PDA: Software Development

Implementation and Testing
Level 8

Iain Paterson
Cohort E15

I.T 1

Screenshot of Encapsulation

```java
public abstract class Competitors {

    private NationalCountry country;
    private ArrayList<Medal> medals;
    private SportPlayed sport;

    public Competitors(NationalCountry country, SportPlayed sport) {
        this.country = country;
        this.sport = sport;
        this.medals = new ArrayList<>();
    }

    public NationalCountry getCountry() { return country; }

    public ArrayList<Medal> getMedals() { return medals; }

    public SportPlayed getSport() { return sport; }

    public void addMedal(Medal medal) { this.medals.add(medal); }
}
```

I.T 2

Screenshot of the use of Inheritance

```java
package com.example.user.raysmusicstore;


public abstract class Instrument {

    private String brand;
    private String colour;
    private String family;


    public Instrument(String brand, String colour, String family) {
        this.brand = brand;
        this.colour = colour;
        this.family = family;
    }

    public String getBrand(){
        return this.brand;
    }

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

    public String getFamily(){
        return this.family;
    }

}

```

A class that inherits from the previous class

```java
package com.example.user.raysmusicstore;


public class KettleDrum extends Instrument implements Playable  {

     String skinType;

    public KettleDrum( String brand, String colour,String family, String skinType ){
        super(brand, colour, family);
        this.skinType = skinType;
    }

    public String play() { return ( "BOOM BOOM BOOM" ); }

    public String getSkin() {
        return skinType;
    }


}
```

An object in the inherited class

```java
public class KettleDrumTest {

    KettleDrum kettleDrum;

    @Before
    public void before(){
        kettleDrum = new KettleDrum( "Timpani", "Bronze", "Percussion", "Cow Hide" );
    }

    @Test
    public void testKettleDrumSkin() { assertEquals( "Cow Hide", kettleDrum.getSkin()); }

    @Test
    public void testKettleDrumMakesNoise() { assertEquals( "BOOM BOOM BOOM", kettleDrum.play()); }

    @Test
    public void testKettleDrumColour() { assertEquals( "Bronze", kettleDrum.getColour()); }

    @Test
    public void testKettleDrumBrand(){
        assertEquals( "Timpani", kettleDrum.getBrand());
    }

    @Test
    public void testKettleDrumFamily() { assertEquals( "Percussion", kettleDrum.getFamily()); }


}
```

A method that uses the information from the inherited class

```java
    @Test
    public void testKettleDrumBrand(){
        assertEquals( "Timpani", kettleDrum.getBrand());
    }
```

```
Run KettleDrumTest.testKettleDrumBrand
1 test passed - 1ms
KettleDrumTest (com.exampl 1ms
    testKettleDrumBrand 1ms
"/Applications/Android Studio.app/Contents/jre/jdk/Contents/Home/bin/java" ...

Process finished with exit code 0
```

I.T 3

Demonstration of searching data in a program

```ruby
def self.find( id )
  sql = "SELECT * FROM animals
  WHERE id = $1;"
  values = [id]
  found = SqlRunner.run( sql, values )
  return found.map{ |animal_type| Animal.new( animal_type )}[0]
end
```

Searched for data

```
[1] pry(main)> Animal.find(13)
=> #<Animal:0x007fad2236d310
 @age=4,
 @date_entered="2017-05-12",
 @id=13,
 @name="Rover",
 @photo="dog.jpg",
 @type="dog",
 @vet_id=5>
```

```
[2] pry(main)> Animal.find(16)
=> #<Animal:0x007fad23099e58
 @age=7,
 @date_entered="2017-04-09",
 @id=16,
 @name="Willow",
 @photo="cat.jpg",
 @type="cat",
 @vet_id=6>
```

I.T 4

Demonstration of sorting data in a program

Array to be sorted

```
movie_titles = [
    'Matrix',
    'Stand By Me',
    'Conspiracy Theory',
    'Avatar',
    'Robocop',
    'Beverly Hills Cop'
]
```

Function that uses the array

```ruby
def get_movies(array)
  return array.sort
end

puts get_movies(movie_titles)
```

Sorted array in alphabetical order

psql | ruby

```
pda_example ruby array_sort.rb
Avatar
Beverly Hills Cop
Conspiracy Theory
Matrix
Robocop
Stand By Me
```

I.T 5 *Demonstrate the use of a Hash*

i_t_5.rb | i_t_6.rb

```ruby
teams_stadiums = { :rangers => 'Ibrox',
                   :juventus => 'Juventus Stadium',
                   :manchester_United => 'Old Trafford',
                   :bayern_Munich => 'Allianz Arena',
                   :barcelona => 'Camp Nou',
                   :benfica => 'Estadio da Luz'}
```

Function that uses hash

```ruby
def rangers_stadium(hash)
  return hash[:rangers]
end


puts rangers_stadium(teams_stadiums)
```

Result of function

```
ruby    ruby app...    psql
→  week_2 ruby i_t_6.rb
Ibrox
→  week_2
```

I.T 6

Demonstration the use of an Array

```ruby
i_t_5.rb    i_t_6.rb
football_teams = [ 'Rangers', 'Juventus', 'Manchester United',
                   'Bayern Munich', 'Barcelona', 'Benfica' ]
```

Function that uses array

```ruby
def get_team(array)
 return array.reverse
end


puts get_team(football_teams)
```

Result of function

```
ruby    ruby app...    psql

[→  week_2 ruby i_t_5.rb
Benfica
Barcelona
Bayern Munich
Manchester United
Juventus
Rangers
→  week_2
```

I.T 7

The use of polymorphism

The Amplifier Class and GuitarPick Class which implements the Sellable interface and calculateMarkup() method

```java
public class Amplifier implements Sellable {

    String type;
    double buyPrice;
    double sellPrice;

    public Amplifier( String type, double buyPrice ){...}

    public double getBuyPrice() { return buyPrice; }

    public String getItemType() { return type; }

    public double calculateMarkup() {
        return (0.5 * buyPrice);
    }

    public double getSellPrice() {
        return ( buyPrice += calculateMarkup());
    }

}
```

```java
public class GuitarPick implements Sellable {

    String type;
    double buyPrice;
    double sellPrice;

    public  GuitarPick( String type, double buyPrice ){...}

    public String getItemType() { return type; }

    public double getBuyPrice() { return buyPrice; }


    public double calculateMarkup() {
        return (0.5 * buyPrice);
    }

    public double getSellPrice() {
        return ( buyPrice += calculateMarkup());
    }
}
```

The Sellable interface with a calculateMarkup() method

```java
package com.example.user.raysmusicstore;

/**
 * Created by user on 09/09/2017.
 */

public interface Sellable {

    double calculateMarkup();
}
```

Result of the calculateMarkup() method being tested on the Amplifier Class

```java
public class AmplifierTest {

    Amplifier amplifier;

    @Before
    public void before(){
        amplifier = new Amplifier( type: "Amplifier",  buyPrice: 20.00);
    }

    @Test
    public void testAmplifierType() { assertEquals( expected: "Amplifier", amplifier.getItemType()); }

    @Test
    public void testAmplifierBuyPrice() { assertEquals( expected: 20.00, amplifier.getBuyPrice(),  delta: 0.01); }

    @Test
    public void testSellPrice() { assertEquals( expected: 30.00, amplifier.getSellPrice(),  delta: 0.01); }


    @Test
    public void testMarkUpOfItem() {
        assertEquals( expected: 10.00, amplifier.calculateMarkup(),  delta: 0.01);
    }
}
```

The test passing

```
Run  AmplifierTest.testMarkUpOfItem
1 test passed – 1ms
AmplifierTest (com.example.user.ray 1ms
    testMarkUpOfItem 1ms
"D:\Program Files\Android\Android Studio\jre\bin\java" ...

Process finished with exit code 0
```

Result of the calculateMarkup() method being tested on the GuitarPick Class

```java
10  public class GuitarPickTest {
11
12      GuitarPick guitarPick;
13
14      @Before
15      public void before() {
16          guitarPick = new GuitarPick( type: "Guitar Pick",  buyPrice: 2.00);
17      }
18
19      @Test
20      public void getType() { assertEquals(  expected: "Guitar Pick", guitarPick.getItemType()); }
23
24      @Test
25      public void testBuyPrice() { assertEquals(  expected: 2.00, guitarPick.getBuyPrice(),  delta: 0.01); }
28
29      @Test
30      public void testSellPrice() { assertEquals(  expected: 3.00, guitarPick.getSellPrice(),  delta: 0.01); }
33
34      @Test
35      public void testMarkUpOfItem() {
36          assertEquals(  expected: 1.00, guitarPick.calculateMarkup(),  delta: 0.01);
37      }
38  }
39
```

The test passing

```
1 test passed – 6ms
GuitarPickTest (com.example.user.ra 6ms
    testMarkUpOfItem 6ms

"D:\Program Files\Android\Android Studio\jre\bin\java" ...

Process finished with exit code 0
```

The Shop Class which has methods to add stock to an ArrayList of Sellable items

```java
import java.util.ArrayList;

public class Shop {


    private ArrayList<Sellable> stock;

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

    public void addStock(Sellable sellable) {
        this.stock.add(sellable);
    }


    public ArrayList<Sellable> getStock() {
        return this.stock;
    }




}
```

The Test to see if Sellable items will be added to the Array List

```java
    @Test
    public void testStockOfSellableItems() {
        shop = new Shop();
        shop.addStock(amplifier);
        shop.addStock(pick);
        assertEquals( expected: 2 , shop.getStock().size());
    }
```

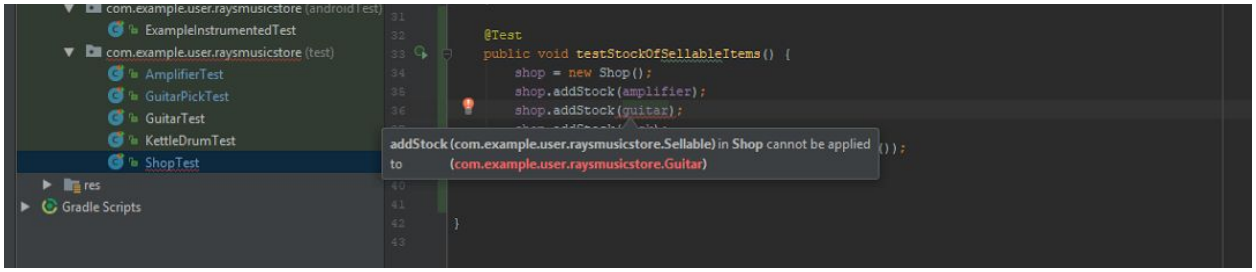
The test passing

```
1 test passed – 1ms
ShopTest (com.example.user.raysmu 1ms
    testStockOfSellableItems 1ms
"D:\Program Files\Android\Android Studio\jre\bin\java" ...

Process finished with exit code 0
```

The test throws a syntax error to tell me the Guitar Class doesn't use the Sellable interface

```
@Test
public void testStockOfSellableItems() {
    shop = new Shop();
    shop.addStock(amplifier);
    shop.addStock(guitar);
    shop.addStock(pick);
    assertEquals(  expected: 2 , shop.getStock().size());
}
```



The Guitar Class which only implements Playable interface but not the Sellable interface leading to the syntax error

```
public class Guitar extends Instrument implements Playable {

    int numberOfStrings;

    public Guitar( String brand, String colour, String family, int numberOfStrings){
        super(brand, colour, family);
        this.numberOfStrings = numberOfStrings;
    }

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

    public int getStrings() { return numberOfStrings; }

}
```