

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.

```
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 x Hotel.java x HotelTest.java x Guest.java x
1 package roomType;
2
3 import java.util.ArrayList;
4 import Guests.Guest;
5
6
7 public abstract class Room {
8     private int capacity;
9     private ArrayList<Guest> guestlist;
10
11     public Room(int capacity) {
12         this.guestlist = new ArrayList<>();
13         this.capacity = capacity;
14     }
15 }
```

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

```
1 package roomType;
2
3
4 public class Conference extends Room {
5     private String name;
6     private double dailyRate;
7
8
9     public Conference (int capacity, String name, double dailyRate) {
10         super(capacity);
11         this.name = name;
12         this.dailyRate = dailyRate;
13     }
14
15
16     public String getName(){
17         return this.name;
18     }
19
20     public double getDailyRate(){
21         return this.dailyRate;
22     }
23 }
24 }
```

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

```
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());
}
```

The screenshot shows an IDE with the following components:

- Project Explorer:** Shows a project named 'CodeClanTowers' with a 'test' directory containing 'ConferenceTest'.
- Code Editor:** Displays the 'ConferenceTest' class. It includes a 'before' method that creates a 'Conference' object and a 'hasCapacity' test method. The 'Conference' class is also visible, extending 'Room' and implementing 'getCapacity'.
- Run Console:** Shows the execution of the 'ConferenceTest' class. It reports 'All 6 tests passed - 3ms' and 'Process finished with exit code 0'.

```
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; }
}
```

Run: ConferenceTest

All 6 tests passed - 3ms

Process finished with exit code 0

I.T 3 - Example of searching

Function that searches all the customer data:

```
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 (not sure if this is required - delete?)

```

[→ 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 (not sure if this is required - delete?)

```
[➔ 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_id =
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

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

```
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

```

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
end

```

A function that uses the hash

```

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
end

```

The result of the function running


```

→ codeclan_cinema git:(master) x 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

```

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;
    }
}

```

The method “play” in the interface IPlay

```

package behaviours;
public interface IPlay {
    String play(String sound);
}

```

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

```

@Test
public void canPlay(){
    assertEquals( expected: "Piano can plink plonk", piano.play( sound: "plink plonk"));
}
}

PianoTest

>> 1 test passed - 2ms

/Library/Java/JavaVirtualMachines/jdk1.8.0_172.jdk/Contents/Home/bin/java ...

Process finished with exit code 0

```