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.

```java
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**

```java
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.**

```java
package com.luist17.combat;

public abstract class Warrior extends Character {

    private double maxAdrenaline, adrenaline;

    public Warrior(String name, double maxHp, double hp, double attPower, double maxAdrenaline, double adrenaline ){
        super(name, maxHp, hp, attPower);
        this.maxAdrenaline = maxAdrenaline;
        this.adrenaline = adrenaline;
    }

    public double getMaxAdrenaline() { return this.maxAdrenaline; }

    public void setMaxAdrenaline(double maxAdrenaline) { this.maxAdrenaline = maxAdrenaline; }

    public double getAdrenaline() { return this.adrenaline; }

    public void setAdrenaline(double adrenaline) { this.adrenaline = adrenaline; }


    public void basicAttack(Character character){...}

    public void specialAttack(Character character){...}

    public void actionBack(Character character){...}
}
```

- An Object from the class.

```java
package com.luist17.combat;

import com.luist17.combat.Warrior;

public class Ninja extends Warrior {
    public Ninja(String name) { super( name, maxHp: 200.0,  hp: 200.0,  attPower: 25.0,  maxAdrenaline: 100.0,  adrenaline: 0); }

}
```

- **A Method that uses information inherited from other classes.**

```java
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";

    }
}
```

- **Testing Knight class.**

```java
public class KnightTest {
    Knight knight;
    Ninja ninja2;

    @Before
    public void before(){
        knight = new Knight( name: "Player 1");
        ninja2 = new Ninja( name: "Android");
    }

    @Test
    public void hasName() { assertEquals( expected: "Player 1", this.knight.getName()); }

    @Test
    public void hasMaxHp() { assertEquals( expected: 200.0, this.knight.getMaxHp(),  delta: 0.01); }

    @Test
    public void hasHp() { assertEquals( expected: 200, this.knight.getHp(),  delta: 0.01); }

    @Test
    public void hasAttPower() { assertEquals( expected: 25, this.knight.getAttPower(),  delta: 0.01); }

    @Test
    public void hasAdrenaline() { assertEquals( expected: 0, this.knight.getAdrenaline(),  delta: 0.01); }
```

```
All 17 tests passed - 9ms

KnightTest (com.example 9ms
    hasAdrenaline 1ms
    canCheckIfIsAlive 0ms
    hasAttPower 0ms
    hasAttackLogic 6ms
    canCheckIfIsAlive_Dea 0ms
    canChangeAdrenaline 0ms
    hasHp 0ms
    canChangeMaxHp 0ms
    hasMaxHp 0ms
    canChangeMaxAdrena 0ms
    canChangeAttPower 0ms
    canChangeName 0ms

"/Applications/Android Studio.app/Contents/jre/jdk/Contents/Home/bin/java" ...
Player 1 has received 29.00 damage! -  Player 1 has 171.00 hp left!!
Android has received 50.00 damage! -  Android has 150.00 hp left!!
Player 1 used a Special attack!
Android has received 35.00 damage! -  Android has 165.00 hp left!!
Android has received 50.00 damage! -  Android has 150.00 hp left!!
Player 1 used a Special attack!

Process finished with exit code 0
```

I.T 3 - Example of searching

```ruby
def self.film_by_id(id)
  sql = "SELECT films.* FROM films WHERE films.id = $1"
  values = [id]
  films_hashes = SqlRunner.run(sql, values)
  film = films_hashes.map {|film| Film.new(film)}
  return film
end
```

Result of the search:

```
weekna_nw 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

array.rb

```ruby
def find_a_pet(dog_name)
  @dogs_array.include?(dog_name)
end
```

array_spec.rb

```ruby
require("minitest/autorun")
require("minitest/rg")
require_relative("../array")


class MyArraysTest < MiniTest::Test
  def setup
    @dogs_array = ["Julieta", "Bongo", "Matheu"]
  end

  def test_find_pet_name__true
    @dogs_array.push("Paco")
    assert_equal(true, find_a_pet("Paco"))
  end

  def test_find_pet_name__false
    assert_equal(false, find_a_pet("Paco"))
  end
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
```

specs/array_spec.rb 8:49 LF UTF-8 Ruby 0 files

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:

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,
```

```
# 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

```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);
        }
```