Evidence for Implementation and Testing Unit

Jane Flucker
E19

I.T 1 Demonstrate one example of encapsulation in a program

```java
package com.example.janeflucker.todolist;

import java.io.Serializable;

/**
 * Created by janeflucker on 24/03/2018.
 */

public class Task implements Serializable {
    private int id, completed;
    private String taskName, taskDescription;

    public Task(int id, String taskName, String taskDescription, int completed) {
        this.id = id;
        this.taskName = taskName;
        this.taskDescription = taskDescription;
        this.completed = completed;
    }

    public Task(String taskName, String taskDescription) {
        this.taskName = taskName;
        this.taskDescription = taskDescription;
        this.completed = 0;
    }

    public int getId() {
        return this.id;
    }

    public String getTaskName() {
        return this.taskName;
    }

    public String getTaskDescription() {
        return this.taskDescription;
    }

    public int getCompleted() {
        return this.completed;
    }
}
```

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

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

    private String material;
    private String colour;
    private InstrumentType type;
    private int buyPrice;
    private int sellPrice;

    public Instruments(String material, String colour, InstrumentType type, int buyPrice, int sellPrice) {
        this.material = material;
        this.colour = colour;
        this.type = type;
        this.buyPrice = buyPrice;
        this.sellPrice = sellPrice;
    }

    public String getMaterial() {
        return this.material;
    }

    public String getColour() {
        return this.colour;
    }

    public String getType() {
        return this.type.getType();
    }

    public int getBuyPrice() {
        return this.buyPrice;
    }

    public int getSellPrice() {
        return this.sellPrice;
    }

    public int calculateMarkup() {
        return sellPrice - buyPrice;
    }
```

```java
public class Guitar extends Instruments{

    int numberStrings;

    public Guitar(String material, String colour, InstrumentType type, int buyPrice, int sellPrice, int numberStrings) {
        super(material, colour, type, buyPrice, sellPrice);
        this.numberStrings = numberStrings;
    }

    public int getNumberStrings() {
        return this.numberStrings;
    }

    public String play() {
        return "Strum strum...";
    }

}
```

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

```java
import org.junit.Before;
import org.junit.Test;

import static org.junit.Assert.assertEquals;

public class GuitarTest {

    Guitar guitar;

    @Before
    public void before() { guitar = new Guitar( material: "wood",  colour: "Black", InstrumentType.STRING,  buyPrice: 50,  sellPrice: 100,  numberStrings: 6); }

    @Test
    public void getMaterial() { assertEquals( expected: "wood", guitar.getMaterial()); }

    @Test
    public void getColour() { assertEquals( expected: "Black", guitar.getColour()); }

    @Test
    public void getType() { assertEquals( expected: "String", guitar.getType()); }

    @Test
    public void canGetBuyPrice() { assertEquals( expected: 50, guitar.getBuyPrice()); }

    @Test
    public void canGetSellPrice() { assertEquals( expected: 100, guitar.getSellPrice()); }

    @Test
    public void getNumberOfStrings() { assertEquals( expected: 6, guitar.getNumberStrings()); }

    @Test
    public void canPlayGuitar() { assertEquals( expected: "Strum strum...", guitar.play()); }

    @Test
    public void getMarkUpValue() { assertEquals( expected: 50, guitar.calculateMarkup()); }

}
```

```java
public enum InstrumentType {
    STRING("String"),
    WOODWIND("Woodwind"),
    PERCUSSION("Percussion"),
    KEYBOARD("Keyboard"),
    BRASS("Brass");

    private String type;

    InstrumentType(String type) { this.type = type; }

    public String getType() { return this.type; }
}
```

I.T 3 Example of searching

```ruby
def find_club_by_name( clubs, name )
  for club in clubs
    if club[:name] == name
      return "You have " + club[:stock].to_s + " clubs in stock of the " + club[:name]
    else
      return "Club not found!"
    end
  end
end

result1 = find_club_by_name(golf_club_hashes, "M1")
result2 = find_club_by_name(golf_club_hashes, "M3")

p result1
p result2
```

```
PDA — janeflucker@Janes-Air — -z
~/my_stuff/PDA
➜  PDA ruby my_hash.rb
"You have 4 clubs in stock of the M1"
"Club not found!"
➜  PDA
```

I.T 4 Example of sorting

sort_my_array.rb — ~/my_stuff/PDA

sort_my_array.rb

```ruby
@my_numbers = [22,34,18,12]

def sort_my_array()
  @my_numbers.sort
end

def sort_my_array_decending()
  @my_numbers.sort.reverse
end

result1 = sort_my_array()
p result1

result2 = sort_my_array_decending()
p result2
```

PDA — janeflucker@Ja

~/my_stuff/PDA

```
➜  PDA ruby sort_my_array.rb
[12, 18, 22, 34]
[34, 22, 18, 12]
➜  PDA
```

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

my_array.rb

```ruby
class DoubleNumbers

  attr_reader :my_numbers

  def initialize(my_numbers)
    @my_numbers = [2,4,8,12]
  end

  def double_my_numbers
    @my_numbers.each { |number| puts number * 2}
  end

end

number = DoubleNumbers.new(0)
number.double_my_numbers
```

PDA — janefl

```
➜  PDA ruby my_array.rb
4
8
16
24
➜  PDA
```

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

my_hash.rb — ~/my_stuff/PDA

my_hash.rb

```ruby
sers/janeflucker/my_stuff/PDA/my_hash.rb
    200, stock: 4 },
  { name: "M2", price: 150, stock: 5 },
  { name: "Spider putter", price: 100, stock: 2 },
  { name: "Wedge 56", price: 80, stock: 3 },
  { name: "Psi", price: 500, stock: 1 },
]

def count_clubs(clubs)
  total_clubs = 0

  for club in clubs
    total_clubs += club[:stock]
  end

  return "You have " + total_clubs.to_s() + " clubs in stock"
end

p count_clubs(golf_club_hashes)
```

PDA — janeflucker@Janes-Air

~/my_stuff/PDA

```
➜  PDA ruby my_hash.rb
"You have 15 clubs in stock"
➜  PDA
```

I.T 7 Example of polymorphism in a program

```java
import java.util.ArrayList;

public class Shop {

    Instruments instruments;
    Accessories accessories;

    private String name;
    private ArrayList<ISell> stock;

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

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

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

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

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

    public int calculateAllMarkup() {

        int total = 0;
        for (ISell item : stock) {
            total += item.calculateMarkup();
        }
        return total;
    }
```

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

```java
public abstract class Accessories implements ISell {
    private String type;
    private int buyPrice;
    private int sellPrice;


    public Accessories(String type, int buyPrice, int sellPrice) {
        this.type = type;
        this.buyPrice = buyPrice;
        this.sellPrice = sellPrice;
    }

    public String getType() {
        return this.type;
    }

    public int getBuyPrice() {
        return this.buyPrice;
    }

    public int getSellPrice() {
        return this.sellPrice;
    }

    public int calculateMarkup() {
        return sellPrice - buyPrice;
    }

}
```

```java
public class DrumSticks extends Accessories {

    int length;

    public DrumSticks(String type, int buyPrice, int sellPrice, int length) {
        super(type, buyPrice, sellPrice);
        this.length = length;
    }

    public int getLengthOfSticks() {
        return this.length;
    }
}
```

op.java × Trumpet.java × Accessories.java × Gu

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