Evidence for Implementation & Testing Unit

Name: Olga Maunsell
Cohort: E18

I.T. 1 - Encapsulation in a program

```
public class Guest {

    private String name;

    public Guest(String name){
        this.name = name;
    }

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

    public void setName(String name) {
        if(name != null && !name.isEmpty()) {
            this.name = name;
        }
    }
}
```

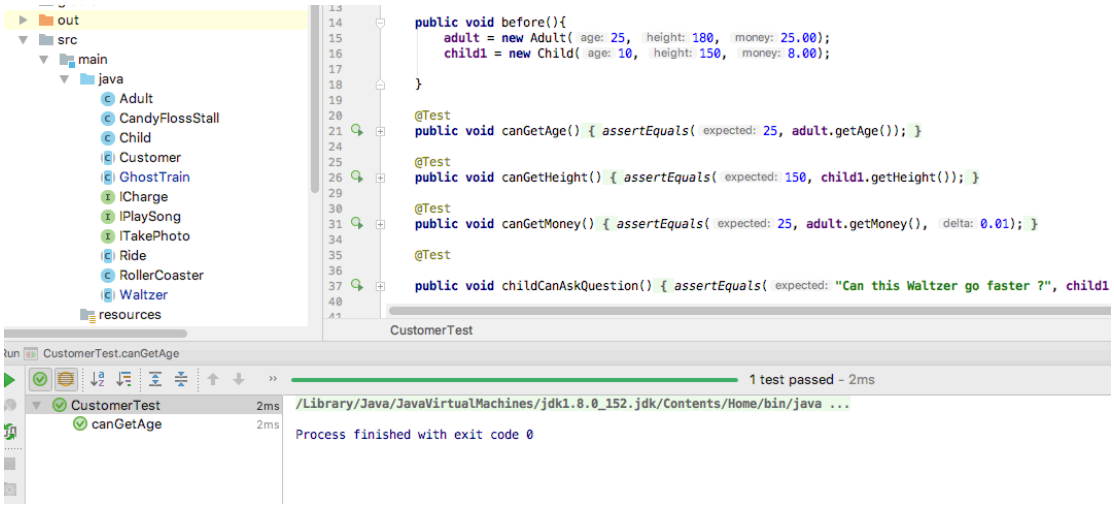
I.T.2 – Inheritance

```
public abstract class Customer {

    private int age;
    private int height;
    private double money;

    public Customer(int age, int height, double money){
        this.age = age;
        this.height = height;
        this.money = money;
    }

    public int getAge() { return this.age; }

    public int getHeight() { return this.height; }

    public double getMoney() { return this.money; }

    public String askQuestion(String ride) { return "Can this " + ride; }

}
```

Adult class inherits from Customer class

```
public class Adult extends Customer {

    public Adult(int age, int height, double money){
        super(age, height, money);

    }

    public String askQuestion(String ride) {

        return super.askQuestion(ride) + " go slower ?";
    }
}
```

Adult object and .getAge method inherited from Customer class



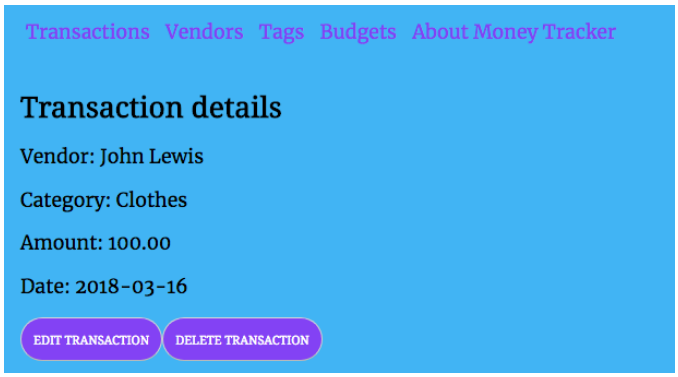
I.T.3 – Demonstrate searching data in a program

```
def self.find(id)
  sql = "SELECT * FROM transactions WHERE id = $1"
  values = [id]
  result = SqlRunner.run(sql, values)
  transaction = Transaction.new(result.first)
  return transaction
end
```

Result of search function running

Screenshot of all transactions

User selects to view 1 transaction and the above search function is then run to produce the result of transaction searched/ found and displayed



I.T.4 – Demonstrate sorting data in a program

```
def self.all()
  sql = "SELECT * FROM transactions ORDER BY transactions.transaction_date DESC"
  values = []
  result = SqlRunner.run(sql, values)
  transactions = Transaction.map_items(result)
  return transactions
end
```

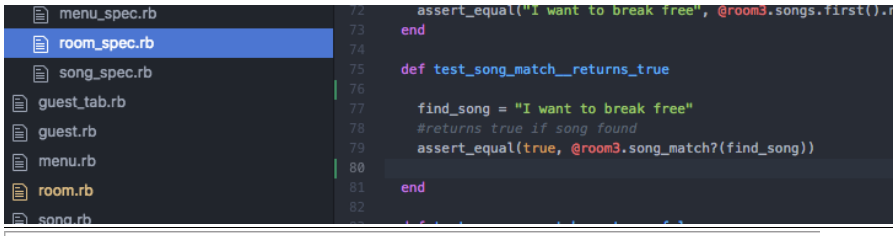
Transactions are retrieved and sorted in descending order to be displayed on the screen in descending order

I.T 5 – Use of an array in a program

```ruby
    #setup to test songs in room
    @song1 = Song.new("I want to break free")
    @song2 = Song.new("Park Life")
    @songs = [@song1, @song2]
    @room3 = Room.new(3, @no_guests, @songs, 3)
def test_song_match__returns_true
  find_song = "I want to break free"
  #returns true if song found
  assert_equal(true, @room3.song_match?(find_song))
end
```

```ruby
 def initialize(number, guests, songs, capacity)
   @number = number
   @guests = guests
   @songs = songs
   @capacity = capacity
   #price is per guest
   @price = 10
 end

def song_match?(fav_song)
  #Create array of song names
  song_names = @songs.map { |song| song.name }
  #check if favourite song is included in list of room songs

  if song_names.include?(fav_song)
    return true
  else
    return false
  end
end
```

Result of function running




```
➜  specs git:(master) × ruby room_spec.rb
Run options: --seed 57785

# Running:

.....................

Finished in 0.002761s, 7605.9399 runs/s, 8692.5027 assertions/s.
21 runs, 24 assertions, 0 failures, 0 errors, 0 skips
➜  specs git:(master) ×
```

I.T 6 – Use of a hash in a program

"admin" hash contained within "pet_shop" hash

```
        ],
        admin: {
          total_cash: 1000,
          pets_sold: 0,
        },
        name: "Camelot of Pets"
      }
end
```

```
  @pet_shop = {
      pets: [
        {
          name: "Sir Percy",
          pet_type: :cat,
          breed: "British Shorthair",
          price: 500
        },
        {
          name: "King Bagdemagus",
```


```
def total_cash(pet_shop)
  # Receives 1 parameter - pet shop hash
  # Returns total cash value from pet shop hash
  return pet_shop[:admin][:total_cash]
end
```

Result of function running

```
➜  weekend_homework git:(master) ✗ ruby specs/pet_shop_spec.rb
Run options: --seed 4644

# Running:

....................

Finished in 0.002450s, 8163.2653 runs/s, 10612.2449 assertions/s.
20 runs, 26 assertions, 0 failures, 0 errors, 0 skips
```

I.T. 7 – Polymorphism

1. **Shop class has an ArrayList of ISellable stockItems**

```
public class Shop {

    private String name;
    private ArrayList<ISellable> stockItems;
    private String category;
    public double till;

    public Shop(String name){
        this.name = name;
        this.stockItems = new ArrayList<>();
        this.till = 0.00;
    }

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

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

    public int getNoOfStockItems() {
        return this.stockItems.size();
    }

    public ArrayList getStockItems() {
        return this.stockItems;
    }

    public void addStockItem(StockItem item) {
        this.stockItems.add(item);
    }
```

---

2. StockItem class implements ISellable interface

```
public abstract class StockItem implements ISellable{

    private String stockItemType;
    private String description;
    private double wholesalePrice;
    private double retailPrice;

    public StockItem(String stockItemType, String description, double wholesalePrice){
        this.stockItemType = stockItemType;
        this.description = description;
        this.wholesalePrice = wholesalePrice;
//  Initialise retailPrice to wholesalePrice until markup takes place
        this.retailPrice = wholesalePrice;
    }

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

    public double getWholesalePrice() {
        return this.wholesalePrice;
    }
```

3. DrumSticks class extends StockItem class and therefore implements ISellable interface

```
public class DrumSticks extends StockItem {

    private String tip;

    public DrumSticks(String stockItemType, String description,double wholesalePrice, String tip){
        super(stockItemType, description, wholesalePrice);
        this.tip = tip;
    }

    public String getTip() { return this.tip; }

    public String sell(){
        String retailPriceString = String.format("%.2f", getRetailPrice());
        return getDescription() + " selling at £" + retailPriceString;
    }
}
```

4. Instrument class extends StockItem class and therefore implements ISellable interface.

```
public abstract class Instrument extends StockItem implements IPlayable {

    private InstrumentCategory category;
    private String colour;
    private String material;
//    private String brand;


    public Instrument(String stockItemType, String description, double wholesalePrice, Instrume
        super(stockItemType, description, wholesalePrice);
        this.category = category;
        this.colour = colour;
        this.material = material;
    }


    public InstrumentCategory getCategory() { return this.category; }

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

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


}
```

4. ISellable interface

```
public interface ISellable {

    String sell();

    double calculateMarkup(double increase);

    double getMarkupAmount();


}
```