

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.

```
public abstract class Character {
    private String name;
    private double maxHp, hp, attPower;

    public Character(String name, double maxHp, double hp, double attPower ){
        this.name = name;
        this.maxHp = maxHp;
        this.hp = hp;
        this.attPower = attPower;
    }

    public String getName() {
        return this.name;
    }

    public double getMaxHp() {
        return this.maxHp;
    }

    public double getHp() {
        return this.hp;
    }

    public double getAttPower() {
        return this.attPower;
    }

    public void setName(String name) {
        this.name = name;
    }

    public void setMaxHp(double maxHp) {
        this.maxHp = maxHp;
    }

    public void setHp(double hp) {
        this.hp = hp;
    }

    public void setAttPower(double attPower) {
        this.attPower = attPower;
    }
}
```

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

- Class

```
public abstract class Character {
    private String name;
    private double maxHp, hp, attPower;

    public Character(String name, double maxHp, double hp, double attPower ){
        this.name = name;
        this.maxHp = maxHp;
        this.hp = hp;
        this.attPower = attPower;
    }

    public String getName() {
        return this.name;
    }

    public double getMaxHp() {
        return this.maxHp;
    }

    public double getHp() {
        return this.hp;
    }

    public double getAttPower() {
        return this.attPower;
    }

    public void setName(String name) {
        this.name = name;
    }

    public void setMaxHp(double maxHp) {
        this.maxHp = maxHp;
    }

    public void setHp(double hp) {
        this.hp = hp;
    }

    public void setAttPower(double attPower) {
        this.attPower = attPower;
    }
}
```

- A class that inherits from previous class.

```
1 package com.luist17.combat;
2
3 public abstract class Warrior extends Character {
4     private double maxAdrenaline, adrenaline;
5
6     public Warrior(String name, double maxHp, double hp, double attPower, double maxAdrenaline, double adrenaline ){
7         super(name, maxHp, hp, attPower);
8         this.maxAdrenaline = maxAdrenaline;
9         this.adrenaline = adrenaline;
10    }
11
12    public double getMaxAdrenaline() { return this.maxAdrenaline; }
13
14    public void setMaxAdrenaline(double maxAdrenaline) { this.maxAdrenaline = maxAdrenaline; }
15
16    public double getAdrenaline() { return this.adrenaline; }
17
18    public void setAdrenaline(double adrenaline) { this.adrenaline = adrenaline; }
19
20    public void basicAttack(Character character){...}
21    public void specialAttack(Character character){...}
22    public void actionBack(Character character){...}
23 }
24
25
```

- An Object from the class.

```
1 package com.luist17.combat;
2
3 import com.luist17.combat.Warrior;
4
5 public class Ninja extends Warrior {
6     public Ninja(String name) { super( name, maxHp: 200.0, hp: 200.0, attPower: 25.0, maxAdrenaline: 100.0, adrenaline: 0); }
7
8 }
9
10
11
```

- A Method that uses information inherited from other classes.

```
public class Arena {
    private ArrayList<Character> fighters;
    private static final int TIME_DELAY = 2000;

    public Arena(){
        this.fighters = new ArrayList<>();
    }

    public ArrayList<Character> getFighters() {
        return this.fighters;
    }

    public void addPlayer(Character character){
        this.fighters.add(character);
    }

    public String fightTillDead() throws InterruptedException{
        Collections.shuffle(fighters);
        System.out.println(String.format("%s is fighting %s: ", fighters.get(0).getName(), fighters.get(1).getName()));
        while (fighters.get(0).isAlive() && fighters.get(1).isAlive()){
            Thread.sleep(TIME_DELAY);
            fighters.get(0).actionBack(fighters.get(1));
            if (fighters.get(1).isAlive()){
                Thread.sleep(TIME_DELAY);
                fighters.get(1).actionBack(fighters.get(0));
            } else {
                return fighters.get(0).getName() + " Has won";
            }
        }
        return fighters.get(1).getName() + " Has won";
    }
}
```

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_nw git:(master) ✗ ruby db/console.rb

From: /Users/user/codeclan_work/week_03/day_5/weeknd_hw/db/console.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

```
array.rb
2 def find_a_pet(dog_name)
3   @dogs_array.include?(dog_name)
4 end
5

array_spec.rb
1 require("minitest/autorun")
2 require("minitest/rg")
3 require_relative("../array")
4
5
6 class MyArraysTest < MiniTest::Test
7   def setup
8     @dogs_array = ["Julieta", "Bongo", "Matheu"]
9   end
10
11   def test_find_pet_name_true
12     @dogs_array.push("Paco")
13     assert_equal(true, find_a_pet("Paco"))
14   end
15
16   def test_find_pet_name_false
17     assert_equal(false, find_a_pet("Paco"))
18   end
19 end

Run options: --seed 5500
# Running:
..
Finished in 0.001070s, 1869.1589 runs/s, 1869.1589 assertions/s.
2 runs, 2 assertions, 0 failures, 0 errors, 0 skips
+ i_t
```

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

```
hashes.rb
1
2 def pets_by_type(shopname, type)
3   count = []
4   for pets in shopname[:pets]
5     if (pets[:pet_type] == type)
6       count.push(pets)
7     end
8   end
9   return count
10 end
11

hashes_spec.rb
5
6 class MyHashTest < MiniTest::Test
7   def setup
8     @le_shop = {
9       pets: [
10        {
11          name: "Lupe",
12          pet_type: :cat,
13          breed: "Lupidiam",
14          price: 50
15        },
16        {
17          name: "Basilus",
18          pet_type: :cat,
19          breed: "Bastium",
20          price: 500
21        },
22        {
23          name: "Le manche",
24          pet_type: :doggo,
25          breed: "Good boy",
26          price: 1000,
27        }
28      ]
29    }
30  end
31
32  def test_all_pets_by_type
33    pets = pets_by_type(@le_shop, :doggo)
34    assert_equal(1, pets.count)
35  end
36 end
37
```

RESULT OF TEST:

```
hashes.rb
1
2 def pets_by_type(shopname, type)
3   count = []
4   for pets in shopname[:pets]
5     if (pets[:pet_type] == type)
6       count.push(pets)
7     end
8   end
9   return count
10 end
11

hashes_spec.rb
5
6 class MyHashTest < MiniTest::Test
7   def setup
8     @le_shop = {
9       pets: [
10        {
11          name: "Lupe",
12          pet_type: :cat,
13          breed: "Lupidiam",
14          price: 50
15        },
16        {
17          name: "Basilus",
18          pet_type: :cat,
19          breed: "Bastium",
20          price: 500
21        },
22        {
23          name: "Le manche",
24          pet_type: :doggo,
25          breed: "Good boy",
26          price: 1000,
27        }
28      ]
29    }
30  end
31
32  def test_all_pets_by_type
33    pets = pets_by_type(@le_shop, :doggo)
34    assert_equal(1, pets.count)
35  end
36 end
37

# Running:
+
Finished in 0.001169s, 855.4320 runs/s, 855.4320 assertions/s.

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


I.T 7 - Example of polymorphism in a program

```
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);
    }
}
```

```
package musicShop.stock;

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

```
package musicShop.stock;

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

```

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);
    }
}

```

```

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);
        }
    }
}

```