Evidence for Implementation and Testing Unit

Fraser Brown - Cohort E17

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

```
public class Bedroom extends Room {
    private int number;
    private BedroomType type;
    private double pricePerNight;
    private int numberOfNights;

    public Bedroom(int number, BedroomType type, double pricePerNight) {
        super(type.getCapacity());
        this.type = type;
        this.number = number;
        this.pricePerNight = pricePerNight;
        this.numberOfNights = 0;
    }
```

```
public int getNumber() { return this.number; }
```

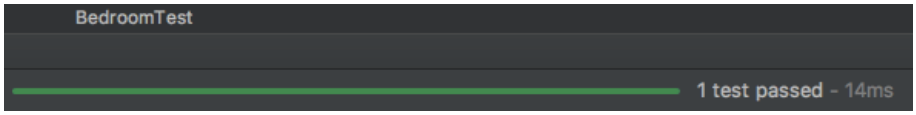
I.T 2 - Example the use of inheritance in a program.

```
import java.util.ArrayList;

public class Room {

    private int capacity;
    private ArrayList<Patron> guests;

    public Room(int capacity){
        this.capacity = capacity;
        this.guests = new ArrayList<>();
    }

    public int getCapacity() {
        return capacity;
    }

    public void setCapacity(int capacity) {
        this.capacity = capacity;
    }

    public int getPatronCount(){
        return this.guests.size();
    }

    public void addPatron(Patron patron) {
        this.guests.add(patron);
    }

    public void removePatron(Patron patron) {
        this.guests.remove(patron);
    }

}
```

```
public class Bedroom extends Room {

    private int roomNumber;
    private RoomType roomType;
    private RoomValue roomValue;

    public Bedroom(int capacity, int roomNumber, RoomType roomType, RoomValue roomValue) {
        super(capacity);
        this.roomNumber = roomNumber;
        this.roomType = roomType;
        this.roomValue = roomValue;
    }

    public int getRoomNumber() {
        return roomNumber;
    }


    public RoomType getRoomType() {
        return roomType;
    }

    public void setRoomType(RoomType roomType) {
        this.roomType = roomType;
    }

    public int getValueFromEnum() {
        return this.roomValue.getValue();
    }

}
```

```
@Before
public void before() { bedroom = new Bedroom( capacity: 2, roomNumber: 23, RoomType.DOUBLE, RoomValue.MEDIUM); }
```

```
@Test
public void getRoomType() { assertEquals(RoomType.DOUBLE, bedroom.getRoomType()); }
```



I.T 3 - Example of searching

```ruby
top_movies = [{name: "Star Wars", rating: 5, year: "1979"},
{name: "Mean Girls", rating: 4, year: "2004"},
{name: "White Chicks", rating: 5, year: "2004"}]

def find_movie_by_year(top_movies, year)
  return top_movies.find{ |movie|
  movie[:year] == year
}
end
puts find_movie_by_year(top_movies, "1979")
```

```
➜ pda_work ruby searching.rb
{:name=>"Star Wars", :rating=>5, :year=>"1979"}
➜ pda_work
```

I.T 4 – Example of sorting

```ruby
foods = ["Burger", "Hot Dog", "Pizza", "Pasta"]

def sort
  foods.sort!
end

puts foods
```

```
➜ pda_work ruby array.rb
Burger
Hot Dog
Pizza
Pasta
```

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

```ruby
foods = ["Burger", "Hot Dog", "Pizza"]

def add_to_array(food, foods)
  foods.push(food)
end

add_to_array("Pasta", foods)

print foods
```

```
➜ pda_work ruby array.rb
["Burger", "Hot Dog", "Pizza", "Pasta"]%
```

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

```ruby
foods_hash = {
  "Burger" => 1,
  "Hot Dog" => 2,
  "Pizza" => 3
}
def add_food_to_hash(foods_hash, food, rating)
  foods_hash.merge!(
    {food => rating}
  )
end

add_food_to_hash(foods_hash, "Pasta", 4)

puts foods_hash

```

```
pda_work ruby hash.rb
{"Burger"=>1, "Hot Dog"=>2, "Pizza"=>3, "Pasta"=>4}
```

I.T 7 - Example of polymorphism in a progra m

```java
package Shop;

import Behaviours.ISell;

import java.util.ArrayList;

public class Shop {

    ArrayList<ISell> stock;

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

    public int stockCount(){
        return stock.size();
    }

    public void addStock(ISell item){
        stock.add(item);
    }

    public void removeStock(ISell item){
        stock.remove(item);
    }

}
```

An example of a shop class using polymorphism. It's list of stock can be any class that implements the Sell interface.

```java
package Behaviours;

public interface ISell {

    double getMarkUp();
}
```

Interface iSell.

```java
package Shop;

import Behaviours.ISell;

public class GuitarStrings extends Accessories implements ISell {

    public GuitarStrings(String item, double buyPrice, double sellPrice){
        this.item = item;
        this.buyPrice = buyPrice;
        this.sellPrice = sellPrice;
    }

    public double getMarkUp(){
        return 1.1;
    }
}
```

```java
package Shop;

import Behaviours.ISell;

public class DrumSticks extends Accessories implements ISell {

    public DrumSticks(String item, double buyPrice, double sellPrice) {

        this.item = item;
        this.buyPrice = buyPrice;
        this.sellPrice = sellPrice;
    }

    public double getMarkUp(){
        return 1.1;
    }
}
```

Two classes implementing the iSell interface.

Below is the Shops test passing with iSell interface.

```java
import Shop.DrumSticks;
import Shop.GuitarStrings;
import Shop.Shop;
import org.junit.Before;
import org.junit.Test;

import static org.junit.Assert.assertEquals;

public class ShopTest {

    DrumSticks drumSticks;
    GuitarStrings guitarStrings;
    Shop shop;

    @Before
    public void before(){

        drumSticks = new DrumSticks( item: "Drum sticks",  buyPrice: 5.0,  sellPrice: 10.0);
        guitarStrings = new GuitarStrings( item: "Guitar Strings",  buyPrice: 10.0,  sellPrice: 20.0);
        shop = new Shop();
    }

    @Test
    public void canAddStock(){
        shop.addStock(drumSticks);
        assertEquals(shop.stockCount(),  actual: 1);
    }

    @Test
    public void canremoveStock(){
        shop.removeStock(guitarStrings);
        assertEquals(shop.stockCount(), actual: 0);
    }


}
```