

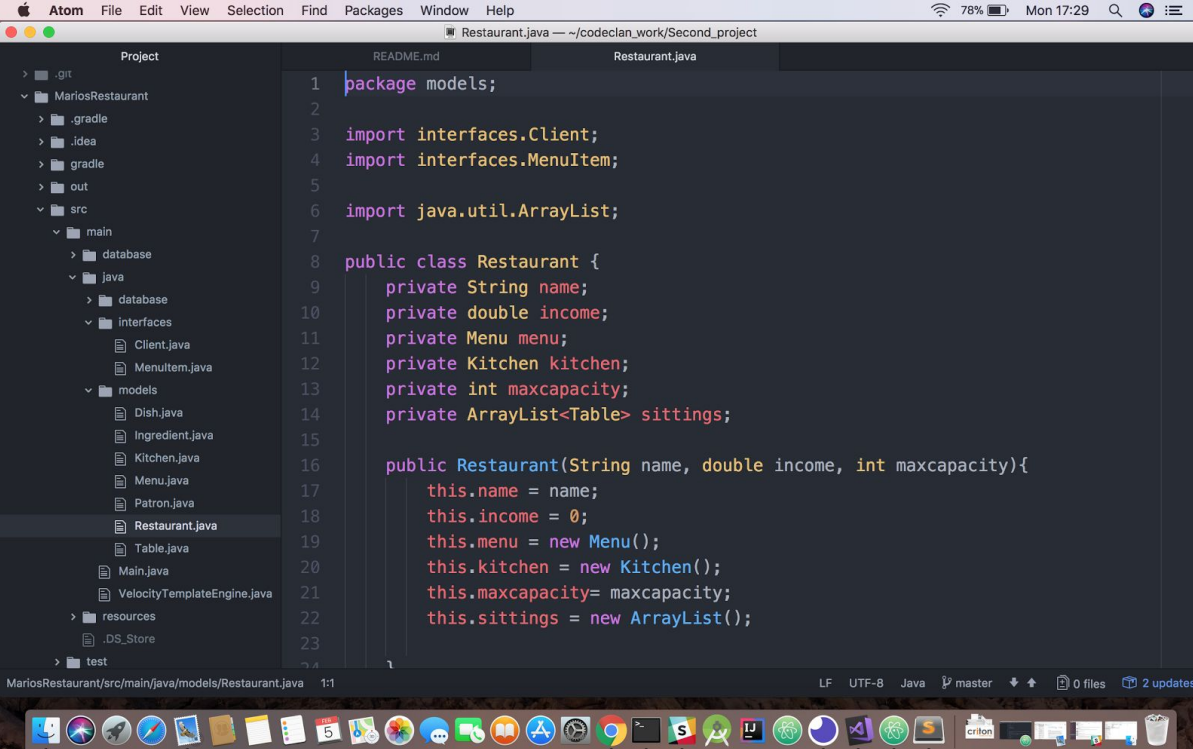
Evidence for Implementation and Testing Unit.

Your name here: Joao Nequinha

Your Cohort: E16

Date here: 05.02.2018

I.T 1- Example of encapsulation in a program:



The screenshot shows the Atom code editor with a project named 'MariosRestaurant'. The file explorer on the left shows the project structure, including a 'models' package containing 'Restaurant.java'. The main editor window displays the code for 'Restaurant.java', which is a Java class implementing encapsulation. The code includes imports for 'Client', 'MenuItem', and 'ArrayList' from the 'models' package and 'java.util'. The 'Restaurant' class has private attributes: 'name' (String), 'income' (double), 'menu' (Menu), 'kitchen' (Kitchen), 'maxcapacity' (int), and 'sittings' (ArrayList<Table>). The constructor 'Restaurant(String name, double income, int maxcapacity)' initializes these attributes. The status bar at the bottom indicates the file is 'Restaurant.java' in the 'models' package, with 1 line of code, UTF-8 encoding, and 2 updates.

```
1 package models;
2
3 import interfaces.Client;
4 import interfaces.MenuItem;
5
6 import java.util.ArrayList;
7
8 public class Restaurant {
9     private String name;
10    private double income;
11    private Menu menu;
12    private Kitchen kitchen;
13    private int maxcapacity;
14    private ArrayList<Table> sittings;
15
16    public Restaurant(String name, double income, int maxcapacity){
17        this.name = name;
18        this.income = 0;
19        this.menu = new Menu();
20        this.kitchen = new Kitchen();
21        this.maxcapacity= maxcapacity;
22        this.sittings = new ArrayList();
23    }
24 }
```

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

A class:

```
package models;

import interfaces.MenuItem;
import java.util.ArrayList;

public class Dish implements MenuItem {
    private String name;
    private double price;
    private ArrayList<Ingredient> ingredients;

    public Dish(String name, double price){
        this.name= name;
        this.price = price;
        this.ingredients = new ArrayList();
    }

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

A Class that inherits from the previous class:

```
public class Salad extends Dish{
    private String name;
    private double price;
    private ArrayList<Ingredient> ingredients;

    public Salad(String name, double price){
        this.name= name;
        this.price = price;
        this.ingredients = new ArrayList();
    }
}
```

An Object in the inherited class:

```
public class Dish implements MenuItem {
    private String name;
    private double price;
    private ArrayList<Ingredient> ingredients;

    public Dish(String name, double price){
        this.name= name;
        this.price = price;
        this.ingredients = new ArrayList();
    }

    public String getName() {
```

A method that that uses information inherited from another class:

```
public void reduceAmountOfIngredients() {  
    for(Ingredient ingredient: ingredients){  
        int amount = ingredient.getAmount();  
        ingredient.setAmount(amount -= 1);  
    }  
}
```

I.T 3 - Demonstrate searching data in a program:

Function that searches data:

```
def self.find(id)  
    sql = " SELECT * FROM merchants WHERE id=$1"  
    values = [id]  
    merchant_found = SqlRunner.run(sql,values)  
    return Merchant.new(merchant_found.first)  
end
```

Result of function running:

From: /Users/user/codeclan_work/first_project/db/seeds.rb @ line 50 :

```
45:  
46:  
47: binding.pry  
48:  
49:  
=> 50: nil
```

```
[1] pry(main)> Merchants.all()  
NameError: uninitialized constant Merchants  
Did you mean?  Merchant  
from (pry):1:in `<main>'  
[2] pry(main)> Merchant.all()  
=> [#<Merchant:0x007f9293b97378 @id=10, @name="Zara">,  
    #<Merchant:0x007f9293b96b80 @id=11, @name="Tesco">,  
    #<Merchant:0x007f9293b965b8 @id=12, @name="Morrison">,  
    #<Merchant:0x007f9293b960b8 @id=13, @name="Fusion">]  
[3] pry(main)> Merchant.find(12)  
=> #<Merchant:0x007f9293ab5ef0 @id=12, @name="Morrison">  
[4] pry(main)> █
```

I.T 4 - Demonstrate sorting data in a program:

Function that sorts data:

```
class ThingToDo
  attr_accessor(:act1, :act2, :act3, :act4);

  def initialize(act1, act2, act3, act4)
    @things = [act1, act2, act3, act4]
  end

  def sort_alpha()
    list = @things
    list.sort! {|a,b| a<=>b}
  end
end

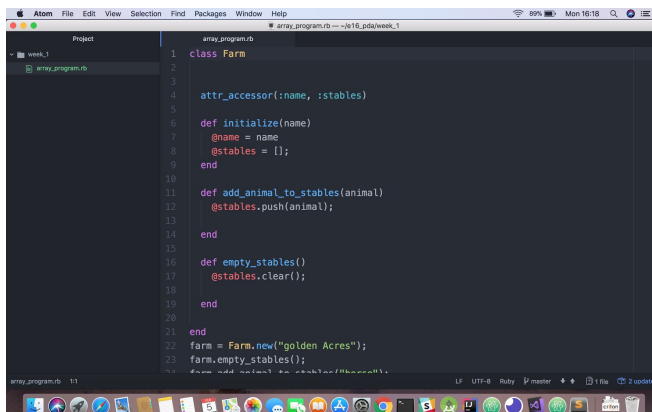
my_things = ThingToDo.new("Pda", "Interviews", "Dinner", "Stressout");

print my_things.sort_alpha();
```

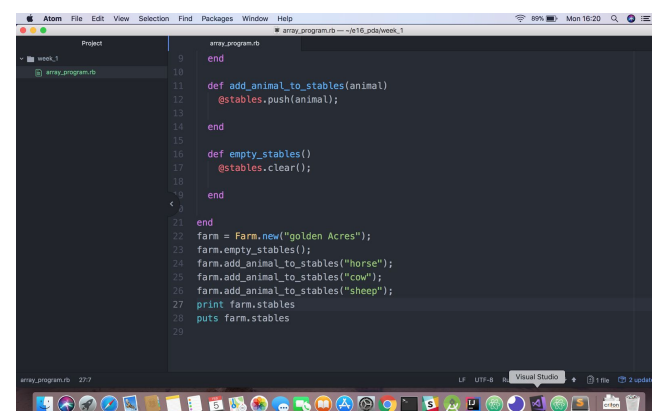
Result:

```
Last login: Thu Feb  8 16:31:26 on ttys007
[→ e16_pda git:(master) ✕ ruby Sort_ruby.rb
["Dinner", "Interviews", "Pda", "Stressout"]
→ e16_pda git:(master) ✕
```

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



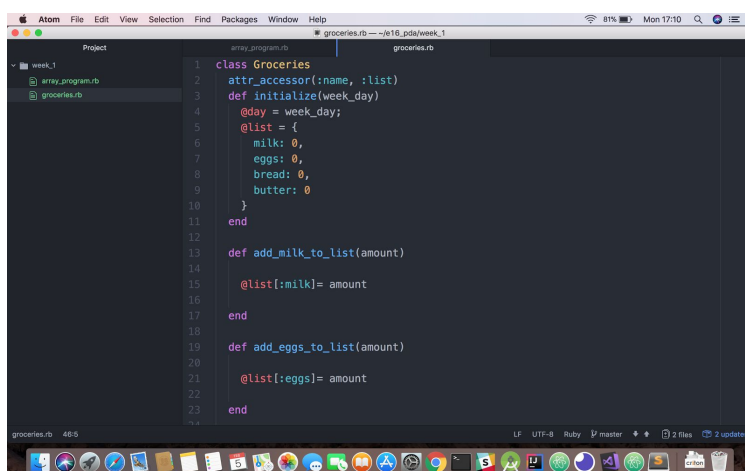
```
1 class Farm
2
3
4   attr_accessor(:name, :stables)
5
6   def initialize(name)
7     @name = name
8     @stables = [];
9   end
10
11   def add_animal_to_stables(animal)
12     @stables.push(animal);
13   end
14
15   def empty_stables()
16     @stables.clear();
17   end
18 end
19
20
21 end
22 farm = Farm.new("golden Acres");
23 farm.empty_stables();
24 farm.add_animal_to_stables("horse");
25 farm.add_animal_to_stables("cow");
26 farm.add_animal_to_stables("sheep");
27 print farm.stables
28 puts farm.stables
29
```



```
9
10
11   def add_animal_to_stables(animal)
12     @stables.push(animal);
13   end
14
15   def empty_stables()
16     @stables.clear();
17   end
18 end
19
20
21 end
22 farm = Farm.new("golden Acres");
23 farm.empty_stables();
24 farm.add_animal_to_stables("horse");
25 farm.add_animal_to_stables("cow");
26 farm.add_animal_to_stables("sheep");
27 print farm.stables
28 puts farm.stables
29
```

```
node - ng TMPDIR=/var/folders/ ..16_pda/week_1 ..16_pda/week_1 ..first_project +
Last login: Mon Feb 5 16:16:56 on ttys003
➔ week_1 git:(master) * ruby array_program.rb
["horse", "cow", "sheep"]horse
cow
sheep
➔ week_1 git:(master) * █
```

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



```
1 class Groceries
2   attr_accessor(:name, :list)
3   def initialize(week_day)
4     @day = week_day;
5     @list = {
6       milk: 0,
7       eggs: 0,
8       bread: 0,
9       butter: 0
10    }
11  end
12
13  def add_milk_to_list(amount)
14
15    @list[:milk]= amount
16
17  end
18
19  def add_eggs_to_list(amount)
20
21    @list[:eggs]= amount
22
23  end
24 end
```

```
Atom  File  Edit  View  Selection  Find  Packages  Window  Help
groceries.rb -- ..16_pda/week_1

Project
- week_1
  groceries.rb
  groceries.rb

26
27 @list[:bread]= amount
28
29 end
30 def add_butter_to_list(amount)
31
32   @list[:butter]= amount;
33
34 end
35
36 end
37
38
39 saturday = Groceries.new("saturday");
40
41 saturday.add_milk_to_list(1);
42 saturday.add_eggs_to_list(6);
43 saturday.add_bread_to_list(1);
44
45 print saturday.list;
46 puts saturday.list;
47
```

```
node - ng TMPDIR=/var/fti  X  ..16_pda/week_1  ..16_pda/week_1  ..week_01/day_3  ..first_project  +
Last login: Mon Feb 5 17:10:18 on ttys004
➔ week_1 git:(master) ✖ ruby groceries.rb
{:milk=>1, :eggs=>6, :bread=>1, :butter=>0}{:milk=>1, :eggs=>6, :bread=>1, :butter=>0}
➔ week_1 git:(master) ✖
```

I.T 7 - Demonstrate the use of polymorphism in a program

```
package models;

import ...

public class Menu {
    private ArrayList<MenuItem> menu;

    public Menu(){
        this.menu= new ArrayList();
    }

    public int dishCount() { return this.menu.size(); }

    public void addDish(MenuItem dish) { this.menu.add(dish); }

    public boolean menuHas(MenuItem dish) { return this.menu.contains(dish); }

    public void removeItem(MenuItem dish) { this.menu.remove(dish); }
}
```

```
import ...

public class Patron implements Client {
    private double bill;
    private ArrayList<MenuItem> order;

    public Patron(){
        this.bill = 0;
        this.order = new ArrayList();
    }

    public double getBill() { return this.bill; }
}
```

```
import models.Ingredient;

public interface MenuItem {

    double getPrice();
    void removeIngredient(Ingredient ingredient);
    void reduceAmountOfIngredients();
    boolean checkIfPossibleToMakeItem();
}
```

```
//creating a new observation List for Payload3 to prevent conflicts with tes
var observationsList = new List<IDevicePatientTestObservation>();
var oruObservationsList = oru.OBXs;
var fieldIndex = 0;

foreach (HL7NotedSegment obs in oruObservationsList)
{
    var ob = new Observation();
    ob.Analyte = oru.OBXs[fieldIndex][3];
    ob.Value = oru.OBXs[fieldIndex][5];
    ob.Units = oru.OBXs[fieldIndex][6];
    ob.Abnormal = oru.OBXs[fieldIndex][8] == "A";

    observationsList.Add(ob);

    fieldIndex++;
}

Observations = observationsList;
}
```