I.T. 1 - Encapsulation

```java
public abstract class MenuItem implements Orderable {

    private ArrayList<Ingredient> ingredients;
    private String name;
    private double price;

    public MenuItem(ArrayList<Ingredient> ingredients, String name, double price) {
        this.ingredients = ingredients;
        this.name = name;
        this.price = price;
    }
```

I.T. 2 - Inheritance

Class:

```java
public class Restaurant {

    private double budget;
    ArrayList<Table> tables;
    com.example.daniel.project2.Menu menu;
    private Kitchen kitchen;

    public Restaurant(ArrayList<Table> tables, com.example.daniel.project2.Menu menu, Kitchen kitchen) {
        this.budget = 0;
        this.tables = tables;
        this.menu = menu;
        this.kitchen = kitchen;
    }

    public double getBudget() {
        return budget;
    }


    public int countTables() {
        return tables.size();
```

Class that inherits:

```java
public class Food extends MenuItem {
    private String course;

    public Food(String course, ArrayList<Ingredient> ingredients, String name, double price) {
        super(ingredients, name, price);
        this.course = course;
    }

    public String getCourse() {
        return course;
    }
}
```

Object in inherited class:

```java
public class DrinkTest {

    Drink drink1;
    Ingredient vodka, lemon, pruneJuice;

    @Before
    public void before() {

        vodka = new Ingredient("vodka", 5);
        lemon = new Ingredient("lemon", 5);
        pruneJuice = new Ingredient("prune juice", 5);

        ArrayList<Ingredient> vodkaPrune = new ArrayList<>();
        vodkaPrune.add(vodka);
        vodkaPrune.add(lemon);
        vodkaPrune.add(pruneJuice);
        drink1 = new Drink(vodkaPrune, "prune cocktail", 7.50, "pint");
    }
```

Method using inherited information:

```java
    public void finalisePatronOrder(Patron patron, Orderable order) {
        ArrayList<Ingredient> orderIngredients = order.getIngredients();

        for(Ingredient eachIngredient : orderIngredients) {
            this.kitchen.reduceAmountByOne(eachIngredient);

        }
        patron.addToOrder(order);
    }
```

I.T. 3 - Demonstration of Searching

```
[➜ PDA ruby pda.rb
3
```

pda.rb

```ruby
array1 = [1, 2, 3]
hash1 = {a: 1, b: 2, c: 3}

def search_for_mulipliers_of_3(array)
  result = array.find {|num| num % 3 == 0}
  puts result
end

puts search_for_mulipliers_of_3(array1)

```

I.T. 4 - Demonstrate Sorting Data

```
[➜ PDA ruby pda.rb
3
2
1
```

pda.rb

```ruby
array1 = [1, 2, 3]
hash1 = {a: 1, b: 2, c: 3}

def sort_desc(array)
  result = array.sort.reverse
  return result
end

puts sort_desc(array1)

```

I.T. 5 - Use of Array

```
[➜  PDA ruby pda.rb
3
```

pda.rb

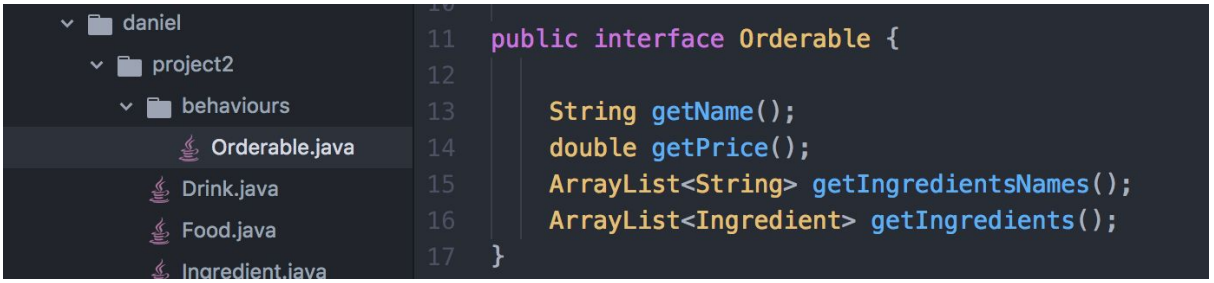
```ruby
array1 = [1, 2, 3]

def length(array)
  puts array.length()
end

puts length(array1)

```

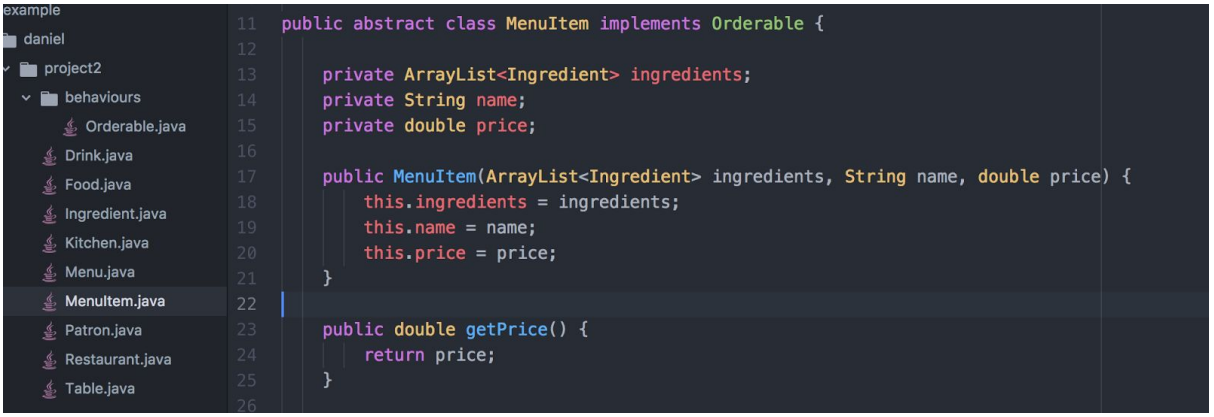
I.T. 6 - Use of Hash

```
[➜  PDA ruby pda.rb
{:a=>1, :b=>2, :c=>3}
```

```ruby
hash1 = {a: 1, b: 2, c: 3}

def show(hash)
  puts hash
end

puts show(hash1)

```

I.T. 7 - Polymorphism




example
daniel
project2
behaviours
Orderable.java
Drink.java
Food.java
Ingredient.java
Kitchen.java
Menu.java
MenuItem.java
Patron.java
Restaurant.java
Table.java
public abstract class MenuItem implements Orderable {
private ArrayList<Ingredient> ingredients;
private String name;
private double price;
public MenuItem(ArrayList<Ingredient> ingredients, String name, double price) {
this.ingredients = ingredients;
this.name = name;
this.price = price;
}
public double getPrice() {
return price;
}