Evidence for Implementation and Testing Unit

Name: Sarah Murphy
Cohort: E21

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

```java
public class Park {
    private String name;
    private double admissionPriceAdult;
    private double till;
    private int visitorCapacity;
    private int paddockCapacity;
    private ArrayList<Paddock> paddockList;
    private ArrayList<Visitor> visitorList;

    public Park(String name, double admissionPriceAdult, int visitorCapacity, int paddockCapacity){
        this.name = name;
        this.admissionPriceAdult = admissionPriceAdult;
        this.till = 0;
        this.visitorCapacity = visitorCapacity;
        this.paddockCapacity = paddockCapacity;
        this.paddockList = new ArrayList<Paddock>();
        this.visitorList =new ArrayList<Visitor>();
    }

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

    public int getVisitorCount() {
        return this.visitorList.size();
    }

    public double getAdmissionPriceAdult() {
        return this.admissionPriceAdult;
    }

    public double getTill() {
        return this.till;
    }
```

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

A class called Room:

Room.java | Hotel.java | HotelTest.java | Guest.java

```java
package roomType;

import java.util.ArrayList;
import Guests.Guest;


public abstract class Room {
    private int capacity;
    private ArrayList<Guest> guestlist;

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

```

A class called Conference that inherits capacity from the previous class (Room):

```java
package roomType;


public class Conference extends Room {
    private String name;
    private double dailyRate;


    public Conference (int capacity, String name, double dailyRate) {
        super(capacity);
        this.name = name;
        this.dailyRate = dailyRate;
    }


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

    public double getDailyRate(){
        return this.dailyRate;
    }
}

```

An object in the inherited class - i.e. a new conference room that inherits capacity

```java
public class ConferenceTest {
    Conference conference;
    Guest guest;

    @Before
    public void before (){
        conference = new Conference( capacity: 10,  name: "Islay Suite",  dailyRate: 150.00);
    }
```

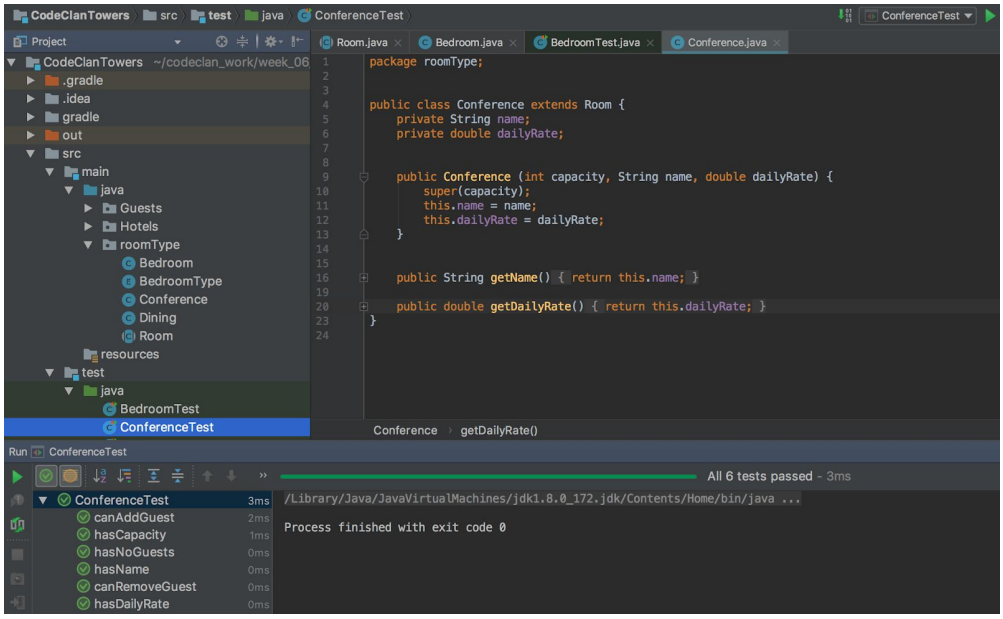
A method that uses the information inherited from another class

Capacity method in Room Class:

```
public abstract class Room {
    private int capacity;
    private ArrayList<Guest> guestlist;

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

    public int getCapacity() {
        return this.capacity;
    }
```

Testing the method inherited from another class:

```
@Before
public void before (){
    conference = new Conference( capacity: 10,  name: "Islay Suite",  dailyRate: 150.00);
}

@Test
public void hasCapacity(){
    assertEquals( expected: 10, conference.getCapacity());
}
```



I.T 3 - Example of searching

Function that searches all the customer data:

```ruby
def self.map_items(customer_data)
  result = customer_data.map { |customer|
  Customer.new(customer) }
  return result
end

def self.all()
  sql = "SELECT * FROM customers"
  customer_data = SqlRunner.run(sql)
  return Customer.map_items(customer_data)
end
```

Result of the function running - Customer.all

```
[➜  codeclan_cinema git:(master) ✗ ruby db/console.rb

From: /Users/user/codeclan_work/week_03/homework/codeclan_cinema/db/console.rb @
 line 72 :

    67:   'customer_id' => customer2.id
    68:   })
    69: ticket4.save()
    70:
    71: binding.pry
 => 72: nil

[1] pry(main)> Customer.all
=> [#<Customer:0x007fbfcdb8c358 @funds="100", @id=55, @name="Brad Pitt">,
 #<Customer:0x007fbfcdb8c240 @funds="800", @id=56, @name="Angelina Jolie">,
 #<Customer:0x007fbfcdb8c128 @funds="800", @id=57, @name="Jennifer Aniston">]
[2] pry(main)>
```

Database view below

```
➜ codeclan_cinema git:(master) ✗ psql -d codeclan_cinema -f db/codeclan_cinema.sql
DROP TABLE
DROP TABLE
DROP TABLE
CREATE TABLE
CREATE TABLE
CREATE TABLE
➜ codeclan_cinema git:(master) ✗ psql -d codeclan_cinema
psql (10.3)
Type "help" for help.

codeclan_cinema=# SELECT * FROM customers;
 id |       name       | funds
----+------------------+-------
  1 | Brad Pitt        |   100
  2 | Angelina Jolie   |   800
  3 | Jennifer Aniston |   800
(3 rows)

codeclan_cinema=#
```

I.T 4 – Example of sorting

Function that sorts data by films - ability to select a customer and return all the films they have tickets for.

```
def films()
  sql = "SELECT films.* FROM films
  INNER JOIN tickets
  ON tickets.film_id = films.id WHERE
  customer_id = $1"
  values = [@id]
  film_data = SqlRunner.run(sql,values)
  return Film.map_items(film_data)
end
```

Result of the function running - customer1.films

```
[➜  codeclan_cinema git:(master) ✗ ruby db/console.rb

From: /Users/user/codeclan_work/week_03/homework/codeclan_cinema/db/console.rb @
 line 72 :

    67:   'customer_id' => customer2.id
    68:   })
    69: ticket4.save()
    70:
    71: binding.pry
 => 72: nil

[1] pry(main)> customer1.films
=> [#<Film:0x007fce532d52e8 @id=4, @price="10", @title="Pulp Fiction">,
 #<Film:0x007fce532d4f50 @id=5, @price="8", @title="A Prophet">]
[2] pry(main)>
```

Database view

```
[➜  codeclan_cinema git:(master) ✗ psql -d codeclan_cinema
psql (10.3)
Type "help" for help.

codeclan_cinema=# SELECT films.* FROM films INNER JOIN tickets ON tickets.film_i
d = films.id WHERE customer_id = 4;
 id |    title     | price
----+--------------+-------
  4 | Pulp Fiction |    10
  5 | A Prophet    |     8
(2 rows)

codeclan_cinema=#
```

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

An array in a program - songs are part of an array.

```
def setup
  @song1 = Song.new("Mama Mia", "Abba")
  @song2 = Song.new("Dancing Queen", "Abba")
  @song3 = Song.new("Waterloo", "Abba")
  @song4 = Song.new("Money Money Money",
  "Abba")
  songs = [@song1, @song2, @song3]
  @room = Room.new("Vegas", 10, @guest_list,
  songs, 20, 0)
```

A function that uses the array

```ruby
def test_add_song_to_room
  @room.add_a_song(@song4)
  assert_equal(4,@room.songs.count())
end
```

```ruby
def add_a_song(song)
  @songs.push(song)
end
```

The result of the function running

```
➜  day_5 git:(master) ✗ ruby specs/room_spec.rb
Run options: --seed 34556

# Running:

....

Finished in 0.001148s, 3484.3206 runs/s, 3484.3206 assertions/s.

4 runs, 4 assertions, 0 failures, 0 errors, 0 skips
```

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

A hash in a program

```ruby
class Customer

  attr_reader :id
  attr_accessor :name, :funds

  def initialize(options)
    @id = options['id'].to_i if options['id']
    @name = options['name']
    @funds = options['funds']
  end
```

A function that uses the hash

```ruby
class TestCustomer < MiniTest::Test

  def setup
    @customer1 = Customer.new({
      "name"=>"Brad Pitt",
      "funds" => 100
    })
  end

  def test_name
    assert_equal("Brad Pitt", @customer1.name())
  end
```

The result of the function running

```
[➜  codeclan_cinema git:(master) ✗ ruby specs/customer_specs.rb
Run options: --seed 23379

# Running:

.

Finished in 0.000987s, 1013.1712 runs/s, 1013.1712 assertions/s.

1 runs, 1 assertions, 0 failures, 0 errors, 0 skips
```

I.T 7 - Example of polymorphism in a program

Piano class inherits from a class called Instrument and implements an interface called IPlay

```java
package Instruments;

import behaviours.IPlay;

public class Piano extends Instrument implements IPlay {

    private String make;

    public Piano(String description, double buyingPrice, double sellingPrice, String colour, String material, InstrumentType instrumentType,
        super(description, buyingPrice, sellingPrice, colour, material, instrumentType);
        this.make = make;
    }

    public String getMake() {
        return this.make;
    }
    public String play(String sound) {
        return "Piano can " + sound;
    }
}
```

Guitar class inherits from a class called Instrument and implements an interface called IPlay

```java
package Instruments;

import behaviours.IPlay;

public class Guitar extends Instrument implements IPlay {

    private int strings;

    public Guitar(String description, double buyingPrice, double sellingPrice, String colour, String material, InstrumentType instrumentType, int strings) {
        super(description, buyingPrice, sellingPrice, colour, material, instrumentType);
        this.strings = strings;
    }

    public int getNumberOfStrings() { return this.strings; }
    public String play(String sound) { return "Guitar can " + sound; }

}
```

Instrument class

```java
package Instruments;
import ...

public abstract class Instrument extends Item{

   private String colour;
    private String material;
    private InstrumentType instrumentType;

    public Instrument(String description, double buyingPrice, double sellingPrice, String colour, String material, Instrum
        super(description,buyingPrice,sellingPrice);
        this.colour = colour;
        this.material = material;
        this.instrumentType = instrumentType;
    }

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

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

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



}
```

Item class

```java
package Items;

import behaviours.ISell;

public abstract class Item implements ISell {
    protected String description;
    protected double buyingPrice;
    protected double sellingPrice;

    public Item(String description, double buyingPrice, double sellingPrice){
        this.description = description;
        this.buyingPrice = buyingPrice;
        this.sellingPrice = sellingPrice;
    }

    public String getDescription() {
        return this.description;
    }

    public double getBuyingPrice() {
        return this.buyingPrice;
    }

    public double getSellingPrice() {
        return this.sellingPrice;
    }
    public double calculateMarkup() {
        return getSellingPrice() - getBuyingPrice();
    }
}
```

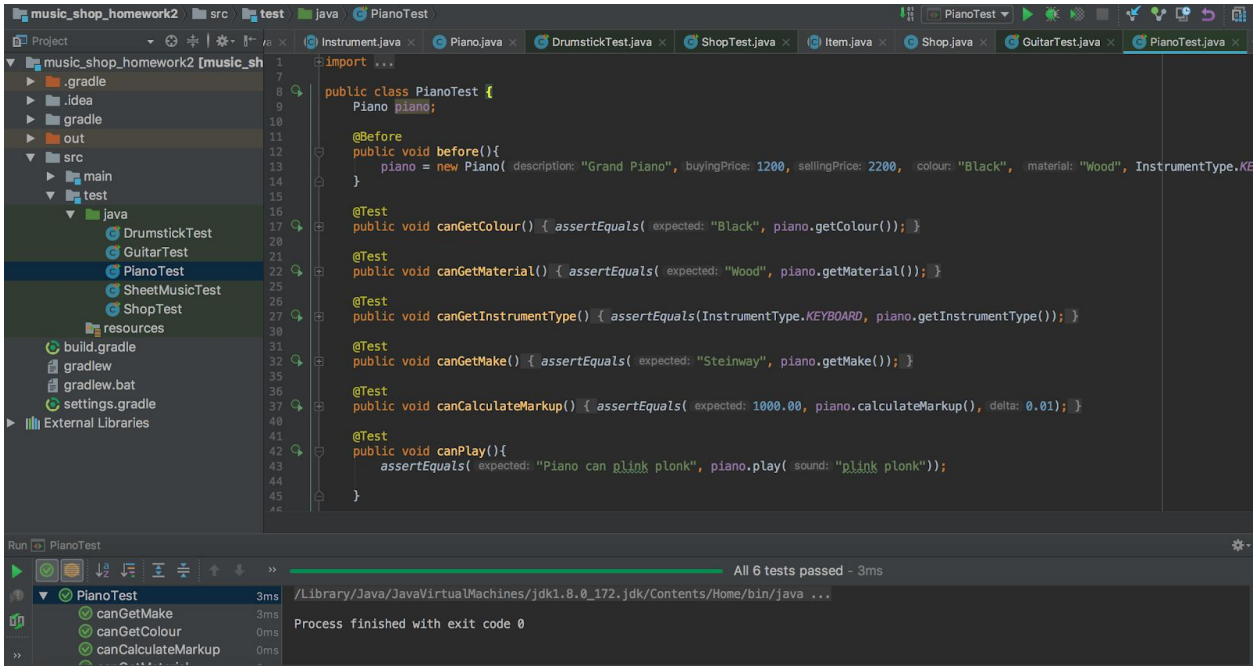
Shop class - contains the array list of isell items

```java
package Shop;

import ...

public class Shop {
    protected String name;
    protected ArrayList<ISell> stockList;

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

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

    public int stockListCount() { return stockList.size(); }


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

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

    public double getProfit() {
        double profit = 0;

        for (ISell item : this.stockList){
            profit += item.calculateMarkup();
        }
        return profit;

    }
}
```

The method “play” in the interface IPlay

```java
package behaviours;

public interface IPlay {

    String play(String sound);
}
```

The method “play” being implemented and tested in the PianoTest


music_shop_homework2 › src › test › java › PianoTest
PianoTest
Project
Instrument.java
Piano.java
DrumstickTest.java
ShopTest.java
Item.java
Shop.java
GuitarTest.java
PianoTest.java
music_shop_homework2 [music_sh
.gradle
.idea
gradle
out
src
main
test
java
DrumstickTest
GuitarTest
PianoTest
SheetMusicTest
ShopTest
resources
build.gradle
gradlew
gradlew.bat
settings.gradle
External Libraries
import ...
public class PianoTest {
Piano piano;
@Before
public void before(){
piano = new Piano(description: "Grand Piano", buyingPrice: 1200, sellingPrice: 2200, colour: "Black", material: "Wood", InstrumentType.KE
}
@Test
public void canGetColour() { assertEquals(expected: "Black", piano.getColour()); }
@Test
public void canGetMaterial() { assertEquals(expected: "Wood", piano.getMaterial()); }
@Test
public void canGetInstrumentType() { assertEquals(InstrumentType.KEYBOARD, piano.getInstrumentType()); }
@Test
public void canGetMake() { assertEquals(expected: "Steinway", piano.getMake()); }
@Test
public void canCalculateMarkup() { assertEquals(expected: 1000.00, piano.calculateMarkup(), delta: 0.01); }
@Test
public void canPlay(){
assertEquals(expected: "Piano can plink plonk", piano.play(sound: "plink plonk"));
}
Run PianoTest
All 6 tests passed - 3ms
PianoTest 3ms
canGetMake 3ms
canGetColour 0ms
canCalculateMarkup 0ms
/Library/Java/JavaVirtualMachines/jdk1.8.0_172.jdk/Contents/Home/bin/java ...
Process finished with exit code 0