Evidence for Implementation and Testing Unit.

Luis Farid Tejero Aoun

E - 21

I.T 1- Demonstrate one example of encapsulation that you have written in a program.

Evidence for unit

I.T 2 - Example the use of inheritance in a program.

Evidence for unit

I.T 3 - Example of searching

```
66
67  def self.film_by_id(id)
68    sql = "SELECT films.* FROM films WHERE films.id = $1"
69    values = [id]
70    films_hashes = SqlRunner.run(sql, values)
71    film = films_hashes.map {|film| Film.new(film)}
72    return film
73  end
```

Result of the search:

```
[➜ weeknd_hw git:(master) ✗ ruby db/console.rb

From: /Users/user/codeclan_work/week_03/day_5/weeknd_hw/db/c
onsole.rb @ line 36 :

    31: ticket3.save()
    32:
    33:
    34:
    35: binding.pry
 => 36: nil

[1] pry(main)> Ticket.film_by_id(2)                          ]
=> [#<Film:0x007fafa2cc20c8 @id=2, @price=15, @title="Drive"
>]
[2] pry(main)>
```

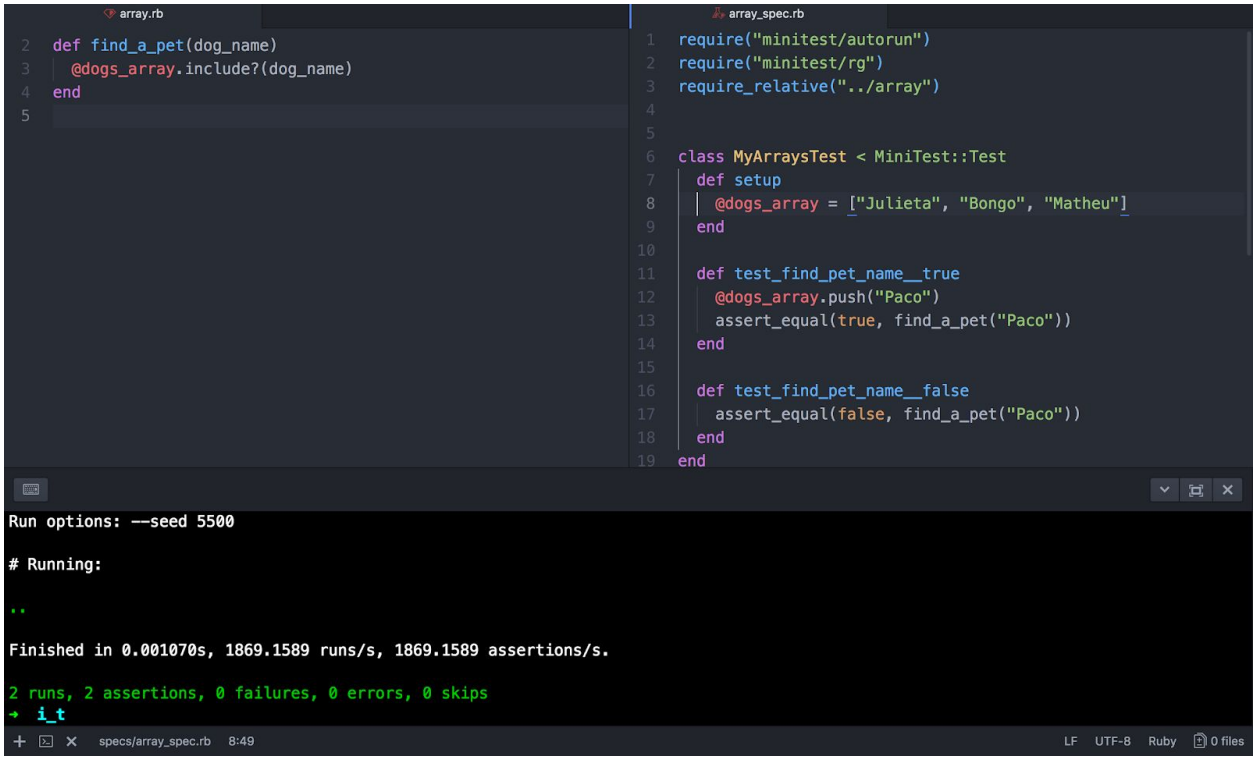
I.T 4 – Example of sorting

```
def self.most_sold()
sql = "SELECT screening_id FROM tickets GROUP BY screening_id ORDER BY COUNT(*) DESC LIMIT
1"
tickets = SqlRunner.run(sql)[0]['screening_id'].to_i
return self.film_by_id(tickets)[0].title + " is the most popular film at: " +
self.screening_by_id(tickets)[0].function_time.to_s
end
```

Result of sorting:

```
[2] pry(main)> Ticket.most_sold()
=> "Infinity War is the most popular film at: 10"
[3] pry(main)>
```

I.T 5 - Example of an array, a function that uses an array and the result



I.T 6 - Example of a hash, a function that uses a hash and the result

hashes.rb

```ruby

def pets_by_type(shopname, type)
  count = []
  for pets in shopname[:pets]
    if (pets[:pet_type] == type)
      count.push(pets)
    end
  end
  return count
end

```

hashes_spec.rb

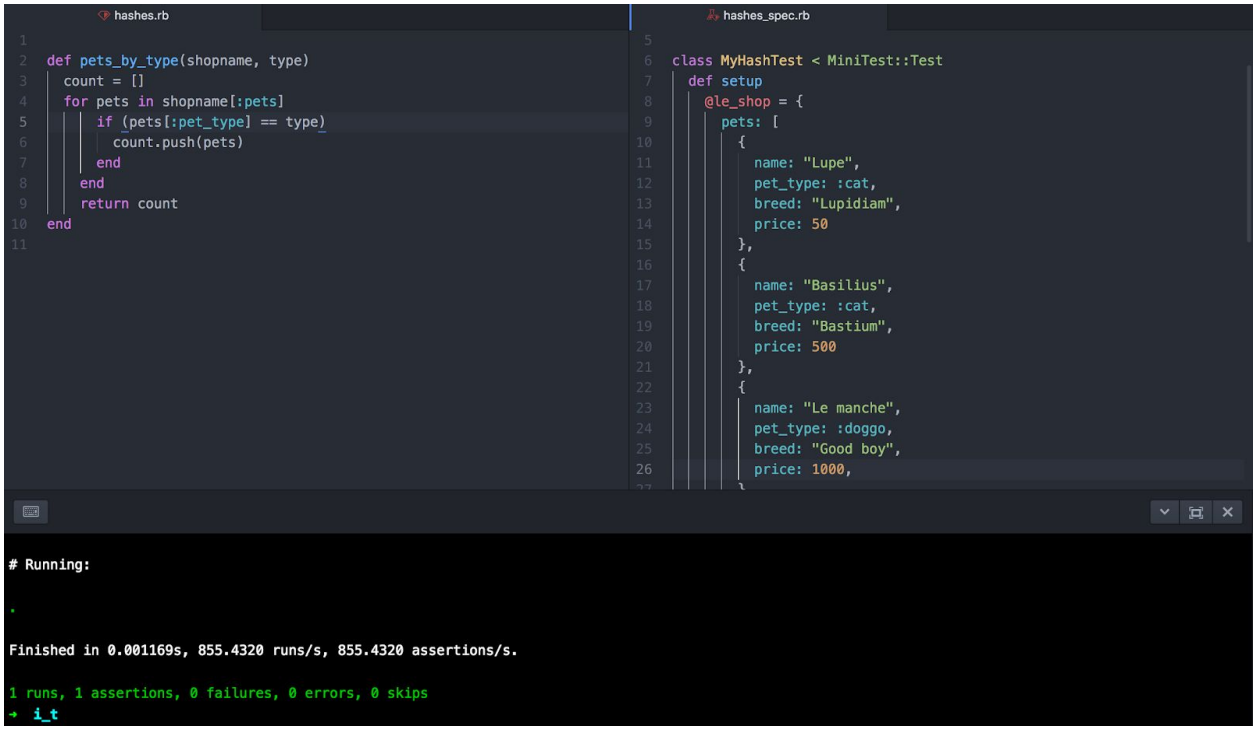
```ruby

class MyHashTest < MiniTest::Test
  def setup
    @le_shop = {
      pets: [
        {
          name: "Lupe",
          pet_type: :cat,
          breed: "Lupidiam",
          price: 50
        },
        {
          name: "Basilius",
          pet_type: :cat,
          breed: "Bastium",
          price: 500
        },
        {
          name: "Le manche",
          pet_type: :doggo,
          breed: "Good boy",
          price: 1000,
        }
      ]
    }
  end

  def test_all_pets_by_type
    pets = pets_by_type(@le_shop, :doggo)
    assert_equal(1, pets.count)
  end
end

```

RESULT OF TEST:



I.T 7 - Example of polymorphism in a program

```java
package musicShop.stock;

public abstract class Instrument implements IPlay, ISell{
    private String colour, brand, type;
    private int buyingPrice, sellingPrice;

    public Instrument(String type, String colour, String brand, int buyingPrice, int sellingPrice){
        this.type = type;
        this.colour = colour;
        this.brand = brand;
        this.buyingPrice = buyingPrice;
        this.sellingPrice = sellingPrice;
    }

    public String getType() {
        return this.type;
    }

    public String getColour() {
        return this.colour;
    }

    public String getBrand() {
        return this.brand;
    }

    public int getBuyingPrice() {
        return this.buyingPrice;
    }

    @Override
    public int getSellingPrice() {
        return this.sellingPrice;
    }

    @Override
    public String play() {
        return "You are playing the " + this.getType();
    }

    @Override
    public int calculateMarkup() {
        return (this.sellingPrice - this.buyingPrice);
    }
}
```

```java
package musicShop.stock;

public interface ISell {
    public int calculateMarkup();
    public int getSellingPrice();
}
```

```java
package musicShop.stock;

public interface IPlay {
    public String play();
}
```

```java
package musicShop.stock;

public abstract class Accessory implements ISell {
    private String type;
    private int buyingPrice, sellingPrice;

    public Accessory(String type, int buyingPrice, int sellingPrice){
        this.type = type;
        this.buyingPrice = buyingPrice;
        this.sellingPrice = sellingPrice;
    }

    public String getType() {
        return this.type;
    }

    public int getBuyingPrice() {
        return this.buyingPrice;
    }

    @Override
    public int getSellingPrice() {
        return this.sellingPrice;
    }
    @Override
    public int calculateMarkup() {
        return (this.sellingPrice - this.buyingPrice);
    }
}
```

```java
import java.util.ArrayList;

public class Shop {
    private ArrayList<IPlay> exhibition;
    private ArrayList<ISell> stock;
    private int till;

    public Shop(){
        this.exhibition = new ArrayList<>();
        this.stock = new ArrayList<>();
        this.till = 0;
    }

    public ArrayList<IPlay> getExhibition() {
        return this.exhibition;
    }

    public int getTill(){
        return  this.till;
    }

    public ArrayList<ISell> getStock() {
        return this.stock;
    }
    public void addStock(ISell iSell){
        this.stock.add(iSell);
    }
    public void removeStock(ISell iSell){
        this.stock.remove(iSell);
    }
    public void addExhibition(IPlay instrument){
        this.exhibition.add(instrument);
    }

    public void addMoney(int amount){
        this.till += amount;
    }

    public void sellItem(ISell iSell, Customer customer){
        if (customer.getWallet() >= iSell.getSellingPrice()){
            this.addMoney(iSell.getSellingPrice());
            customer.removeMoneyFromWallet(iSell.getSellingPrice());
            customer.addItemTobag(iSell);
            this.removeStock(iSell);
        }
    }
```