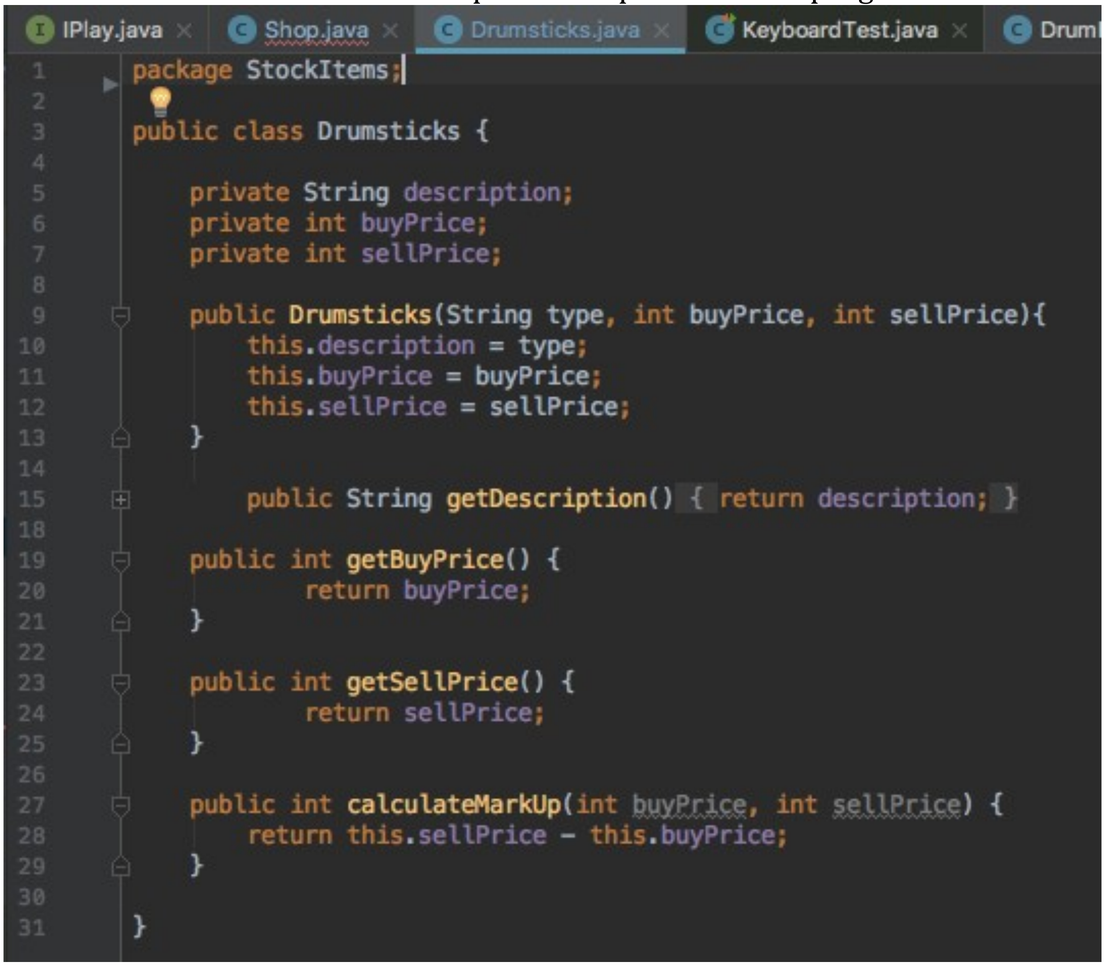
Evidence for Implementation and Testing Unit

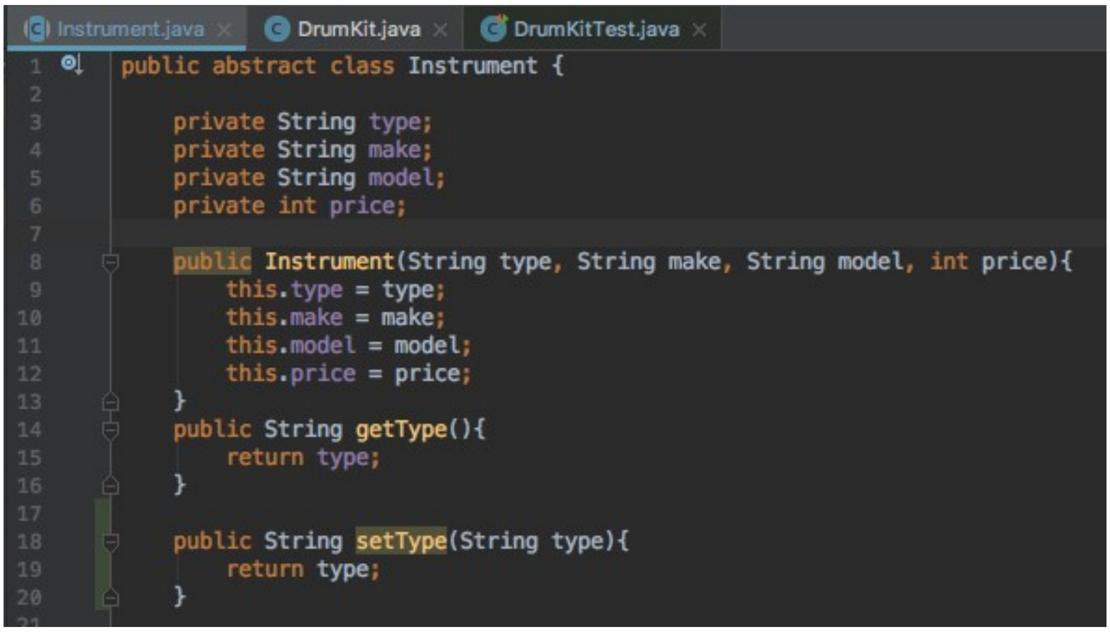
Oliver Berry
Edinburgh 19

I.T 1 Take a screenshot of an example of encapsulation in a program.



```java
package StockItems;

public class Drumsticks {

    private String description;
    private int buyPrice;
    private int sellPrice;

    public Drumsticks(String type, int buyPrice, int sellPrice){
        this.description = type;
        this.buyPrice = buyPrice;
        this.sellPrice = sellPrice;
    }

        public String getDescription() { return description; }

    public int getBuyPrice() {
            return buyPrice;
    }

    public int getSellPrice() {
            return sellPrice;
    }

    public int calculateMarkUp(int buyPrice, int sellPrice) {
        return this.sellPrice - this.buyPrice;
    }

}
```

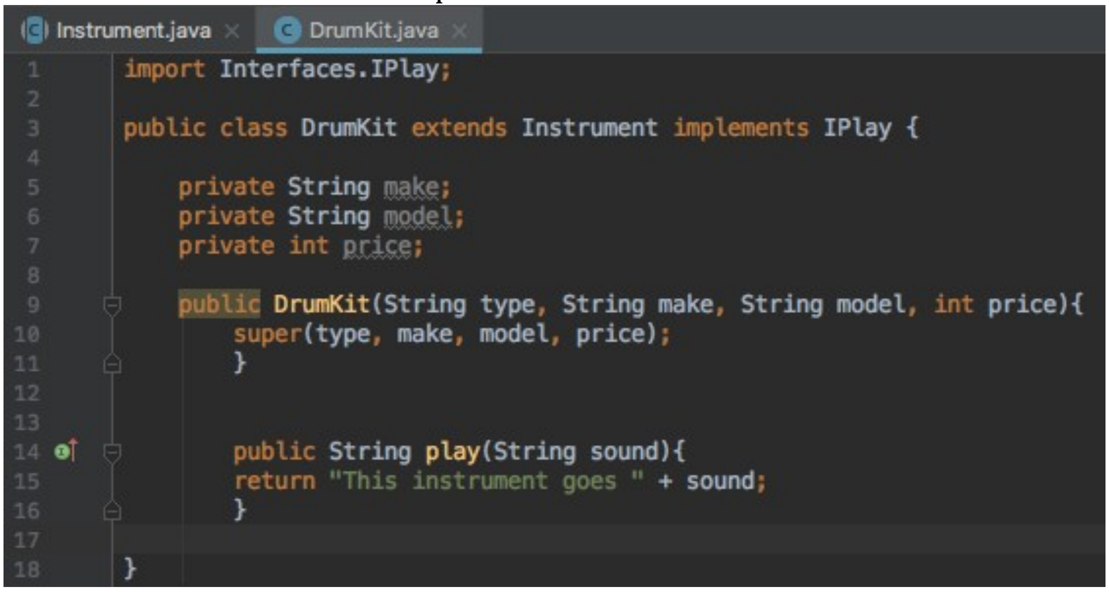
I.T 2 Take a screenshot of the use of Inheritance in a program.
Take screenshots of:
● A Class



Instrument.java × DrumKit.java × DrumKitTest.java ×

```java
public abstract class Instrument {

    private String type;
    private String make;
    private String model;
    private int price;

    public Instrument(String type, String make, String model, int price){
        this.type = type;
        this.make = make;
        this.model = model;
        this.price = price;
    }
    public String getType(){
        return type;
    }

    public String setType(String type){
        return type;
    }
```

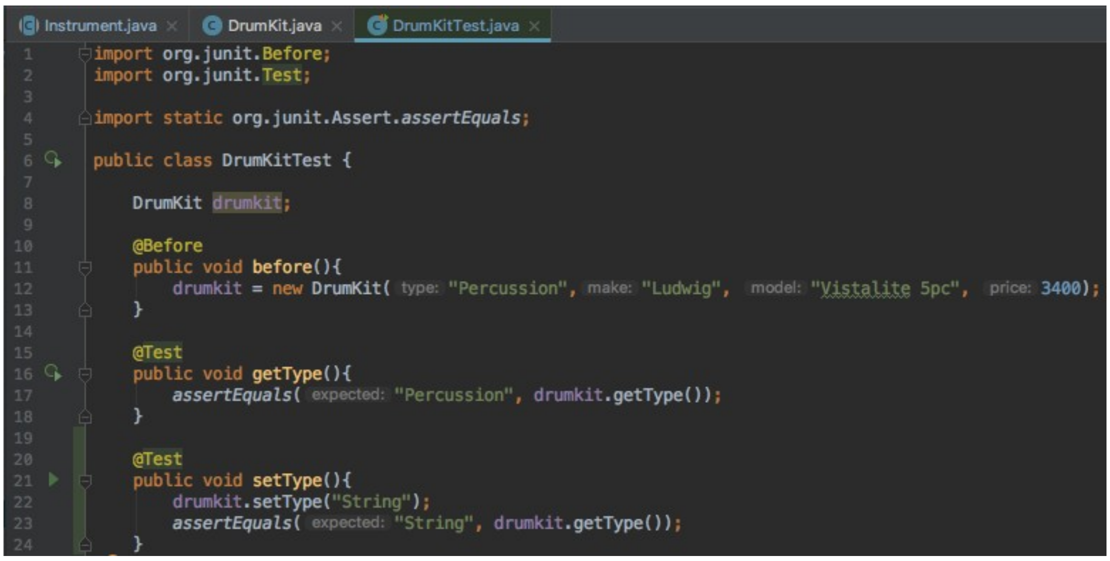
● A Class that inherits from the previous class



Instrument.java × DrumKit.java ×

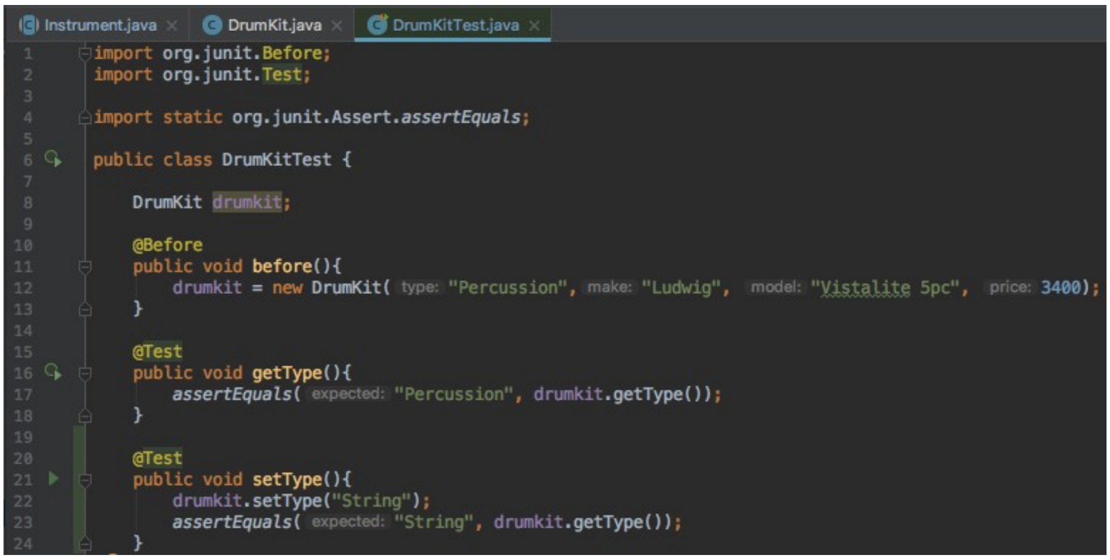
```java
import Interfaces.IPlay;

public class DrumKit extends Instrument implements IPlay {

    private String make;
    private String model;
    private int price;

    public DrumKit(String type, String make, String model, int price){
        super(type, make, model, price);
        }


        public String play(String sound){
        return "This instrument goes " + sound;
        }

}
```

● An Object in the inherited class



```java
import org.junit.Before;
import org.junit.Test;

import static org.junit.Assert.assertEquals;

public class DrumKitTest {

    DrumKit drumkit;

    @Before
    public void before(){
        drumkit = new DrumKit( type: "Percussion", make: "Ludwig",  model: "Vistalite 5pc",  price: 3400);
    }

    @Test
    public void getType(){
        assertEquals( expected: "Percussion", drumkit.getType());
    }

    @Test
    public void setType(){
        drumkit.setType("String");
        assertEquals( expected: "String", drumkit.getType());
    }
```

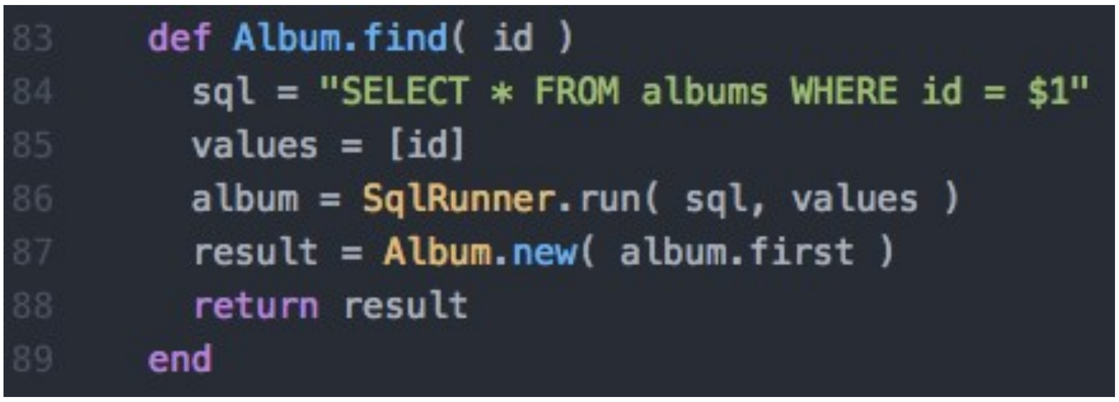
● A Method that uses the information inherited from another class.



```java
import org.junit.Before;
import org.junit.Test;

import static org.junit.Assert.assertEquals;

public class DrumKitTest {

    DrumKit drumkit;

    @Before
    public void before(){
        drumkit = new DrumKit( type: "Percussion", make: "Ludwig",  model: "Vistalite 5pc",  price: 3400);
    }

    @Test
    public void getType(){
        assertEquals( expected: "Percussion", drumkit.getType());
    }

    @Test
    public void setType(){
        drumkit.setType("String");
        assertEquals( expected: "String", drumkit.getType());
    }
```

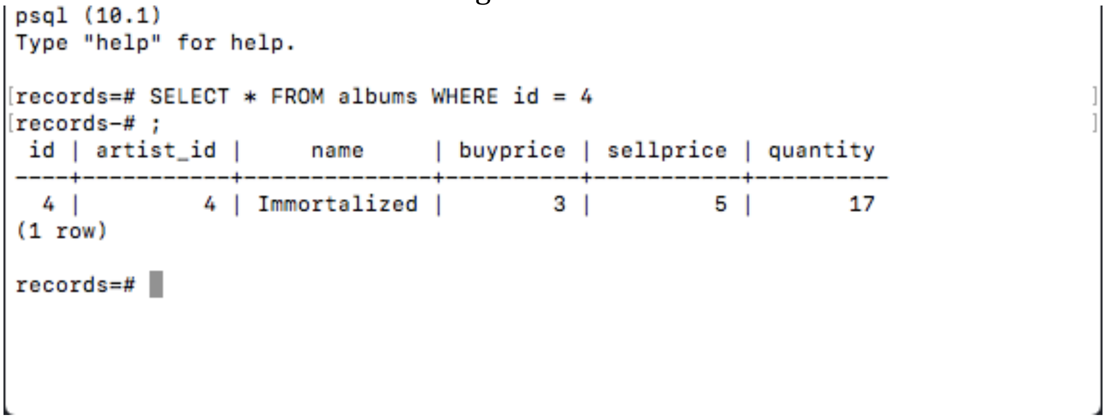
I.T 3 Demonstrate searching data in a program.
Take screenshots of:

● Function that searches data



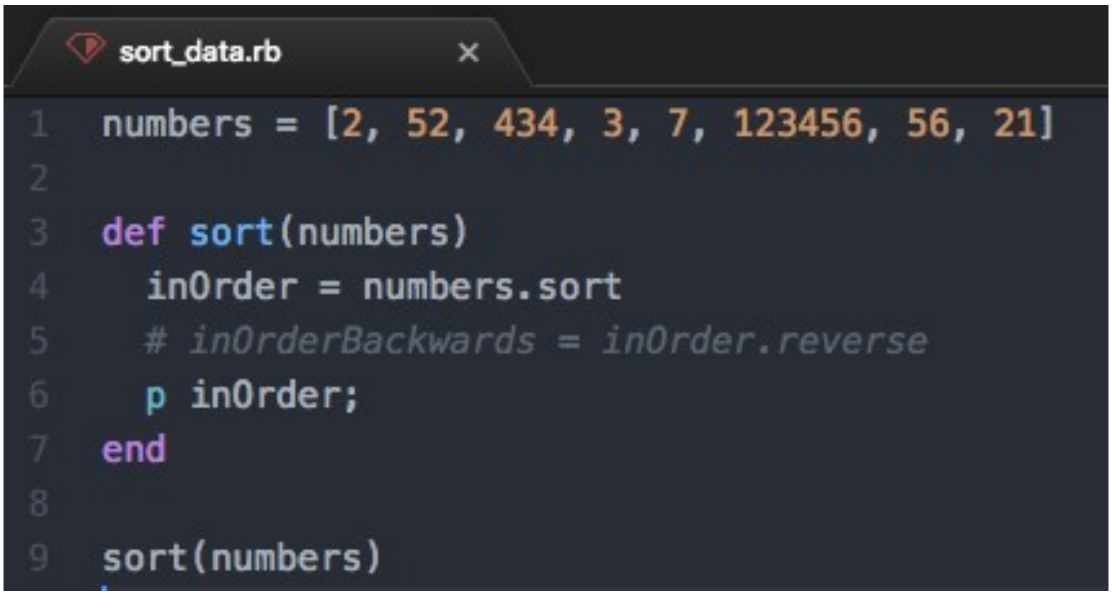
```
    def Album.find( id )
      sql = "SELECT * FROM albums WHERE id = $1"
      values = [id]
      album = SqlRunner.run( sql, values )
      result = Album.new( album.first )
      return result
    end
```

● The result of the function running



```
psql (10.1)
Type "help" for help.

records=# SELECT * FROM albums WHERE id = 4
records-# ;
 id | artist_id |     name      | buyprice | sellprice | quantity
----+-----------+---------------+----------+-----------+----------
  4 |         4 | Immortalized  |        3 |         5 |       17
(1 row)

records=# 
```

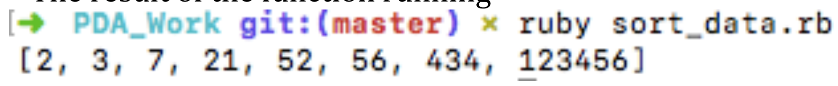
I.T 4 Demonstrate sorting data in a program.
Take screenshots of:
● Function that sorts data



sort_data.rb

```ruby
numbers = [2, 52, 434, 3, 7, 123456, 56, 21]

def sort(numbers)
  inOrder = numbers.sort
  # inOrderBackwards = inOrder.reverse
  p inOrder;
end

sort(numbers)
```

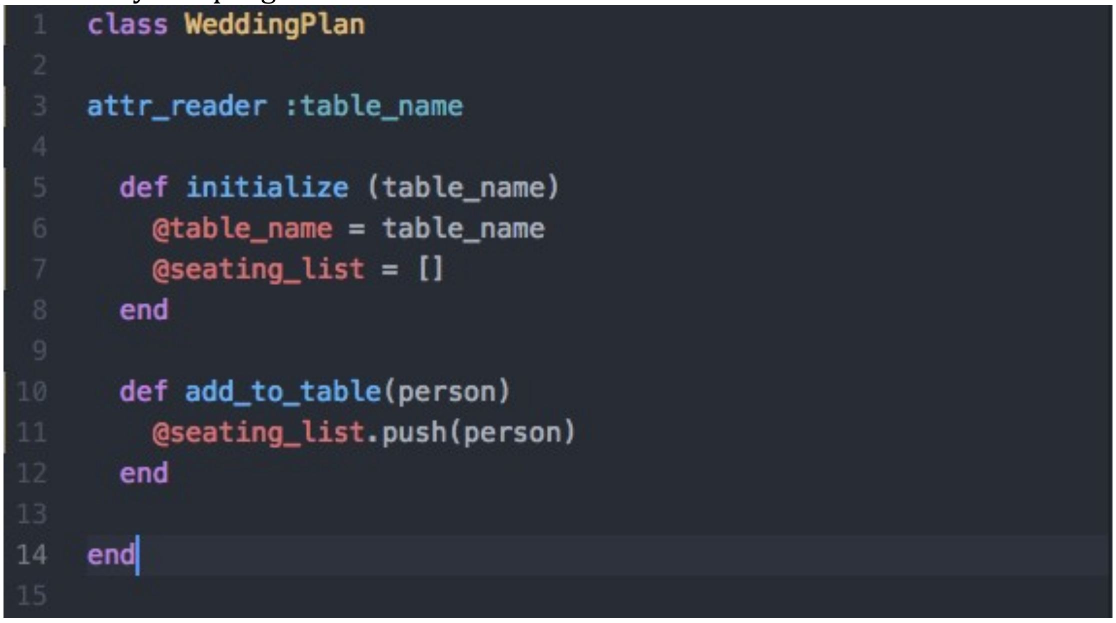
● The result of the function running



```
[→ PDA_Work git:(master) ✗ ruby sort_data.rb
[2, 3, 7, 21, 52, 56, 434, 123456]
```

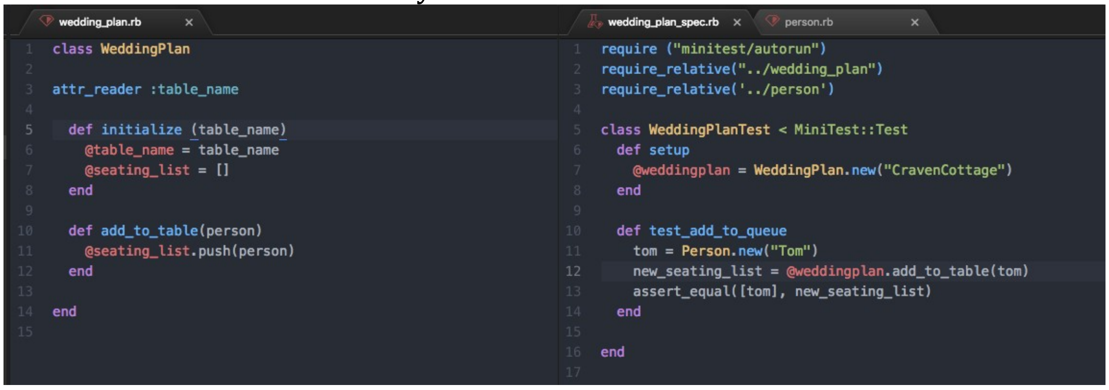
I.T 5 Demonstrate the use of an array in a program.
Take screenshots of:

● An array in a program



```ruby
class WeddingPlan

attr_reader :table_name

  def initialize (table_name)
    @table_name = table_name
    @seating_list = []
  end

  def add_to_table(person)
    @seating_list.push(person)
  end

end
```

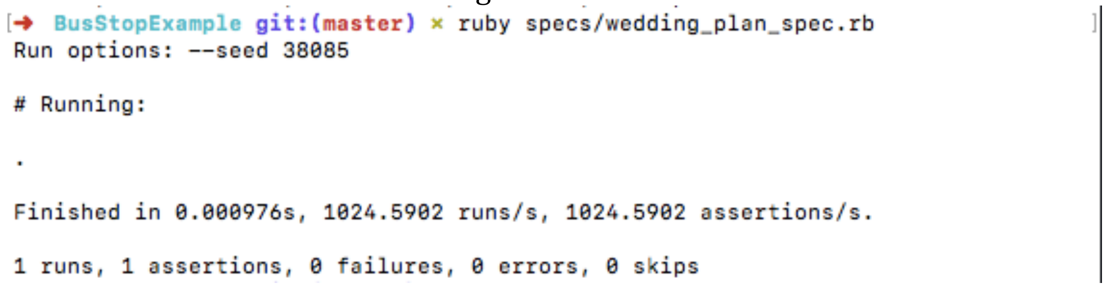
● A function that uses the array



```ruby
class WeddingPlan

attr_reader :table_name

  def initialize (table_name)
    @table_name = table_name
    @seating_list = []
  end

  def add_to_table(person)
    @seating_list.push(person)
  end

end
```

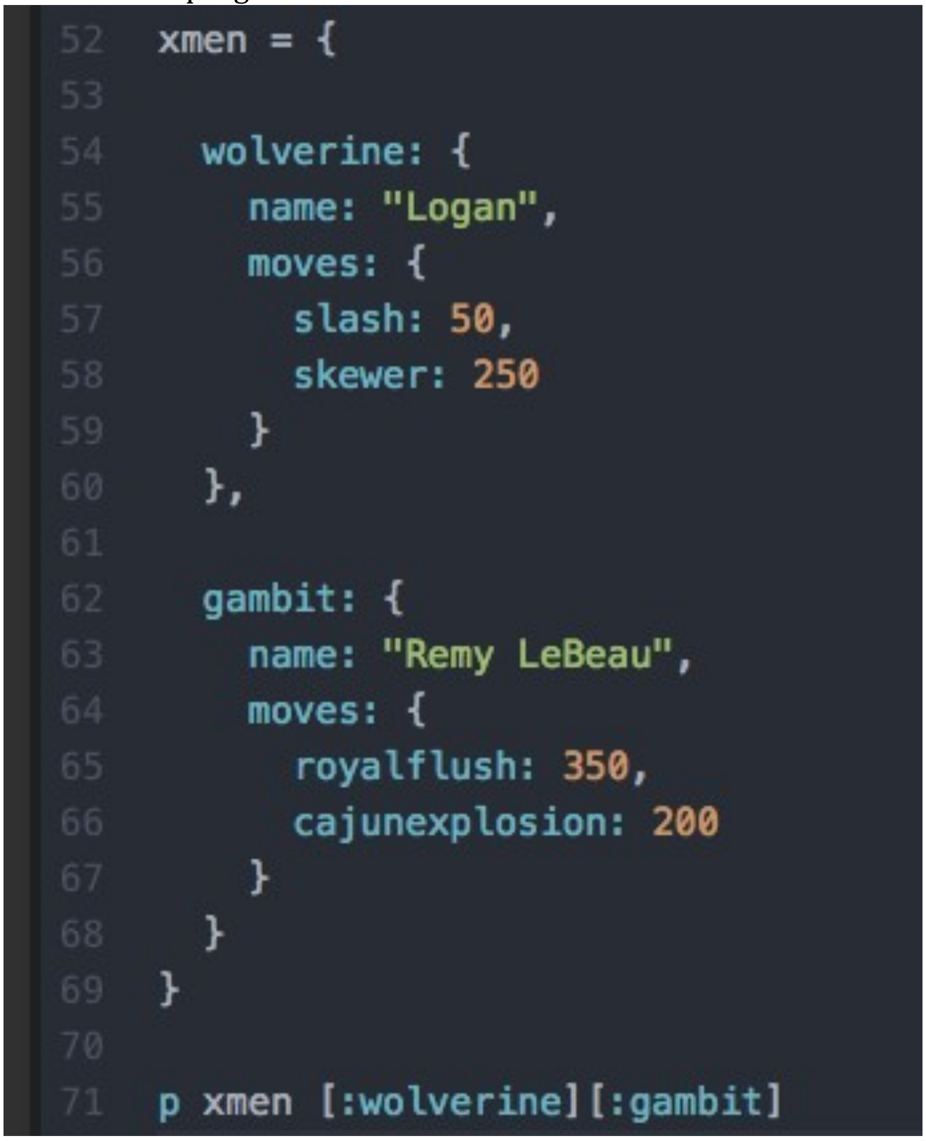
```ruby
require ("minitest/autorun")
require_relative("../wedding_plan")
require_relative('../person')

class WeddingPlanTest < MiniTest::Test
  def setup
    @weddingplan = WeddingPlan.new("CravenCottage")
  end

  def test_add_to_queue
    tom = Person.new("Tom")
    new_seating_list = @weddingplan.add_to_table(tom)
    assert_equal([tom], new_seating_list)
  end

end
```

● The result of the function running



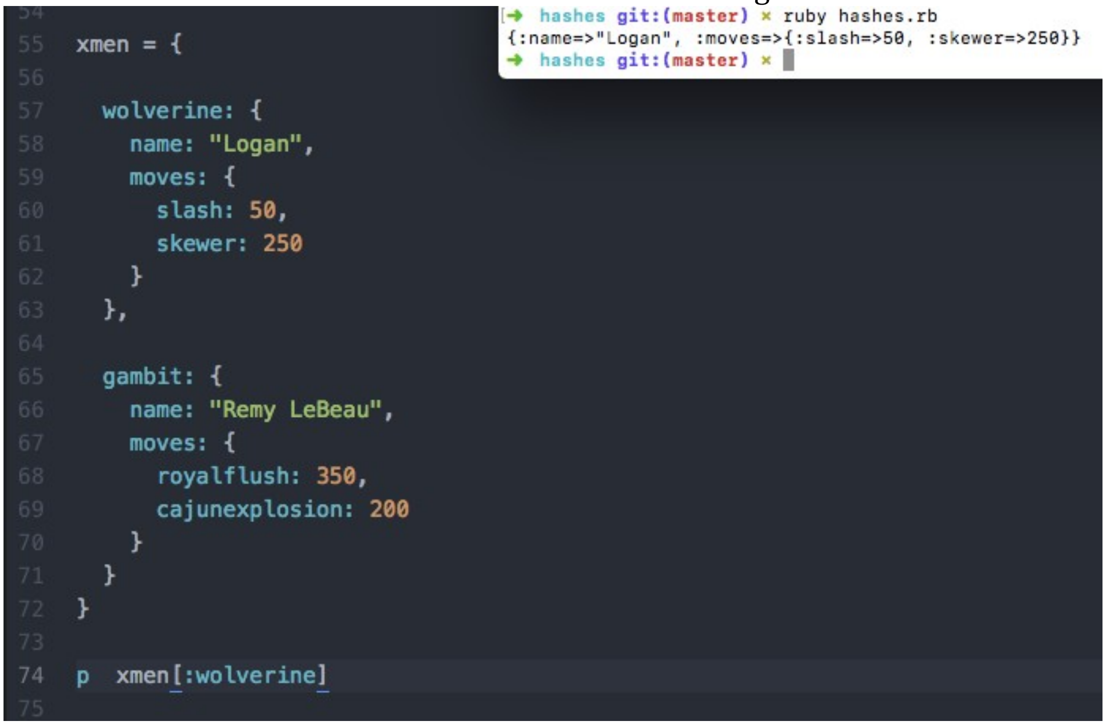
```
[➜  BusStopExample git:(master) ✗ ruby specs/wedding_plan_spec.rb
Run options: --seed 38085

# Running:

.

Finished in 0.000976s, 1024.5902 runs/s, 1024.5902 assertions/s.

1 runs, 1 assertions, 0 failures, 0 errors, 0 skips
```

I.T 6 Demonstrate the use of a hash in a program.
Take screenshots of:
● A hash in a program



```
xmen = {

  wolverine: {
    name: "Logan",
    moves: {
      slash: 50,
      skewer: 250
    }
  },

  gambit: {
    name: "Remy LeBeau",
    moves: {
      royalflush: 350,
      cajunexplosion: 200
    }
  }
}

p xmen [:wolverine][:gambit]
```

● A function that uses the hash and the function running

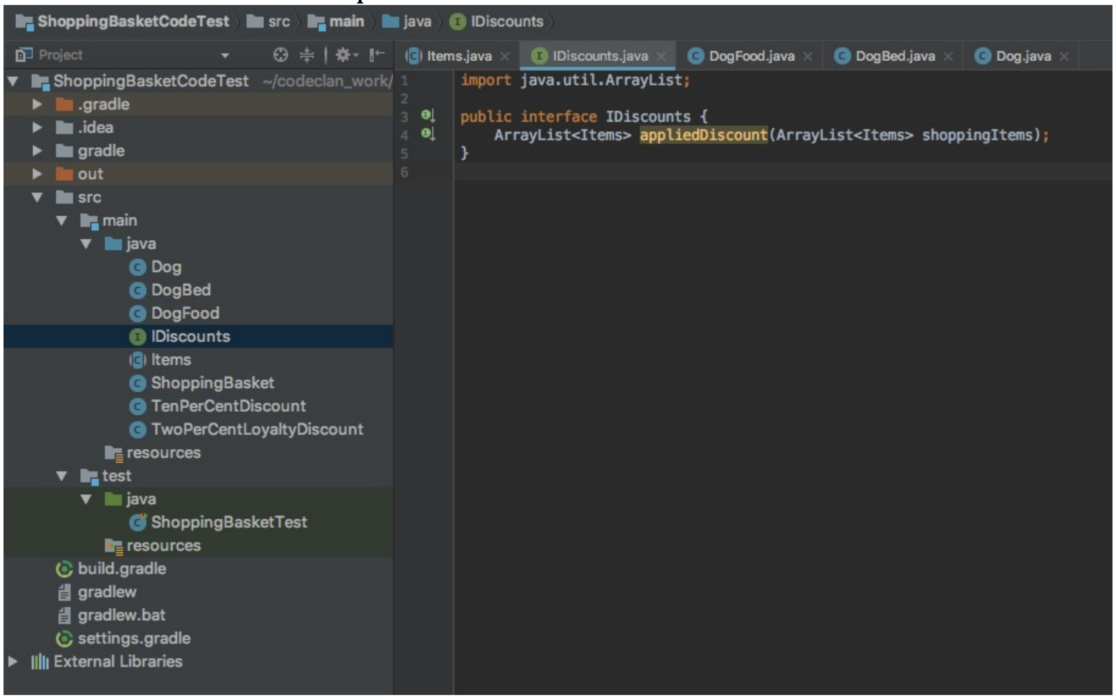


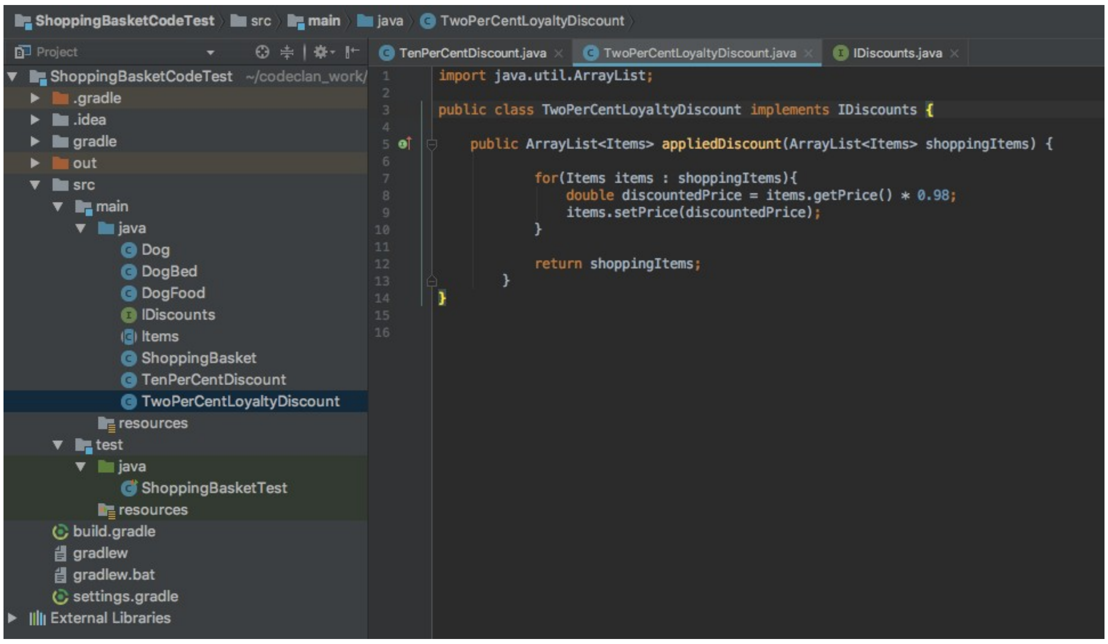
```
➜ hashes git:(master) ✗ ruby hashes.rb
{:name=>"Logan", :moves=>{:slash=>50, :skewer=>250}}
➜ hashes git:(master) ✗
```

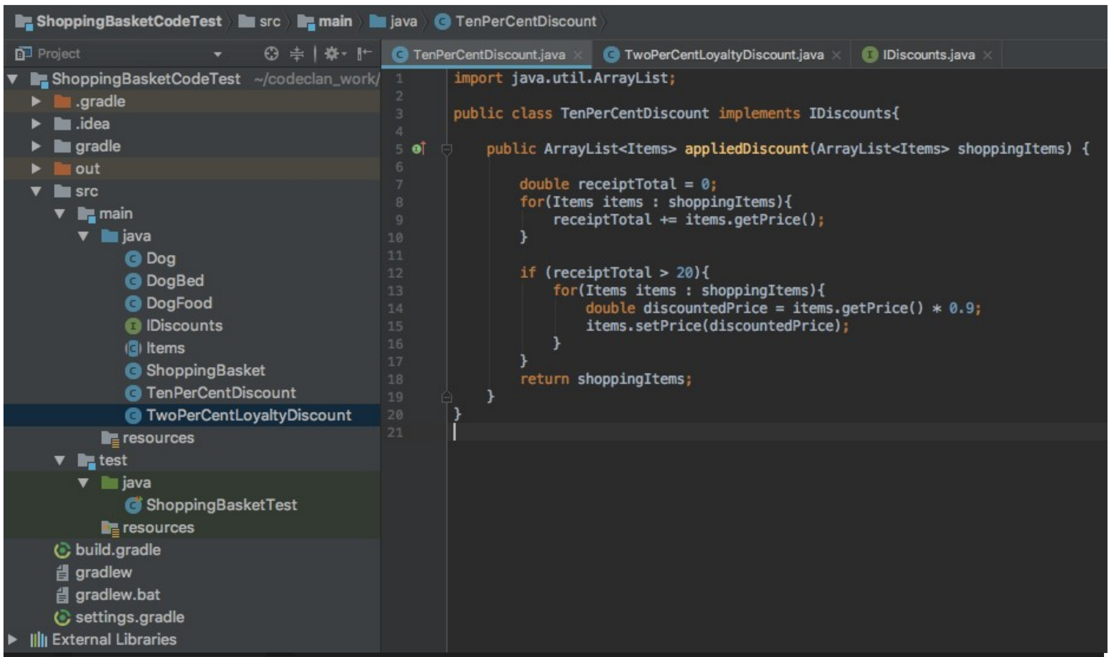
```ruby
xmen = {

  wolverine: {
    name: "Logan",
    moves: {
      slash: 50,
      skewer: 250
    }
  },

  gambit: {
    name: "Remy LeBeau",
    moves: {
      royalflush: 350,
      cajunexplosion: 200
    }
  }
}

p  xmen[:wolverine]
```

I.T 7 Demonstrate the use of Polymorphism in a program.

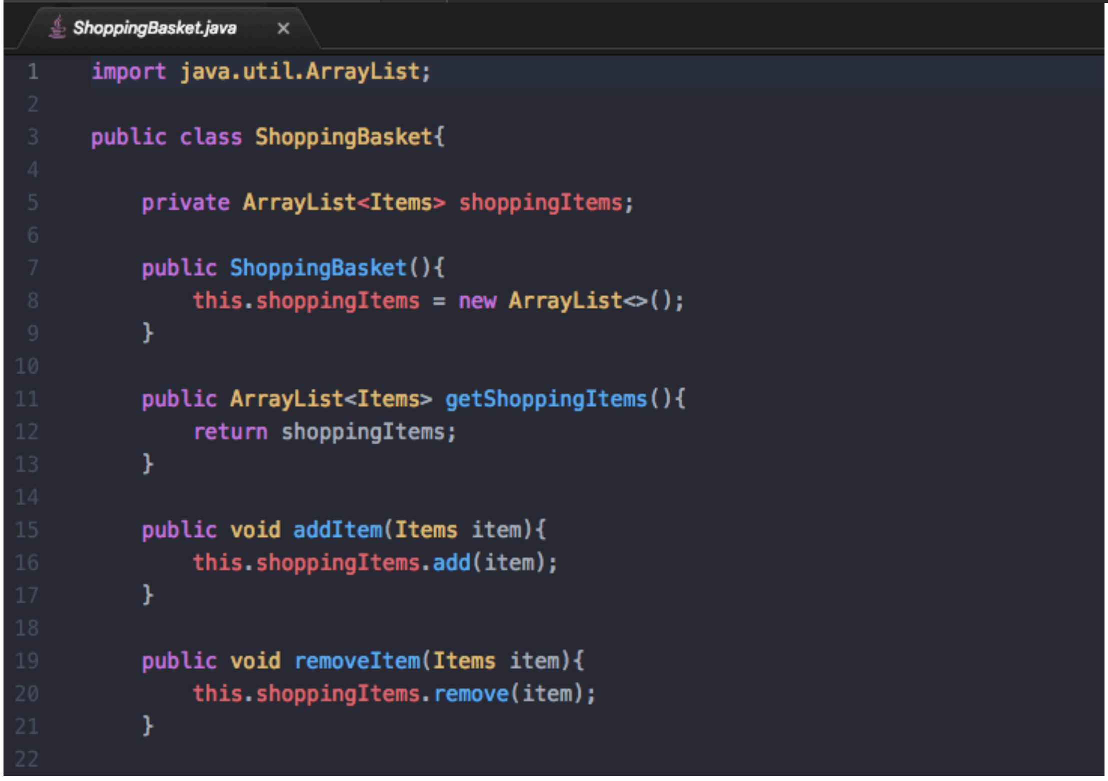
An interface used in multiple instances


```java
import java.util.ArrayList;

public interface IDiscounts {
    ArrayList<Items> appliedDiscount(ArrayList<Items> shoppingItems);
}
```

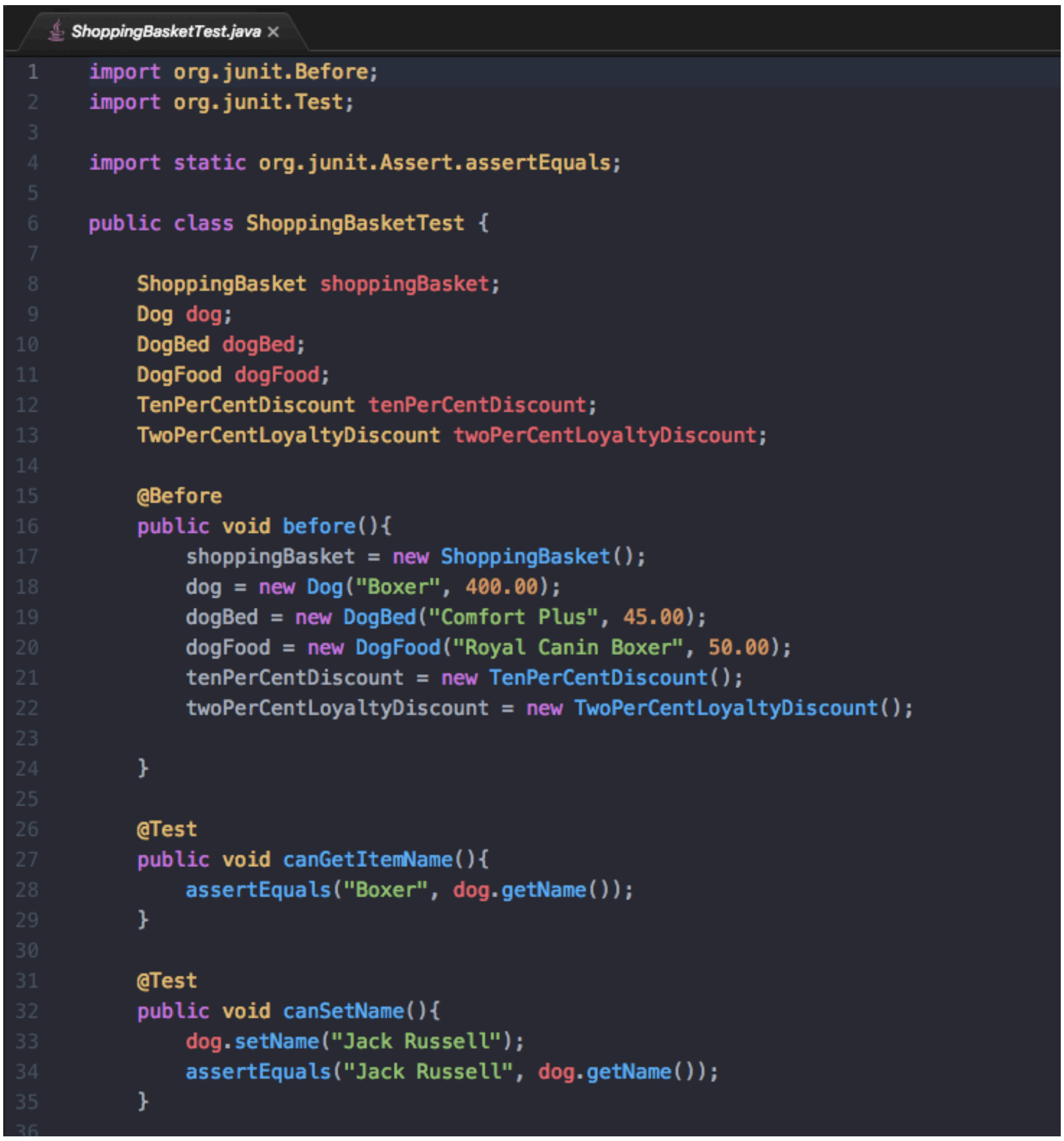

```java
import java.util.ArrayList;

public class TwoPerCentLoyaltyDiscount implements IDiscounts {

    public ArrayList<Items> appliedDiscount(ArrayList<Items> shoppingItems) {

            for(Items items : shoppingItems){
                double discountedPrice = items.getPrice() * 0.98;
                items.setPrice(discountedPrice);
            }

            return shoppingItems;
        }
}
```



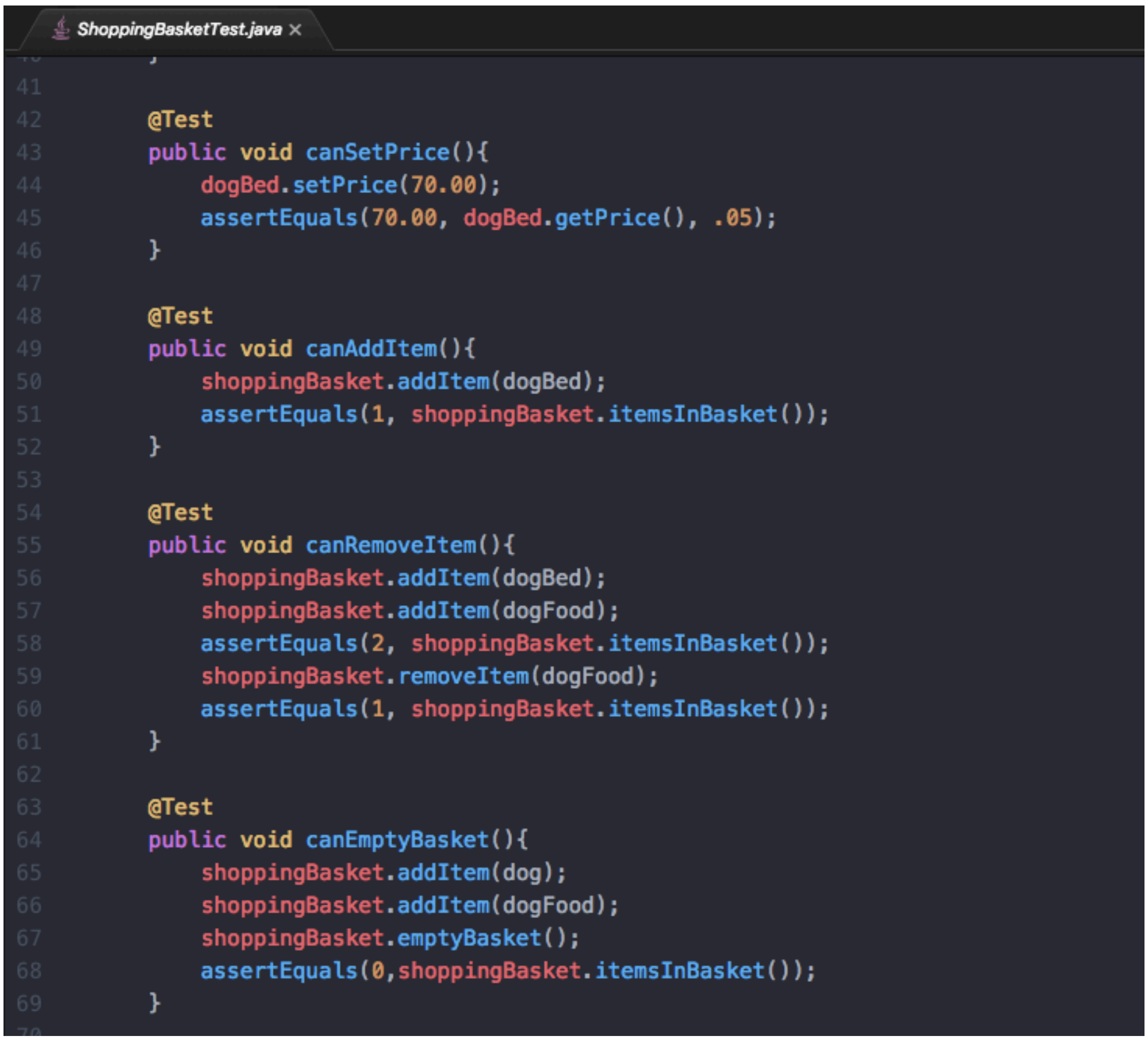
```java
import java.util.ArrayList;

public class TenPerCentDiscount implements IDiscounts{

    public ArrayList<Items> appliedDiscount(ArrayList<Items> shoppingItems) {

        double receiptTotal = 0;
        for(Items items : shoppingItems){
            receiptTotal += items.getPrice();
        }

        if (receiptTotal > 20){
            for(Items items : shoppingItems){
                double discountedPrice = items.getPrice() * 0.9;
                items.setPrice(discountedPrice);
            }
        }
        return shoppingItems;
    }
}
```



```java
import java.util.ArrayList;

public class ShoppingBasket{

    private ArrayList<Items> shoppingItems;

    public ShoppingBasket(){
        this.shoppingItems = new ArrayList<>();
    }

    public ArrayList<Items> getShoppingItems(){
        return shoppingItems;
    }

    public void addItem(Items item){
        this.shoppingItems.add(item);
    }

    public void removeItem(Items item){
        this.shoppingItems.remove(item);
    }
```



```java
import org.junit.Before;
import org.junit.Test;

import static org.junit.Assert.assertEquals;

public class ShoppingBasketTest {

    ShoppingBasket shoppingBasket;
    Dog dog;
    DogBed dogBed;
    DogFood dogFood;
    TenPerCentDiscount tenPerCentDiscount;
    TwoPerCentLoyaltyDiscount twoPerCentLoyaltyDiscount;

    @Before
    public void before(){
        shoppingBasket = new ShoppingBasket();
        dog = new Dog("Boxer", 400.00);
        dogBed = new DogBed("Comfort Plus", 45.00);
        dogFood = new DogFood("Royal Canin Boxer", 50.00);
        tenPerCentDiscount = new TenPerCentDiscount();
        twoPerCentLoyaltyDiscount = new TwoPerCentLoyaltyDiscount();

    }

    @Test
    public void canGetItemName(){
        assertEquals("Boxer", dog.getName());
    }

    @Test
    public void canSetName(){
        dog.setName("Jack Russell");
        assertEquals("Jack Russell", dog.getName());
    }
```



ShoppingBasketTest.java

```
    @Test
    public void canSetPrice(){
        dogBed.setPrice(70.00);
        assertEquals(70.00, dogBed.getPrice(), .05);
    }

    @Test
    public void canAddItem(){
        shoppingBasket.addItem(dogBed);
        assertEquals(1, shoppingBasket.itemsInBasket());
    }

    @Test
    public void canRemoveItem(){
        shoppingBasket.addItem(dogBed);
        shoppingBasket.addItem(dogFood);
        assertEquals(2, shoppingBasket.itemsInBasket());
        shoppingBasket.removeItem(dogFood);
        assertEquals(1, shoppingBasket.itemsInBasket());
    }

    @Test
    public void canEmptyBasket(){
        shoppingBasket.addItem(dog);
        shoppingBasket.addItem(dogFood);
        shoppingBasket.emptyBasket();
        assertEquals(0,shoppingBasket.itemsInBasket());
    }
```