: "Waffles with your choice of blueberries or strawberries",
                vegetarian: true,
                price: 3.59);
    }
    1 usage
    public void addItem(MenuItem menuItem) throws IllegalArgumentException {
        if (menuItem == null) {
            throw new IllegalArgumentException("El menu-Item no puede ser null");
        }
        menuItems.add(menuItem);
    }
```

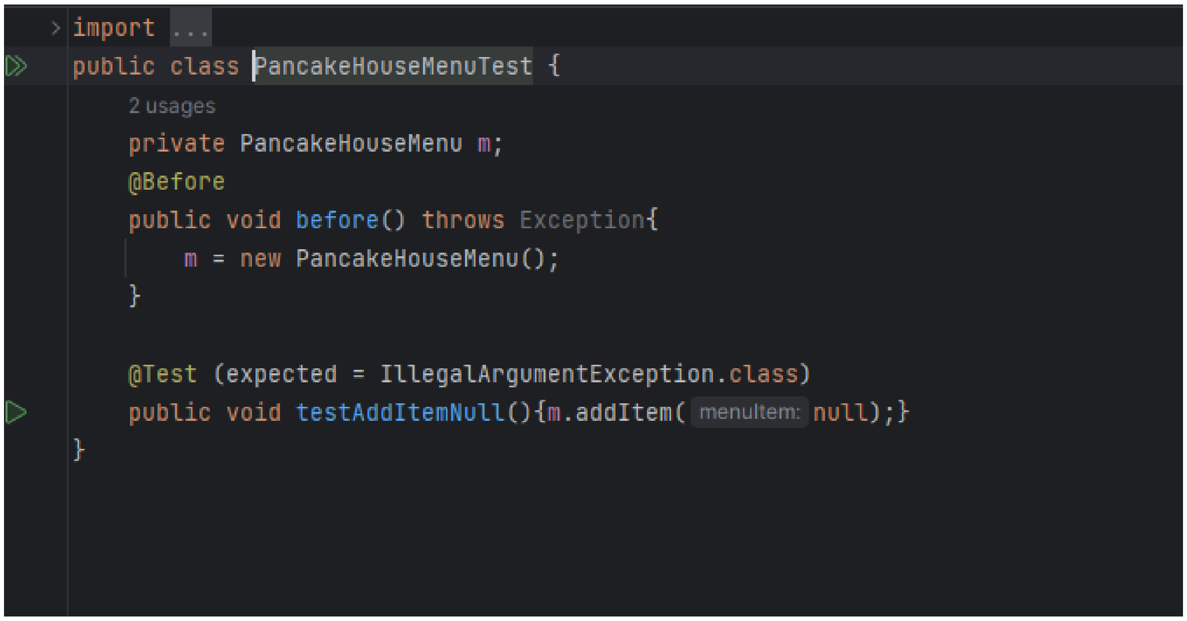


```
    4 usages
    public void addItem(String name, String description,
                        boolean vegetarian, double price){
        menuItems.add( new MenuItem(name, description, vegetarian,price));

    }
    no usages
    public List<MenuItem> getMenuItems() { return menuItems; }
}
```

Test:



```
import ...
public class MenuItemTest {
    3 usages
    private MenuItem mi;
    @Before
    public void setUp() throws Exception {
        mi = new MenuItem();
    }
    @Test
    public void getDescription() {
        mi = new MenuItem( name: "K&B´s Pancake Breakfast",
                description: "   Pancakes    with scrambled eggs, and toast ",
                vegetarian: false,
                price: 2.99);
        String esperado = "pancakes with scrambled eggs, and toast";
        String obtenido = mi.getDescription().toLowerCase();
        assertEquals(esperado,obtenido);
    }
}
```



```
import ...
public class PancakeHouseMenuTest {
    2 usages
    private PancakeHouseMenu m;
    @Before
    public void before() throws Exception{
        m = new PancakeHouseMenu();
    }

    @Test (expected = IllegalArgumentException.class)
    public void testAddItemNull(){m.addItem( menuItem: null);}
}
```