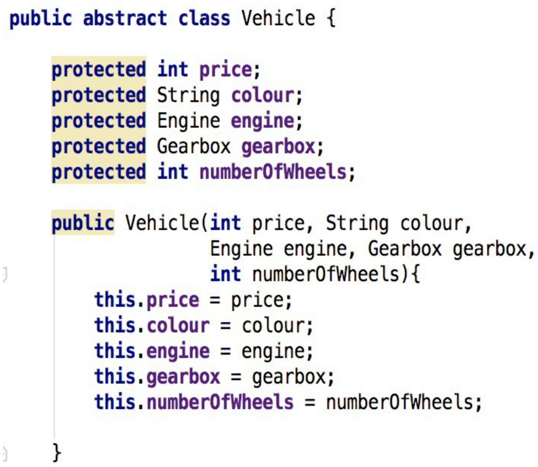
Evidence for Implementation and Testing Unit.

Hamish Hoad

Cohort E-21

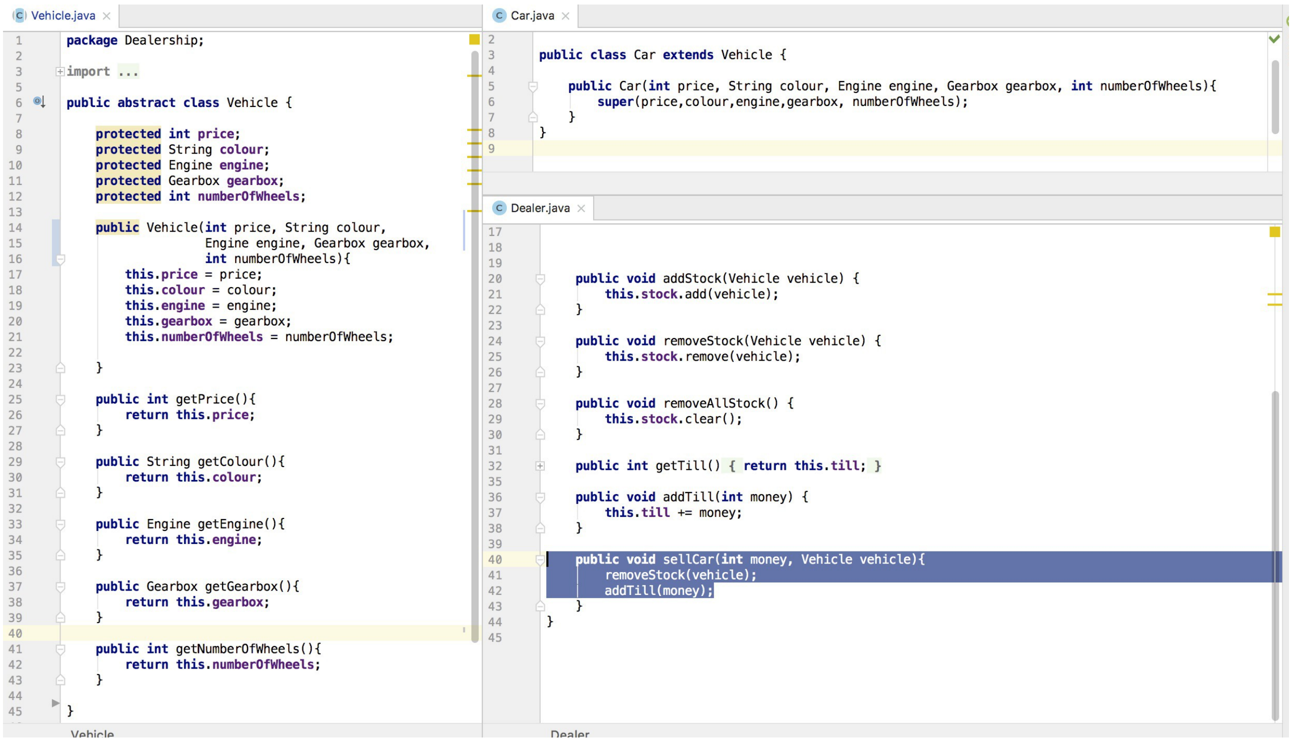
I.T 1- Demonstrate one example of encapsulation that you have written in a program.



```java
public abstract class Vehicle {

    protected int price;
    protected String colour;
    protected Engine engine;
    protected Gearbox gearbox;
    protected int numberOfWheels;

    public Vehicle(int price, String colour,
                   Engine engine, Gearbox gearbox,
                   int numberOfWheels){
        this.price = price;
        this.colour = colour;
        this.engine = engine;
        this.gearbox = gearbox;
        this.numberOfWheels = numberOfWheels;

    }
```

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

Example of ...
A Class.
A Class that inherits from another Class.
An object in the inherited Class.
A method (highlighted) that uses the information inherited.



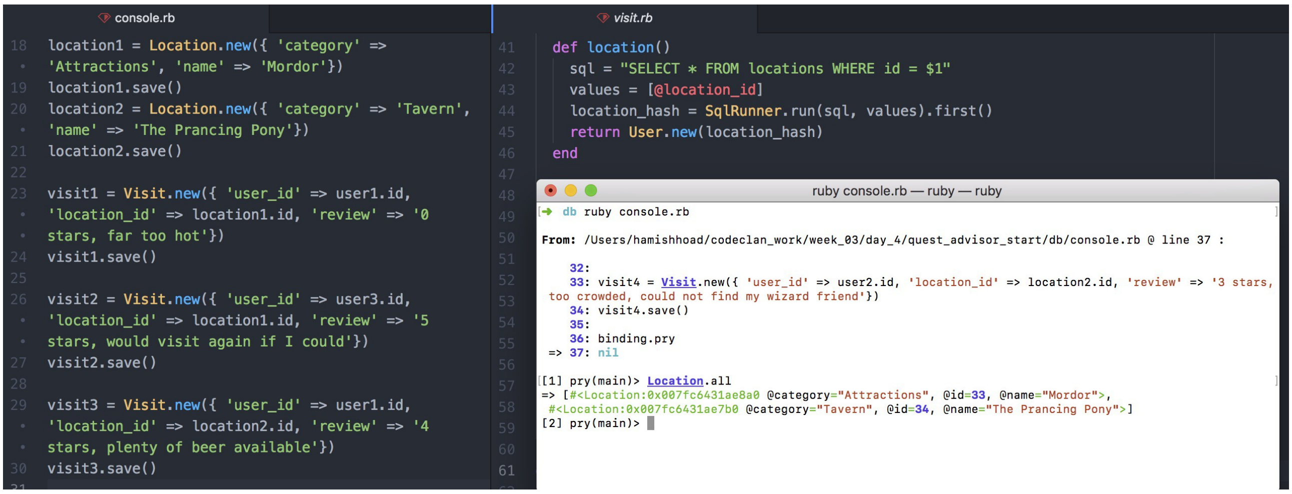
```java
package Dealership;

import ...

public abstract class Vehicle {

    protected int price;
    protected String colour;
    protected Engine engine;
    protected Gearbox gearbox;
    protected int numberOfWheels;

    public Vehicle(int price, String colour,
                   Engine engine, Gearbox gearbox,
                   int numberOfWheels){
        this.price = price;
        this.colour = colour;
        this.engine = engine;
        this.gearbox = gearbox;
        this.numberOfWheels = numberOfWheels;

    }

    public int getPrice(){
        return this.price;
    }

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

    public Engine getEngine(){
        return this.engine;
    }

    public Gearbox getGearbox(){
        return this.gearbox;
    }

    public int getNumberOfWheels(){
        return this.numberOfWheels;
    }

}
```

```java

public class Car extends Vehicle {

    public Car(int price, String colour, Engine engine, Gearbox gearbox, int numberOfWheels){
        super(price,colour,engine,gearbox, numberOfWheels);
    }
}

```

```java



    public void addStock(Vehicle vehicle) {
        this.stock.add(vehicle);
    }

    public void removeStock(Vehicle vehicle) {
        this.stock.remove(vehicle);
    }

    public void removeAllStock() {
        this.stock.clear();
    }

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

    public void addTill(int money) {
        this.till += money;
    }

    public void sellCar(int money, Vehicle vehicle){
        removeStock(vehicle);
        addTill(money);
    }
}

```

I.T 3 - Example of searching

Example of a function searching data and the output result.



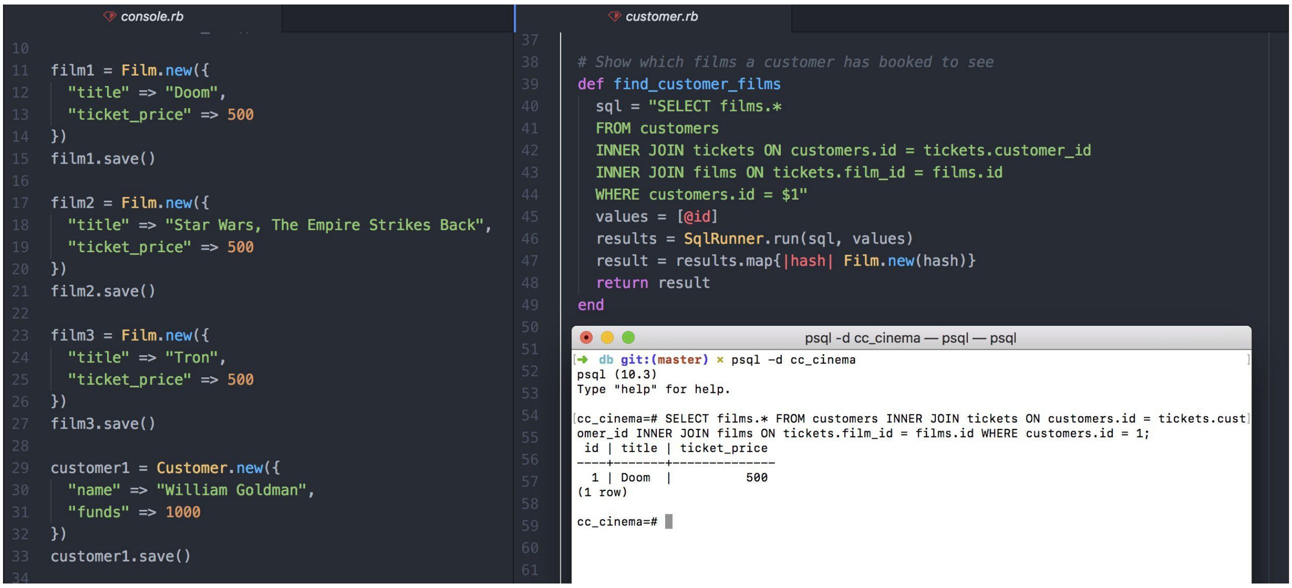
```
location1 = Location.new({ 'category' =>
'Attractions', 'name' => 'Mordor'})
location1.save()
location2 = Location.new({ 'category' => 'Tavern',
'name' => 'The Prancing Pony'})
location2.save()

visit1 = Visit.new({ 'user_id' => user1.id,
'location_id' => location1.id, 'review' => '0
stars, far too hot'})
visit1.save()

visit2 = Visit.new({ 'user_id' => user3.id,
'location_id' => location1.id, 'review' => '5
stars, would visit again if I could'})
visit2.save()

visit3 = Visit.new({ 'user_id' => user1.id,
'location_id' => location2.id, 'review' => '4
stars, plenty of beer available'})
visit3.save()
```

```
def location()
  sql = "SELECT * FROM locations WHERE id = $1"
  values = [@location_id]
  location_hash = SqlRunner.run(sql, values).first()
  return User.new(location_hash)
end
```

```
➜  db ruby console.rb

From: /Users/hamishhood/codeclan_work/week_03/day_4/quest_advisor_start/db/console.rb @ line 37 :

    32:
    33: visit4 = Visit.new({ 'user_id' => user2.id, 'location_id' => location2.id, 'review' => '3 stars,
 too crowded, could not find my wizard friend'})
    34: visit4.save()
    35:
    36: binding.pry
 => 37: nil

[1] pry(main)> Location.all
=> [#<Location:0x007fc6431ae8a0 @category="Attractions", @id=33, @name="Mordor">,
 #<Location:0x007fc6431ae7b0 @category="Tavern", @id=34, @name="The Prancing Pony">]
[2] pry(main)>
```

I.T 4 – Example of sorting

Example of a function sorting data and outputting a table of results.



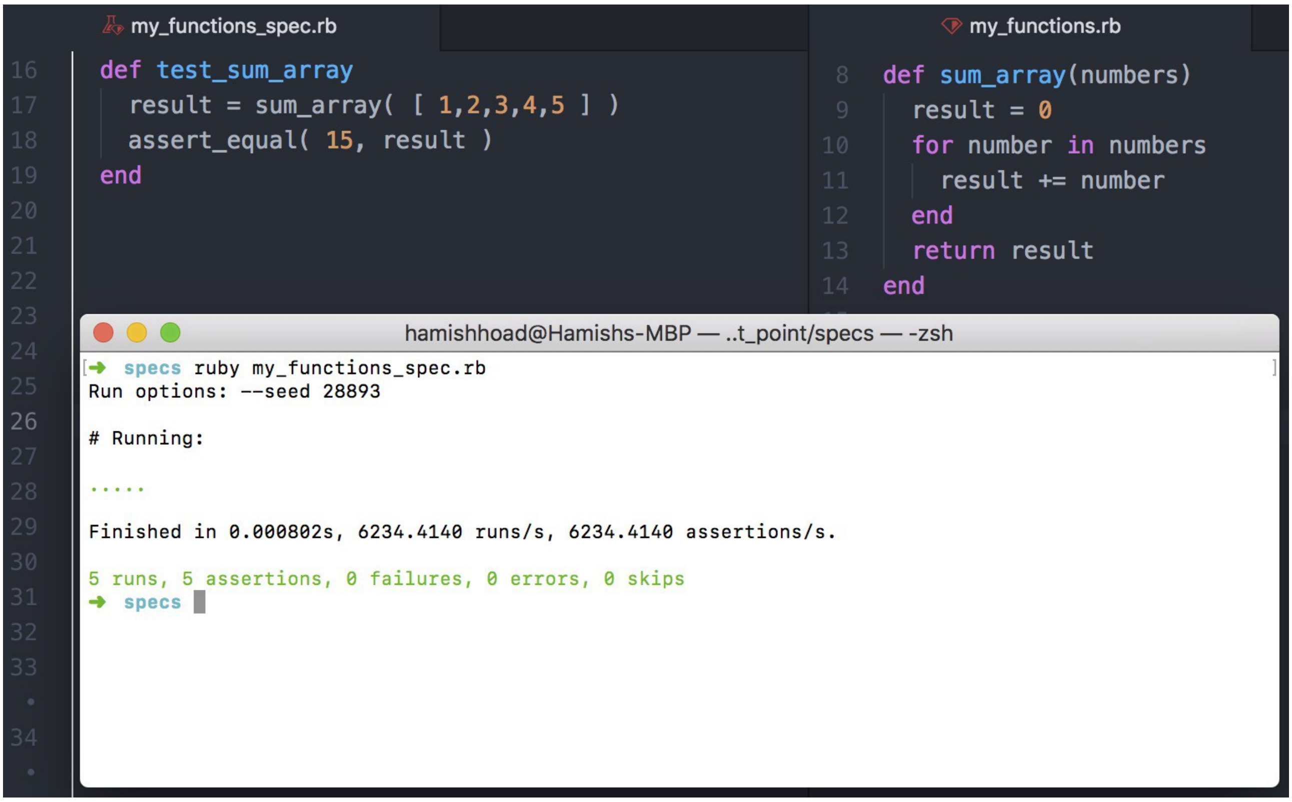
```
film1 = Film.new({
  "title" => "Doom",
  "ticket_price" => 500
})
film1.save()

film2 = Film.new({
  "title" => "Star Wars, The Empire Strikes Back",
  "ticket_price" => 500
})
film2.save()

film3 = Film.new({
  "title" => "Tron",
  "ticket_price" => 500
})
film3.save()

customer1 = Customer.new({
  "name" => "William Goldman",
  "funds" => 1000
})
customer1.save()
```

```
# Show which films a customer has booked to see
def find_customer_films
  sql = "SELECT films.*
  FROM customers
  INNER JOIN tickets ON customers.id = tickets.customer_id
  INNER JOIN films ON tickets.film_id = films.id
  WHERE customers.id = $1"
  values = [@id]
  results = SqlRunner.run(sql, values)
  result = results.map{|hash| Film.new(hash)}
  return result
end
```

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

cc_cinema=# SELECT films.* FROM customers INNER JOIN tickets ON customers.id = tickets.cust
omer_id INNER JOIN films ON tickets.film_id = films.id WHERE customers.id = 1;
 id | title | ticket_price
----+-------+--------------
  1 | Doom  |          500
(1 row)

cc_cinema=#
```

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

Example of a test array in a program, the function and the test results.



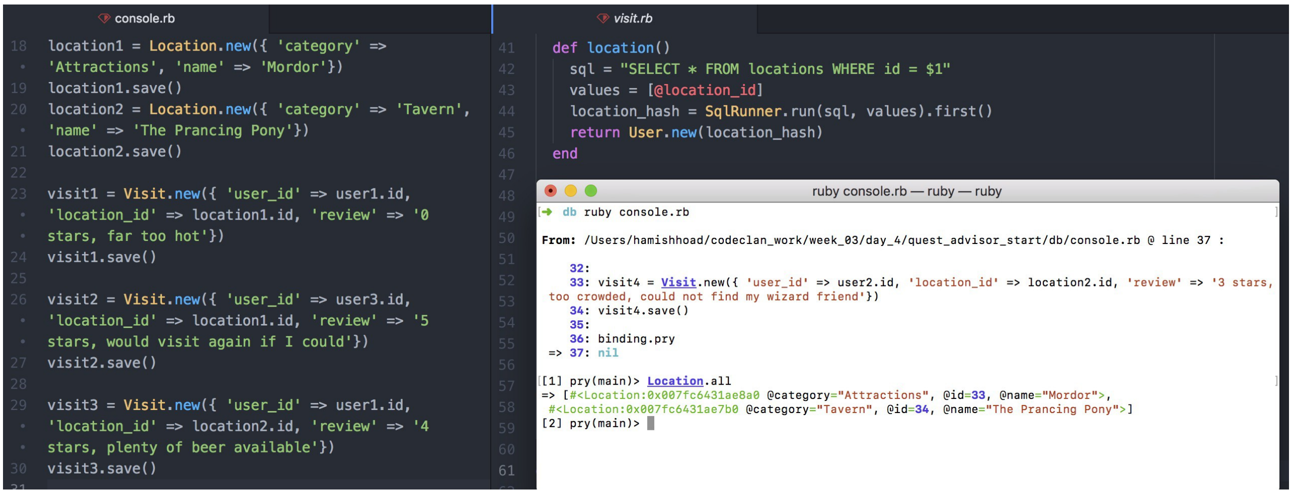
```ruby
def test_sum_array
  result = sum_array( [ 1,2,3,4,5 ] )
  assert_equal( 15, result )
end
```

```ruby
def sum_array(numbers)
  result = 0
  for number in numbers
    result += number
  end
  return result
end
```

```
➜  specs ruby my_functions_spec.rb
Run options: --seed 28893

# Running:

.....

Finished in 0.000802s, 6234.4140 runs/s, 6234.4140 assertions/s.

5 runs, 5 assertions, 0 failures, 0 errors, 0 skips
➜  specs
```

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

Example of a hash in a program, a function that calls the hash and the output result.



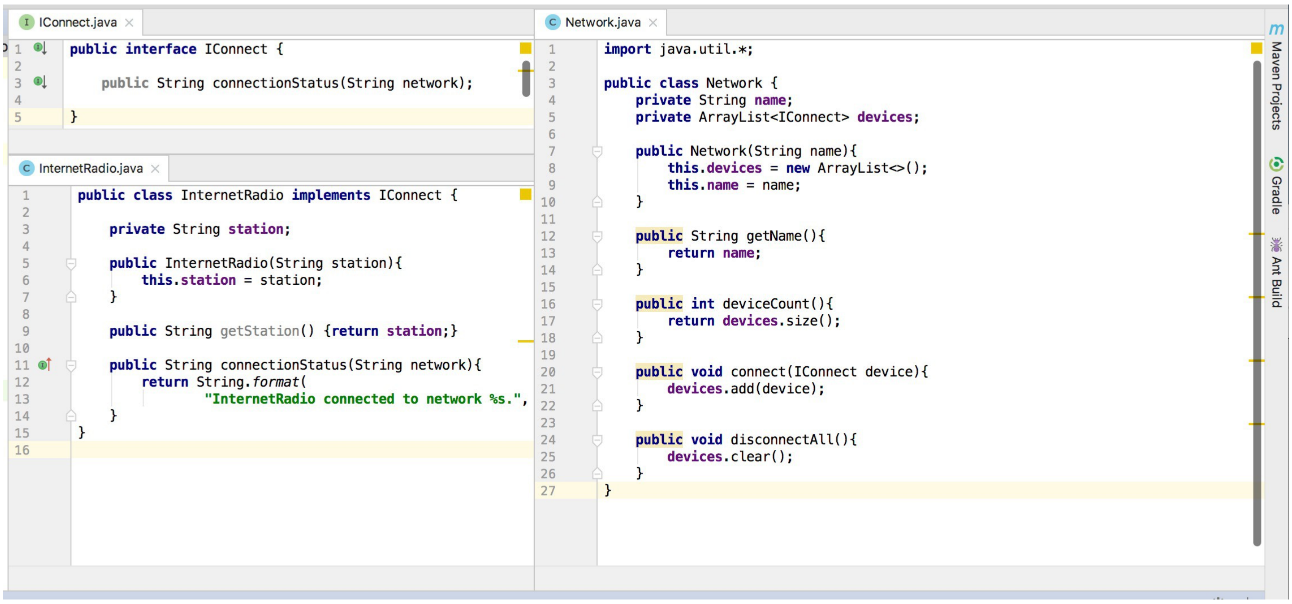
```ruby
location1 = Location.new({ 'category' =>
'Attractions', 'name' => 'Mordor'})
location1.save()
location2 = Location.new({ 'category' => 'Tavern',
'name' => 'The Prancing Pony'})
location2.save()

visit1 = Visit.new({ 'user_id' => user1.id,
'location_id' => location1.id, 'review' => '0
stars, far too hot'})
visit1.save()

visit2 = Visit.new({ 'user_id' => user3.id,
'location_id' => location1.id, 'review' => '5
stars, would visit again if I could'})
visit2.save()

visit3 = Visit.new({ 'user_id' => user1.id,
'location_id' => location2.id, 'review' => '4
stars, plenty of beer available'})
visit3.save()
```

```ruby
def location()
  sql = "SELECT * FROM locations WHERE id = $1"
  values = [@location_id]
  location_hash = SqlRunner.run(sql, values).first()
  return User.new(location_hash)
end
```

```
➜  db ruby console.rb

From: /Users/hamishhoad/codeclan_work/week_03/day_4/quest_advisor_start/db/console.rb @ line 37 :

    32:
    33: visit4 = Visit.new({ 'user_id' => user2.id, 'location_id' => location2.id, 'review' => '3 stars,
 too crowded, could not find my wizard friend'})
    34: visit4.save()
    35:
    36: binding.pry
 => 37: nil

[1] pry(main)> Location.all
=> [#<Location:0x007fc6431ae8a0 @category="Attractions", @id=33, @name="Mordor">,
 #<Location:0x007fc6431ae7b0 @category="Tavern", @id=34, @name="The Prancing Pony">]
[2] pry(main)>
```

I.T 7 - Example of polymorphism in a program

Example of a polymorphism in a program where a 'Radio' class implements an 'iConnect' interface in order to add an instance to a 'Network' and tune to a radio station.



```java
public interface IConnect {

    public String connectionStatus(String network);

}
```

```java
public class InternetRadio implements IConnect {

    private String station;

    public InternetRadio(String station){
        this.station = station;
    }

    public String getStation() {return station;}

    public String connectionStatus(String network){
        return String.format(
                "InternetRadio connected to network %s.",
    }
}
```

```java
import java.util.*;

public class Network {
    private String name;
    private ArrayList<IConnect> devices;

    public Network(String name){
        this.devices = new ArrayList<>();
        this.name = name;
    }

    public String getName(){
        return name;
    }

    public int deviceCount(){
        return devices.size();
    }

    public void connect(IConnect device){
        devices.add(device);
    }

    public void disconnectAll(){
        devices.clear();
    }
}
```