Evidence for Implementation and Testing Unit.

Katarina Zemplenyiova
Cohort E17
02/March/2018

I.T 1 – Demonstrate an example of encapsulation that you have written in a program.

```
public abstract class Person {
    private int age;
    private double wallet;

    public Person(int age, double wallet){
        this.age = age;
        this.wallet = wallet;
    }

    public double getWallet() {
        return wallet;
    }

    public int getAge() {

        return age;
    }

}
```

I.T 2 Demonstrate the use of Inheritance in a program.

```
package Staff.Management;

import Staff.Employee;

public class Manager extends Employee {

    String deptName;

    public Manager(String name, String NI, double salary, String deptName) {
        super(name, NI, salary);
        this.deptName = deptName;
    }


    public String getDeptName() {
        return deptName;
    }

}
```

```java
package Staff;
public class Employee {
    protected String name;
    protected String NI;
    protected double salary;

    public Employee (String name, String NI, double salary){
        this.name = name;
        this.NI = NI;
        this.salary = salary;
    }

    public String getName() {
        return name;
    }
    public String getNI() {
        return NI;
    }

    public double getSalary() {
        return salary;
    }

    public void raiseSalary(double raise) {
        this.salary += raise;
    }

    public double payBonus() {
        double salary = this.salary;
        return salary/ 100 ;
    }

    public void setName(String inputName) {
        if (inputName != null || inputName != " ")
            this.name = inputName;
        else name = name;
    }
}
```

I.T 3 Example of searching

```ruby
orders= [{name: 'Katarina', item:"book", price: "£22"},
{name:'Anna', item:"laptop", price:"£1350"},
{name:"Tom", item:"sofa", price:"£476"}]

def find_order_by_name(orders, name)
  return orders.find{ |all_orders|
  all_orders[:name] == name
}
end

puts find_order_by_name(orders, "Anna")
```

```
➜  pda ruby list.rb
{:name=>"Anna", :item=>"laptop", :price=>"£1350"}
➜  pda
```

I.T 4 Example of sorting

```ruby
age = [ 25, 67, 38, 12, 8, 53, 11, 89, 45]

def sort_ascending(array)
  array.sort
end

puts sort_ascending(age)
```

```
➜  pda ruby sorting.rb
8
11
12
25
38
45
53
67
89
➜  pda
```

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

```ruby
items_in_wardrobe=["jeans", "shirt", "dress", "skirt", "top"]

def count_items(items)
  total_items = 0

  for item in items
    total_items +=1
  end

return "There are " + total_items.to_s + " items in wardrobe."

end

puts count_items(items_in_wardrobe)
```

```
➜  pda ruby task.rb
There are 5 items in wardrobe.
➜  pda
```

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

```ruby
my_animal = {
  name: "Molly",
  type: "caw",
  sound: "Moooooo!",
  can_make_sound: true
}

def animal_sound (animal)
  if animal[:can_make_sound] == true
    return animal[:sound]
  else
    return "I don't make any sounds."
  end
end

puts animal_sound(my_animal)
```

```
➜  pda ruby task.rb
Moooooo!
➜  pda
```

I.T 7 – Example of polymorphism in a program

```
package instruments;

public interface ISell {
    public double calculateMarkup();
}
```

```
package shops;

import instruments.ISell;

import java.util.ArrayList;

public class Shop {
    String name;
    ArrayList<ISell>stock;

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

    public void add(ISell newInput) { stock.add(newInput); }

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

    public void clearStock() { this.stock.clear(); }

    public void remove(int i) { this.stock.remove(i); }

    public double total(){
        double total = 0.0;
        for (ISell item : stock) {
            total += item.calculateMarkup();
        }
        return total;
    }
```

```java
package instruments;

public abstract class Instrument implements IPlay,ISell {
    String material;
    String colour;
    InstrumentType type;
    double priceSell;
    double priceBuy;

    public Instrument(String material, String colour, InstrumentType type, double priceSell, double priceBuy) {
        this.material = material;
        this.colour = colour;
        this.type = type;
        this.priceSell = priceSell;
        this.priceBuy = priceBuy;
    }

    @Override
    public double calculateMarkup() {
        return this.priceSell - priceBuy;
    }

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

    public String getMaterial() {
        return material;
    }

    public String getColour() {
        return colour;
    }


    public void setType(InstrumentType type) {
        this.type = type;
    }

}
```

```java
package items;

import instruments.ISell;

public abstract class Item implements ISell {

    double priceBuy;
    double addPercentage;

    public Item(double priceBuy, double addPercentage) {
        this.priceBuy = priceBuy;
        this.addPercentage = addPercentage;
    }

    public double getPriceBuy() {
        return priceBuy;
    }

    public void setPriceBuy(double priceBuy) {
        this.priceBuy = priceBuy;
    }

    public double getAddPercentage() {
        return addPercentage;
    }

    @Override
    public double calculateMarkup() {
        return this.priceBuy + (priceBuy  * addPercentage);
    }
}
```