

## Evidence for Implementation and Testing Unit

Rachel Johnson

E19

### I.T 1 - Example of encapsulation

```
public abstract class Kaiju implements IActionsForKaiju {
    private String name;
    private int healthValue;
    private int attackValue;

    public Kaiju(String name, int healthValue, int attackValue) {
        this.name = name;
        this.healthValue = healthValue;
        this.attackValue = attackValue;
    }

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

    public int getHealthValue() { return this.healthValue; }

    public int getAttackValue() { return this.attackValue; }

    public void attack(Vehicle vehicle) { vehicle.reduceHealthValue(this.attackValue); }

    public void attack(Building building) {
        building.reduceHealthValue(this.attackValue);
    }

    public void reduceHealthValue(int number) {
        if (number > this.healthValue) {
            this.healthValue = 0;
        } else {
            this.healthValue -= number;
        }
    }
}
```

```
public abstract class Vehicle implements IGettersForVehicleAndBuilding, IActionsForVehicle {
    private String type;
    private int healthValue;

    public Vehicle(String type, int healthValue) {
        this.type = type;
        this.healthValue = healthValue;
    }

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

    public int getHealthValue() { return this.healthValue; }

    public void setHealthValue(int number) { this.healthValue = number; }

    public void reduceHealthValue(int number) {
        if (number > this.healthValue) {
            this.healthValue = 0;
        } else {
            this.healthValue -= number;
        }
    }

    public void attackWithTearGas(Kaiju kaiju) {
        kaiju.reduceHealthValue( number: 20);
    }

    public void attackWithGrenades(Kaiju kaiju) { kaiju.reduceHealthValue( number: 40); }
}
```

## I.T 2 - Example of inheritance

### Screenshot of parent class (Instrument):

```
import Enums.InstrumentType;
import Interfaces.IPlay;

public abstract class Instrument extends Stock implements IPlay {

    protected String material;
    protected String colour;
    protected InstrumentType instrumentType;

    public Instrument(int uniqueID, double buyPrice, double sellPrice, String description, String material, String colour, InstrumentType instrumentType) {
        super(uniqueID, buyPrice, sellPrice, description);
        this.material = material;
        this.colour = colour;
        this.instrumentType = instrumentType;
    }

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

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

    public InstrumentType getType() { return this.instrumentType; }

    public abstract String playSound();
}
```

### Screenshot of inheriting class (Guitar):

```
import Enums.GuitarType;
import Enums.InstrumentType;

public class Guitar extends Instrument {

    private GuitarType guitarType;
    private int numberOfStrings;

    public Guitar(int uniqueID, double buyPrice, double sellPrice, String description, String material, String colour, GuitarType guitarType, int numberOfStrings) {
        super(uniqueID, buyPrice, sellPrice, description, material, colour, InstrumentType.STRING);
        this.guitarType = guitarType;
        this.numberOfStrings = numberOfStrings;
    }

    @Override
    public String playSound() { return "Strum strum"; }

    public GuitarType getGuitarType() { return this.guitarType; }

    public int getNumberOfStrings() { return this.numberOfStrings; }
}
```

Screenshot of Guitar object and methods from the inherited class:

```
import Enums.GuitarType;
import Enums.InstrumentType;
import org.junit.Before;
import org.junit.Test;

import static org.junit.Assert.assertEquals;

public class GuitarTest {

    private Guitar guitar;
    private GuitarType guitarType;

    @Before
    public void before() {
        guitar = new Guitar( uniqueID: 1, buyPrice: 350, sellPrice: 490, description: "guitar", material: "maple",
                             colour: "surf green", guitarType.ELECTRIC, numberOfStrings: 6);
    }

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

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

    @Test
    public void canGetType() {
        assertEquals(InstrumentType.STRING, guitar.getType());
    }

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

    @Test
    public void canGetGuitarType() {
        assertEquals(GuitarType.ELECTRIC, guitar.getGuitarType());
    }
}
```

### I.T 3 - Example of searching

#### Table to be searched

id	title	author
1	Brave New World	Aldous Huxley
2	Oryx and Crake	Margaret Atwood
3	The Timetraveller's Wife	Audrey Niffenegger
4	Charlie and the Chocolate Factory	Roald Dahl
5	The Girl with All the Gifts	M R Carey
6	Chinese Cinderella	Adeline Yen Mah
7	Cloud Atlas	David Mitchell
8	A Clockwork Orange	Anthony Burgess
9	The Lion, the Witch and the Wardrobe	C S Lewis
10	1984	George Orwell

(10 rows)

#### Search function

Function searches within the table for a specified id number and returns the associated book

```
1 require('pg')
2
3 class Book
4
5   attr_reader :id, :title, :author
6
7   def initialize(inputs)
8     @id = inputs['id'].to_i if inputs['id']
9     @title = inputs['title']
10    @author = inputs['author']
11  end
12
13  def Book.find_by_id(id)
14    db = PG.connect({dbname: "books", host: "localhost"})
15    sql = "SELECT * FROM books WHERE id = $1;"
16    values = [id]
17    db.prepare("find", sql)
18    result = db.exec_prepared("find", values)
19    db.close()
20    return result.map {|book| Book.new(book)}
21  end
22
23 end
24
25 p Book.find_by_id(5)
26
```

#### Result

```
[→ PDA git:(master) × ruby IT3_evidence.rb
[#<Book:0x007fcc45148778 @id=5, @title="The Girl with All the Gifts", @author="M R Carey">]
→ PDA git:(master) × ]
```

## I.T 4 - Example of sorting

Table to be sorted is same as that used for I.T 3.

### Function

Function sorts the table by title in ascending order.

```
1  require('pg')
2
3  class Book
4
5      attr_reader :id, :title, :author
6
7      def initialize(inputs)
8          @id = inputs['id'].to_i if inputs['id']
9          @title = inputs['title']
10         @author = inputs['author']
11     end
12
13     def Book.sort()
14         db = PG.connect({dbname: "books", host: "localhost"})
15         sql = "SELECT * FROM books ORDER BY title ASC;"
16         db.prepare("sort", sql)
17         result = db.exec_prepared("sort", [])
18         db.close()
19         return result.map {|book| Book.new(book)}
20     end
21
22 end
23
24 Book.sort().each{|book| p book}
25 # Added .each{...} functionality above so output of sort function is easier to
26 # read in the terminal
27
```

### Result

```
[→ PDA git:(master) × ruby IT4_evidence.rb ]
#<Book:0x007ffc910891d0 @id=10, @title="1984", @author="George Orwell">
#<Book:0x007ffc91089108 @id=8, @title="A Clockwork Orange", @author="Anthony Burgess">
#<Book:0x007ffc91088f28 @id=1, @title="Brave New World", @author="Aldous Huxley">
#<Book:0x007ffc91088ac8 @id=4, @title="Charlie and the Chocolate Factory", @author="Roald Dahl">
#<Book:0x007ffc91088938 @id=6, @title="Chinese Cinderella", @author="Adeline Yen Mah">
#<Book:0x007ffc910887f8 @id=7, @title="Cloud Atlas", @author="David Mitchell">
#<Book:0x007ffc91088708 @id=2, @title="Oryx and Crake", @author="Margaret Atwood">
#<Book:0x007ffc910882a8 @id=5, @title="The Girl with All the Gifts", @author="M R Carey">
#<Book:0x007ffc91088050 @id=9, @title="The Lion, the Witch and the Wardrobe", @author="C S Lewis">
#<Book:0x007ffc9108a918 @id=3, @title="The Timetraveller's Wife", @author="Audrey Niffenegger">
→ PDA git:(master) ×
```

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

### Array and function

```
1  cheesecake_ingredients = ["biscuits", "butter", "cream cheese", "sugar", "double cream", "raspberries"]
2
3  def display_ingredients(ingredients)
4    p "The ingredients in this dish are:"
5    ingredients.each {|ingredient| p ingredient}
6  end
7
8  display_ingredients(cheesecake_ingredients)
```

### Result

```
[➔ PDA git:(master) ✕ ruby IT5_evidence.rb ]
"The ingredients in this dish are:"
"biscuits"
"butter"
"cream cheese"
"sugar"
"double cream"
"raspberries"
➔ PDA git:(master) ✕
```



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

### Hash and function

```
1  menu_prices = {
2    starter: 6.50,
3    main: 17.50,
4    dessert: 5.50
5  }
6
7  def total_cost(menu)
8    running_total = 0
9    menu.each {|key, value| running_total += value}
10   p "The total cost of the meal is £#{ '%.2f' % running_total}"
11 end
12
13 total_cost(menu_prices)
14
```

### Result

```
[→ PDA git:(master) ✖ ruby IT6_evidence.rb
"The total cost of the meal is £29.50"
→ PDA git:(master) ✖ █]
```

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

The following code snippets illustrate the implementation of polymorphism in a Java program:

```
1 public interface IDriveable {
2     int driveDistance(int distance);
3 }
4
```

```
1 public abstract class Vehicle implements IDriveable {
2
3     double averageSpeed;
4
5     public Vehicle(double averageSpeed) { this.averageSpeed = averageSpeed; }
6
7     public double getAverageSpeed() { return this.averageSpeed; }
8
9     @Override
10    public int driveDistance(int distance) { return (int) (distance / getAverageSpeed()); }
11 }
12
```

```
1 public class Driver {
2
3     String name;
4     IDriveable vehicle;
5
6     public Driver(String name, IDriveable vehicle) {
7         this.name = name;
8         this.vehicle = vehicle;
9     }
10
11    public String getName() {
12        return this.name;
13    }
14
15    public IDriveable getVehicle() { return this.vehicle; }
16
17    public int driveDistance(int distance) { return this.vehicle.driveDistance(distance); }
18
19    public void setVehicle(IDriveable vehicle) { this.vehicle = vehicle; }
20 }
21
```



```
▼ githomework-week7-day3 ~/codeclan_work/
├── .gradle
├── .idea
├── gradle
├── out
├── src
│   └── main
│       ├── java
│       │   ├── DodgegCar
│       │   ├── Driver
│       │   ├── IDriveable
│       │   ├── QuadBike
│       │   ├── Van
│       │   └── Vehicle
│       ├── resources
│       └── test
│           └── java
│               ├── DodgegCarTest
│               ├── DriverTest
│               ├── QuadBikeTest
│               └── VanTest
1 public class DodgegCar extends Vehicle {
2
3     int numberOfSeats;
4
5     public DodgegCar(double averageSpeed, int numberOfSeats) {
6         super(averageSpeed);
7         this.numberOfSeats = numberOfSeats;
8     }
9
10    public int getNumberOfSeats() {
11        return this.numberOfSeats;
12    }
13
14 }
```

```
▼ githomework-week7-day3 ~/codeclan_work/
├── .gradle
├── .idea
├── gradle
├── out
├── src
│   └── main
│       ├── java
│       │   ├── DodgegCar
│       │   ├── Driver
│       │   ├── IDriveable
│       │   ├── QuadBike
│       │   ├── Van
│       │   └── Vehicle
│       ├── resources
│       └── test
│           └── java
│               ├── DodgegCarTest
│               ├── DriverTest
│               ├── QuadBikeTest
│               └── VanTest
1 public class QuadBike extends Vehicle {
2
3     int engineCapacity;
4
5     public QuadBike(double averageSpeed, int engineCapacity) {
6         super(averageSpeed);
7         this.engineCapacity = engineCapacity;
8     }
9
10    public int getEngineCapacity() {
11        return this.engineCapacity;
12    }
13
14 }
```