

Evidence for Implementation and Testing Unit.

Raul Ruiz

Cohort 15

17/09/17

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

```
public class VideoFile extends MediaFile {  
  
    private VideoCodec videoCodec;  
    private VideoType videoType;  
  
    public VideoFile (double lenght, String title, String nationality, String  
author, VideoCodec videoCodec, VideoType videoType) {  
        super(lenght, title, nationality, author);  
        this.videoType = videoType;  
        this.videoCodec = videoCodec;  
    }  
  
    public VideoCodec getVideoCodec(){  
        return this.videoCodec;  
    }  
  
    public VideoType getVideoType(){  
        return this.videoType;  
    }  
}
```

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

```
public class MediaFile{

    double length;
    String title;
    String nationality;
    String author;

    public MediaFile(double length, String title, String nationality, String
author){
        this.length = length;
        this.title = title;
        this.nationality = nationality;
        this.author = author;
    }

    public double getLenght(){
        return this.length;
    }

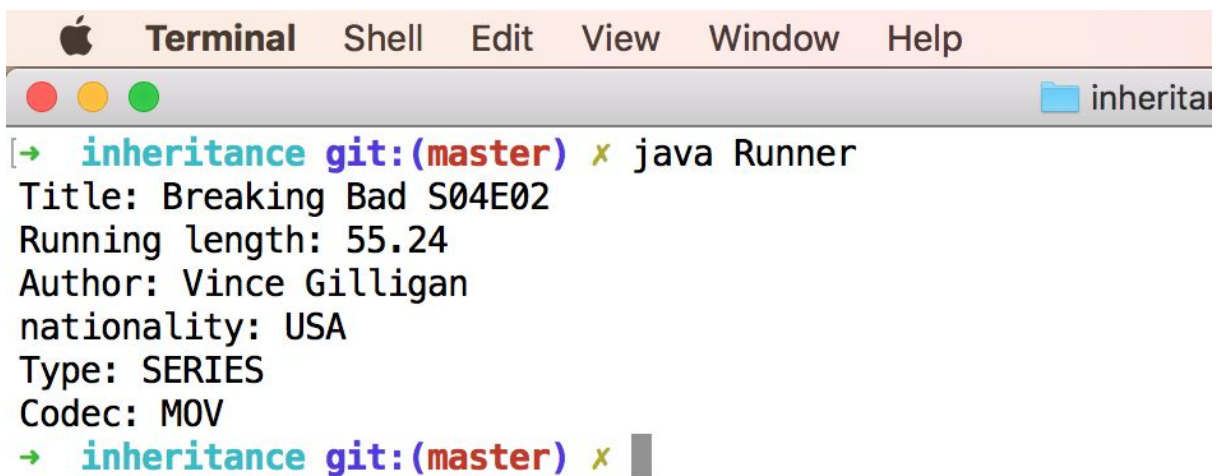
    public String getTitle(){
        return this.title;
    }

    public String getNationality(){
        return this.nationality;
    }

    public String getAuthor(){
        return this.author;
    }
}
```

```
1 public class VideoFile extends MediaFile {
2
3     private VideoCodec videoCodec;
4     private VideoType videoType;
5
6     public VideoFile (double lenght, String title, String nationality,
7     • String author, VideoCodec videoCodec, VideoType videoType) {
8         super(lenght, title, nationality, author);
9         this.videoType = videoType;
10        this.videoCodec = videoCodec;
11    }
12
13    public VideoCodec getVideoCodec(){
14        return this.videoCodec;
15    }
16
17    public VideoType getVideoType(){
18        return this.videoType;
19    }
20
21    public void displayVideoInfo(){
22        System.out.println("Title: " + super.getTitle());
23        System.out.println("Running length: " + super.getLenght());
24        System.out.println("Author: " + super.getAuthor());
25        System.out.println("nationality: " + super.getNationality());
26        System.out.println("Type: " + this.getVideoType().toString());
27        System.out.println("Codec: " + this.getVideoCodec().toString());
28    }
29
29
```

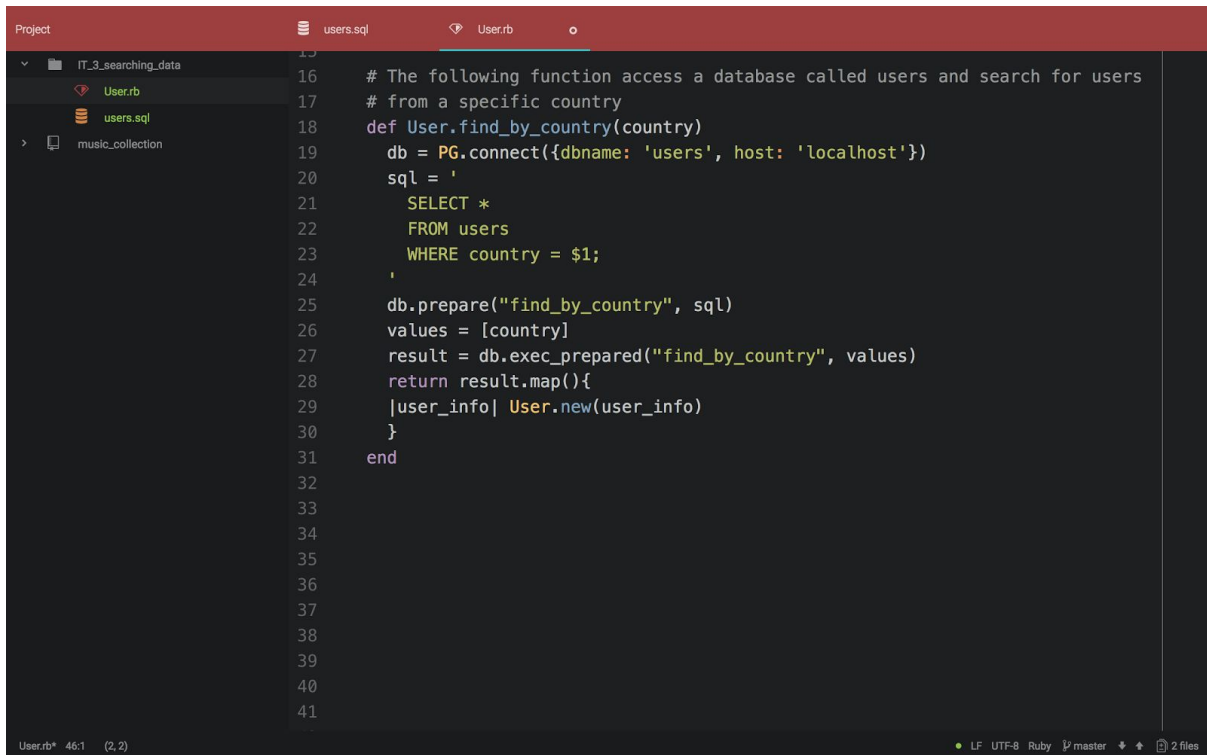
```
public class Runner{  
  
    public static void main(String[] args){  
  
        // Object from the subclass //  
        VideoFile videoFile = new VideoFile(55.24, "Breaking Bad S04E02", "USA",  
        "Vince Gilligan", VideoCodec.MOV, VideoType.SERIES);  
  
        // Method that uses information inherited from another class //  
        videoFile.displayVideoInfo();  
    }  
}
```



A screenshot of a macOS Terminal window. The title bar shows the Apple logo, the word "Terminal", and menu items "Shell", "Edit", "View", "Window", and "Help". The window has three colored window control buttons (red, yellow, green) on the left and a folder icon labeled "inheritance" on the right. The terminal content shows the command `java Runner` being executed, followed by its output: "Title: Breaking Bad S04E02", "Running length: 55.24", "Author: Vince Gilligan", "nationality: USA", "Type: SERIES", and "Codec: MOV". The prompt `inheritance git:(master) x` is visible at the bottom.

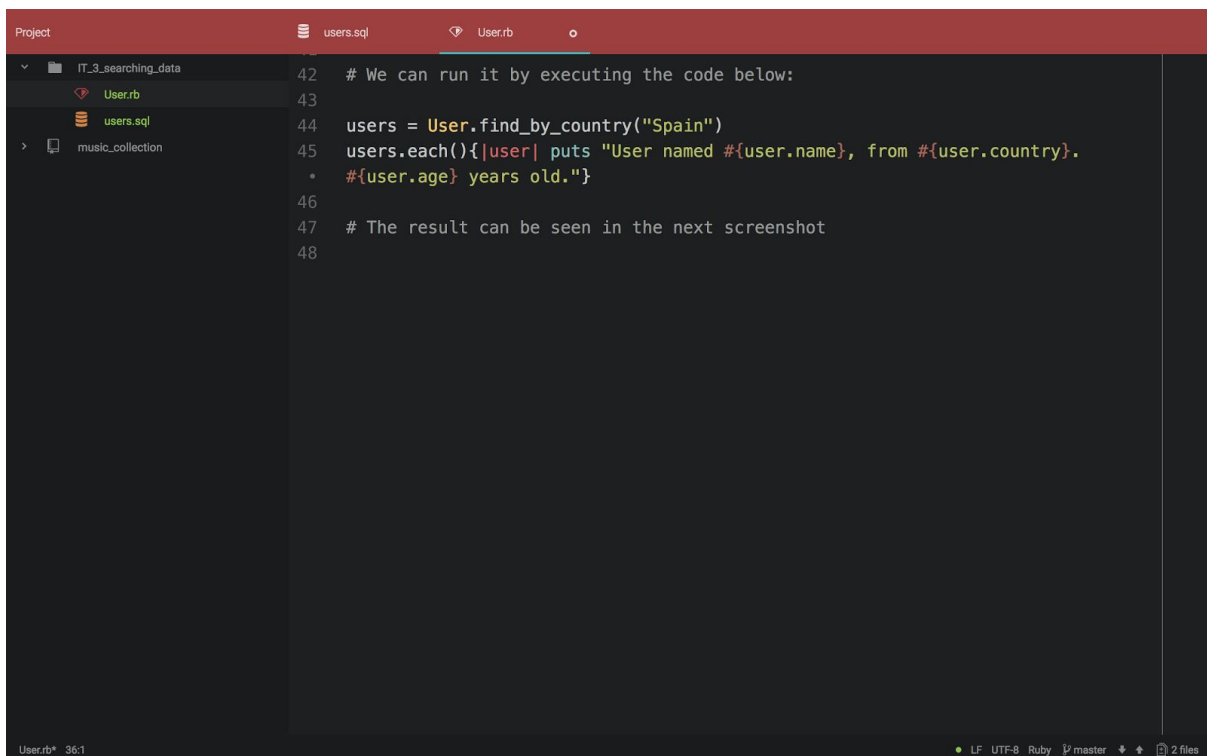
```
Terminal  Shell  Edit  View  Window  Help  
[→ inheritance git:(master) x java Runner  
Title: Breaking Bad S04E02  
Running length: 55.24  
Author: Vince Gilligan  
nationality: USA  
Type: SERIES  
Codec: MOV  
→ inheritance git:(master) x
```

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



The screenshot shows a code editor with a project named 'IT_3_searching_data'. The file explorer on the left shows 'User.rb' and 'users.sql'. The main editor window displays the 'User.rb' file, which contains a function 'find_by_country' that connects to a PostgreSQL database, executes a SQL query to find users by country, and returns the results as a list of User objects.

```
16 # The following function access a database called users and search for users
17 # from a specific country
18 def User.find_by_country(country)
19   db = PG.connect({dbname: 'users', host: 'localhost'})
20   sql = '
21     SELECT *
22     FROM users
23     WHERE country = $1;
24   '
25   db.prepare("find_by_country", sql)
26   values = [country]
27   result = db.exec_prepared("find_by_country", values)
28   return result.map(){
29     |user_info| User.new(user_info)
30   }
31 end
32
33
34
35
36
37
38
39
40
41
```



The screenshot shows the same code editor with the 'User.rb' file. The code now includes a call to the 'find_by_country' function with 'Spain' as the argument, and a loop that prints the details of each user found. Comments indicate that the code can be run by executing the code below and that the result can be seen in the next screenshot.

```
42 # We can run it by executing the code below:
43
44 users = User.find_by_country("Spain")
45 users.each(){|user| puts "User named #{user.name}, from #{user.country}.
46   *   #{user.age} years old."}
47
48 # The result can be seen in the next screenshot
49
```

```
psql | ..earching_data | +
→ IT_3_searching_data git:(master) ✕ ruby User.rb
User named Raul, from Spain. 29 years old.
User named Alvaro, from Spain. 24 years old.
→ IT_3_searching_data git:(master) ✕ █
```

I.T 4 - Example of sorting data in a program.

```
Project      users.sql      User.rb
IT_3_searching_data
  User.rb
  users.sql
  music_collection

32
33 # The following function returns all the elemtns within the database
34 # sorted by age in ascending order
35
36 def User.sort_users_by_age()
37   db = PG.connect({dbname: 'users', host: 'localhost'})
38   sql = '
39     SELECT *
40     FROM users
41   '
42   result = db.exec(sql)
43   users = result.map(){
44     |user_info| User.new(user_info)
45   }
46   sorted_users = users.sort!() {
47     |user_a, user_b| user_a.age <=> user_b.age
48   }
49   return sorted_users
50 end
51
```

```
Project      users.sql      User.rb
IT_3_searching_data
  User.rb
  users.sql
  music_collection

62 # We will demonstrate the results by calling the function through
63 # the code below:
64
65 result = User.sort_users_by_age
66
67 result.each(){
68   |user|
69   puts "User name:#{user.name}, #{user.age} years old."
70 }
71
```

```
psql                                     ..earching_data +
→ IT_3_searching_data git:(master) x ruby User.rb
User name:Vladimir, 18 years old.
User name:Evans, 19 years old.
User name:Clark, 20 years old.
User name:Trott, 24 years old.
User name:Alvaro, 24 years old.
User name:Hills, 25 years old.
User name:Ghosh, 27 years old.
User name:Jones, 29 years old.
User name:Raul, 29 years old.
User name:Reily, 29 years old.
User name:Usman, 29 years old.
User name:Baker, 30 years old.
User name:Frank, 30 years old.
User name:Irwin, 36 years old.
User name:Klein, 40 years old.
User name:Zafar, 42 years old.
User name:Valdo, 50 years old.
→ IT_3_searching_data git:(master) x █
```


I.T 5 - Example of the use of an array in a program.

```
IT5_arrays.rb
1
2
3 # We want to hold an array of people who are queuing in a barbershop
4 # waiting for their turn to get a hair cut
5
6 barber_queue = ["Mark", "Benjamin", "Harry"]
7
```

```
IT5_arrays.rb
9 # Whenever a new customer joins the queue they get pushed into the array.
10 # Whenever the barber calls the following customer on the queue, the first
11 # element from the array will be returned and removed from the array.
12
13 def add_new_customer_to_queue(customer, queue)
14     result = customer.push(customer)
15     return result
16 end
17
18 def call_next_customer_from_queue(queue)
19     next_customer = queue.shift()
20     return next_customer
21 end
22
```

```
IT5_arrays.rb
22
23 puts "The current state of the barber queue is #{barber_queue}"
24 puts "We will now call the next customer in the queue..."
25
26 next_customer = call_next_customer_from_queue(barber_queue)
27
28 puts "The next customer is #{next_customer}"
29 puts "The current state of the barber queue is #{barber_queue}"
30 puts
31
32 puts "A new customer arrived to the barbar shop and we will add it to the queue"
33 puts "Adding a new customer 'Michael' to the barber queue..."
34
35 add_new_customer_to_queue("Michael", barber_queue)
36
37 puts "The current state of the barber queue is #{barber_queue}"
38
```

```
[→ PDA_evidences git:(master) ✕ ruby IT5_arrays.rb
The current state of the barber queue is ["Mark", "Benjamin", "Harry"]
We will now call the next customer in the queue...
The next customer is Mark
The current state of the barber queue is ["Benjamin", "Harry"]

A new customer arrived to the barbar shop and we will add it to the queue
Adding a new customer 'Michael' to the barber queue...
The current state of the barber queue is ["Benjamin", "Harry", "Michael"]
→ PDA_evidences git:(master) ✕ █
```

I.T 6 - Example of the use of a hash in a program.

```
IT5_arrays.rb IT6_hashes.rb o
1
2 # I want to create a hash that will hold my whole collection of movies
3 # and that will provide me with some functions to add, remove, return or
4 # perform some sorting over my movies
5
6 # First I will create an array of hashes with the movies that I have
7
8 my_movies = [
9   {
10     title: "Inside Out",
11     year: 2015,
12     genre: [:animation, :comedy]
13   },
14   {
15     title: "Django Unchained",
16     year: 2012,
17     genre: [:drama, :action]
18   },
19   {
20     title: "La La Land",
21     year: 2016,
22     genre: [:musical]
23   },
24   {
25     title: "Gravity",
26     year: 2013,
27     genre: [:drama, :sci-fi]
28   },
29   {
30     title: "Mad Max: Fury Road",
31     year: 2015,
32     genre: [:drama, :action]
33   }
34 ]
35
```

```
IT5_arrays.rb IT6_hashes.rb o
35
36 # # I want to create a function that will loop through the movies within my array
37 # of hases and that will return only the hashes that include a movie of the
38 # specified genre
39
40 def return_movies_by_genre(movies, genre)
41   result = movies.select{|movie| movie[:genre].include?(genre)}
42   return result
43 end
```

```
IT5_arrays.rb IT6_hashes.rb
45
46 puts "The current movies within my movie collection are:"
47 puts my_movies
48 puts
49 puts "I want now the program to return only my drama movies"
50 puts
51 puts "DRAMA MOVIES"
52 puts "-----"
53 puts return_movies_by_genre(my_movies, :drama)
```

```
→ PDA_evidences git:(master) x ruby IT6_hashes.rb ]
```

The current movies within my movie collection are:

```
{:title=>"Inside Out", :year=>2015, :genre=>[:animation, :comedy]}  
{:title=>"Django Unchained", :year=>2012, :genre=>[:drama, :action]}  
{:title=>"La La Land", :year=>2016, :genre=>[:musical]}  
{:title=>"Gravity", :year=>2013, :genre=>[:drama, :sci-fi]}  
{:title=>"Mad Max: Fury Road", :year=>2015, :genre=>[:drama, :action]}
```

I want now the program to return only my drama movies

DRAMA MOVIES

```
-----  
{:title=>"Django Unchained", :year=>2012, :genre=>[:drama, :action]}  
{:title=>"Gravity", :year=>2013, :genre=>[:drama, :sci-fi]}  
{:title=>"Mad Max: Fury Road", :year=>2015, :genre=>[:drama, :action]}
```

```
→ PDA_evidences git:(master) x █
```

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

```
Edible.java — ~/codeclan_work/PDA_evidence/code

1  • public interface Edible{
2      float getCaloriesPerKilo();
3      float getWeightInKg();
4      float getTotalCalories();
5      String getName();
6  }
7

Project — ~/codeclan_work/PDA_evidence/code

1  public class Carrot implements Edible {
2
3      private float caloriesPerKilo;
4      private float weightInKg;
5      private String name;
6
7
8      public Carrot(float weightInKg){
9          this.caloriesPerKilo = 400.0f;
10         this.weightInKg = weightInKg;
11         this.name = "Carrot";
12     }
13
14     public float getCaloriesPerKilo(){
15         return this.caloriesPerKilo;
16     }
17
18     public float getWeightInKg(){
19         return this.weightInKg;
20     }
21
22     public String getName(){
23         return this.name;
24     }
25
26     public float getTotalCalories(){
27         return (this.weightInKg * this.caloriesPerKilo);
28     }
29
30 }
```

```
1 public class Beef implements Edible {
2
3     private float caloriesPerKilo;
4     private float weightInKg;
5     private String name;
6
7     public Beef(float weightInKg){
8         this.caloriesPerKilo = 2500.0f;
9         this.weightInKg = weightInKg;
10        this.name = "Beef";
11    }
12
13    public float getCaloriesPerKilo(){
14        return this.caloriesPerKilo;
15    }
16
17    public float getWeightInKg(){
18        return this.weightInKg;
19    }
20
21    public String getName(){
22        return this.name;
23    }
24
25    public float getTotalCalories(){
26        return (this.weightInKg * this.caloriesPerKilo);
27    }
28
29 }
29 }
30 }
```

```
1  import java.util.ArrayList;
2
3  public class Recipe{
4
5      private ArrayList<Edible> ingredientList;
6
7      public Recipe(){
8          this.ingredientList = new ArrayList<>();
9      }
10
11     public ArrayList<Edible> getEdibleList(){
12         return this.ingredientList;
13     }
14
15     public void addEdible(Edible ingredient){
16         this.ingredientList.add(ingredient);
17     }
18
19     public float getTotalCalories(){
20         float totalCalories = 0f;
21         for (Edible ingredient: ingredientList){
22             totalCalories += ingredient.getTotalCalories();
23         }
24         return totalCalories;
25     }
26
27 }
27 }
```



```
Runner.java — ~/codeclan_work/PDA_evidence/code

Ed... Be... Ca... Re... .D... Runner.java

1  • public class Runner{
2
3      public static void main (String[] args){
4
5          Recipe recipe = new Recipe();
6          Carrot carrot = new Carrot(0.3f);
7          Beef beef = new Beef(0.5f);
8          recipe.addEdible(carrot);
9          recipe.addEdible(beef);
10
11         System.out.println("Beef total calories: " +
12         • beef.getTotalCalories());
13         System.out.println("Carrot total calories: " +
14         • carrot.getTotalCalories());
15
16         System.out.println("Recipe total calories: " +
17         • recipe.getTotalCalories());
18     }
19 }
```

```
polymorphism git:(master) x java Runner
Beef total calories: 1250.0
Carrot total calories: 120.00001
Recipe total calories: 1370.0
polymorphism git:(master) x
```