Evidence for Implementation and Testing Unit.

Juan Mata Ruiz
E21

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

Paddock.java

```java
package Areas;

import Dinosaur.Specie;
import Enum.DinosaurType;
import Enum.PaddockType;

import java.util.ArrayList;

public class Paddock {

    private String name;
    PaddockType paddockType;
    private ArrayList<Specie> species;

    public Paddock(String name, PaddockType paddockType) {
        this.name = name;
        this.paddockType = paddockType;
        this.species = new ArrayList<>();
    }

    public String getName() { return name; }

    public PaddockType getType() { return paddockType; }

    public int getCount() { return species.size(); }

    public void addDinosaur(Specie specie) { this.species.add(specie); }

    public void removeDinosaur() { this.species.remove( index: 0); }

    public boolean herbivoreOnly(){
        for (Specie specie : this.species) {
            if (specie.getType() == DinosaurType.HERBIVORE) {
            }
        }
        return true;
    }

}
```

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

TyrannosaurusRex.java

```java
package Dinosaur;

import Areas.Paddock;
import Behaviour.IHunt;
import Enum.DinosaurType;
import ThemePark.Park;

public class TyrannosaurusRex extends Specie implements IHunt {

    public TyrannosaurusRex(int stomach, int rampage, DinosaurType dinosaurType) {
        super(stomach, rampage, dinosaurType);
    }

    public void hunt(Paddock paddock, Park park) {
        if (getRampage() >= this.rampage ) {
            paddock.removeDinosaur();
            park.removeVisitor();
            this.dinosaurIsFed();
        }
    }


}
```

```java
public abstract class Specie {

    private int stomach;
    protected int rampage;
    private DinosaurType dinosaurType;

    public Specie(int stomach, int rampage, DinosaurType dinosaurType) {
        this.stomach = stomach;
        this.rampage = rampage;
        this.dinosaurType = dinosaurType;
    }

    public DinosaurType getType() {
        return dinosaurType;
    }

    public int getStomach() {
        return stomach;
    }

    public void dinosaurIsFed() {
        this.stomach++;
    }

    public int getRampage() {
      Random rand = new Random();
        return rand.nextInt( bound: 10) + 1;
    }

}
```

```java
public class TyrannosaurusRexTest {

    TyrannosaurusRex tyrannosaurusRex1, tyrannosaurusRex2;
    Park park;
    Paddock paddock;
    Visitor visitor1, visitor2;

    @Before
    public void setup() {
        park = new Park();

        tyrannosaurusRex1 = new TyrannosaurusRex( stomach: 4,  rampage: 5, DinosaurType.CARNIVORE);
        tyrannosaurusRex2 = new TyrannosaurusRex( stomach: 6,  rampage: 5, DinosaurType.CARNIVORE);

        paddock = new Paddock( name: "T-Rex Paddock", PaddockType.PREDATORS);

        visitor1 = new Visitor( name: "Alan Grant", speed: 6);
        visitor2 = new Visitor( name: "Ellie Sattler",  speed: 8);
    }

    @Test
    public void dinosaurHasType() { assertEquals(DinosaurType.CARNIVORE, tyrannosaurusRex1.getType()); }

    @Test
    public void dinosaurHasStomach() { assertEquals( expected: 4, tyrannosaurusRex1.getStomach()); }

    @Test
    public void dinosaurCanBeFed() {...}

    @Test
    public void dinosaurCanRampage() {...}

    @Test
    public void dinosaurCanHunt(){...}

    @Test
    public void dinosaurCannotHunt() {...}
```

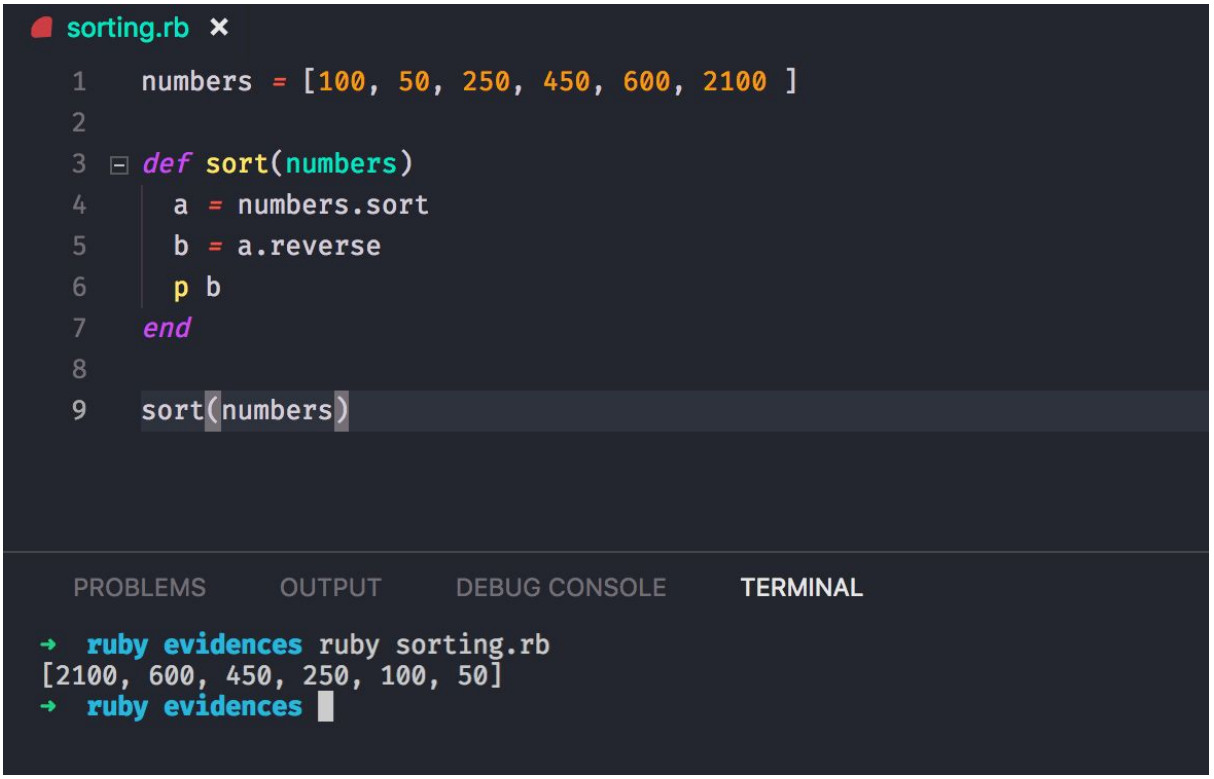
I.T 3 - Example of searching.

fruits.rb

```ruby
food = ["Mango", "Watermelon", "Melon", "Strawberry", "Banana"]

def find_fruit(food, fruit )
  if food.include?(fruit)
    print "#{fruit} is in the list"
  else
    "This is not a fruit"
  end
end

find_fruit(food, "Mango")
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

➜  ruby evidences ruby fruits.rb
Mango is in the list
➜  ruby evidences
```

I.T 4 - Example of sorting.



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

```ruby
def test_all_foods
  foods = ["charcuterie", "soup", "bread","ratatouille", "stew", "spaghetti",
  "spinach"]
  result = all_foods(@people).length
  assert_equal(7, result)
end
```

```ruby
def all_foods(people)
  foods = []
  for person in people
    foods.concat(person[:favourites][:things_to_eat])
  end
  return foods
end
```

```
# Running:

..........

Finished in 0.001607s, 6222.7754 runs/s, 6845.0529 assertions/s.

10 runs, 11 assertions, 0 failures, 0 errors, 0 skips
➜  starting_point git:(master) ×
```

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

```
def test_get_first_key
  wallets = {
    'Sandy' => 12,
    'John'  => 10,
    'Finn'  => 1356,
    'Zsolt' => 1
  }
  result = get_first_key( wallets )
  assert_equal( 'Sandy', result )
end
```

```
def get_first_key(hash)
  return hash.keys().first()
end
```

```
# Running:

.....

Finished in 0.001193s, 4191.1148 runs/s, 4191.1148 assertions/s.

5 runs, 5 assertions, 0 failures, 0 errors, 0 skips
➜  hash_array_loops_start_point git:(master) ✗
```

I.T 7 - Example of polymorphism in a program.

```
public class Shop {

    private String name;
    private double till;
    private ArrayList<ISell> stock;


    public Shop(String name, double till) {
        this.name = name;
        this.till = till;
        stock = new ArrayList<>();
    }

    public String getName() {
        return name;
    }

    public double getTill() {
        return till;
    }

    public ArrayList<ISell> getStock() {
        return stock;
    }

    public void addInstrument(Piano piano) {
        this.stock.add(piano);
    }

    public void removeInstrument() {
        this.stock.remove( index: 0);
    }

    public void addAccessory(DrumStick drumStick) {
        this.stock.add(drumStick);
    }

    public void removeAccessory() {
        this.stock.remove( index: 0);
    }
```

```java
public abstract class Instrument implements IPlay, ISell {

    private String colour;
    private double boughtPrice;
    private double soldPrice;
    private Material material;
    private InstrumentType instrumentType;
    private String soundPlayed;


    public Instrument(String colour, double boughtPrice, double soldPrice, Material material,
                      InstrumentType instrumentType, String soundPlayed) {
        this.colour = colour;
        this.boughtPrice = boughtPrice;
        this.soldPrice = soldPrice;
        this.material = material;
        this.instrumentType = instrumentType;
        this.soundPlayed = soundPlayed;
    }

    public String getColour() {
        return colour;
    }

    public double getBoughtPrice() {
        return boughtPrice;
    }

    public double getSoldPrice() {
        return soldPrice;
    }

    public Material getMaterial() {
        return material;
    }

    public InstrumentType getType() {
        return instrumentType;
    }
```

```java
public class Piano extends Instrument {

    private int numberOfKeys;
    private int numberOfStrings;

    public Piano(String colour, double boughtPrice, double soldPrice, Material material, InstrumentType instrumentType,
                 String soundPlayed, int numberOfKeys, int numberOfStrings) {
        super(colour, boughtPrice, soldPrice, material, instrumentType, soundPlayed);
        this.numberOfKeys = numberOfKeys;
        this.numberOfStrings = numberOfStrings;
    }

    public int getNumberOfKeys() {
        return numberOfKeys;
    }

    public int getNumberOfStrings() {
        return numberOfStrings;
    }


    @Override
    public String play() {
        return "I make this sound: " + this.getSoundPlayed();
    }

    public double getMarkUp(){
        return this.getSoldPrice() - this.getBoughtPrice();
    }



}
```

```java
public class Violin extends Instrument {

    private int numberOfStrings;

    public Violin(String colour, double boughtPrice, double soldPrice, Material material,
                  InstrumentType instrumentType, int numberOfStrings, String soundPlayed) {
        super(colour, boughtPrice, soldPrice, material, instrumentType, soundPlayed);
        this.numberOfStrings = numberOfStrings;
    }

    public int getNumberOfStrings() { return numberOfStrings; }

    @Override
    public String play() { return "I make this sound: " + this.getSoundPlayed(); }

    public double getMarkUp() { return this.getSoldPrice() - this.getBoughtPrice(); }
}
```

```java
public interface ISell {

    double getMarkUp();
}
```