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.

```java
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;
  }
}
```

Ed... Be... Ca... Re... Ru... VideoFile.java Re...

```java
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;
    }

    public void displayVideoInfo(){
        System.out.println("Title: " + super.getTitle());
        System.out.println("Running length: " + super.getLenght());
        System.out.println("Author: " + super.getAuthor());
        System.out.println("nationality: " + super.getNationality());
        System.out.println("Type: " + this.getVideoType().toString());
        System.out.println("Codec: " + this.getVideoCodec().toString());
    }
}
```

```java
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();

  }

}
```

Terminal Shell Edit View Window Help

inherita

```
→  inheritance git:(master) ✗ java Runner
Title: Breaking Bad S04E02
Running length: 55.24
Author: Vince Gilligan
nationality: USA
Type: SERIES
Codec: MOV
→  inheritance git:(master) ✗
```

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

```ruby
# The following function access a database called users and search for users
# from a specific country
def User.find_by_country(country)
  db = PG.connect({dbname: 'users', host: 'localhost'})
  sql = '
    SELECT *
    FROM users
    WHERE country = $1;
  '
  db.prepare("find_by_country", sql)
  values = [country]
  result = db.exec_prepared("find_by_country", values)
  return result.map(){
  |user_info| User.new(user_info)
  }
end
```

```ruby
# We can run it by executing the code below:

users = User.find_by_country("Spain")
users.each(){|user| puts "User named #{user.name}, from #{user.country}.
#{user.age} years old."}

# The result can be seen in the next screenshot
```

```
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

```ruby

# The following function returns all the elemtns within the database
# sorted by age in ascending order

 def User.sort_users_by_age()
   db = PG.connect({dbname: 'users', host: 'localhost'})
   sql = '
     SELECT *
     FROM users
   '
   result = db.exec(sql)
   users = result.map(){
   |user_info| User.new(user_info)
   }
   sorted_users = users.sort!() {
     |user_a, user_b| user_a.age <=> user_b.age
   }
   return sorted_users
 end

```

Project | users.sql | User.rb

IT_3_searching_data
User.rb
users.sql
music_collection

```ruby
# We will demonstrate the results by calling the funcion through
# the code below:

result = User.sort_users_by_age

result.each(){
  |user|
  puts "User name:#{user.name}, #{user.age} years old."
}

```

psql ..earching_data

```
➜  IT_3_searching_data git:(master) ✗ 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) ✗
```

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

IT5_arrays.rb

```ruby


# We want to hold an array of people who are queuing in a barbershop
# waiting for their turn to get a hair cut

barber_queue = ["Mark", "Benjamin", "Harry"]

```

IT5_arrays.rb

```ruby
# Whenever a new customer joins the queue they get pushed into the array.
# Whenever the barber calls the following customer on the queue, the first
# element from the array will be returned and removed from the array.

def add_new_customer_to_queue(customer, queue)
  result = customer.push(customer)
  return result
end

def call_next_customer_from_queue(queue)
  next_customer = queue.shift()
  return next_customer
end
```

IT5_arrays.rb

```ruby

puts "The current state of the barber queue is #{barber_queue}"
puts "We will now call the next customer in the queue..."

next_customer = call_next_customer_from_queue(barber_queue)

puts "The next customer is #{next_customer}"
puts "The current state of the barber queue is #{barber_queue}"
puts

puts "A new customer arrived to the barbar shop and we will add it to the queue"
puts "Adding a new customer 'Michael' to the barber queue..."

add_new_customer_to_queue("Michael", barber_queue)

puts "The current state of the barber queue is #{barber_queue}"

```

```
→  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

```ruby
# I want to create a hash that will hold my whole collection of movies
# and that will provide me with some functions to add, remove, return or
# perform some sorting over my movies

# First I will create an array of hashes with the movies that I have

my_movies = [
  {
    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]
  }
]

```

IT6_hashes.rb* 0 0 0 40:1 (2, 2) LF UTF-8 Ruby master 1 file

IT5_arrays.rb IT6_hashes.rb

```ruby

# # I want to create a function that will loop through the movies within my array
# of hases and that will return only the hashes that include a movie of the
# specified genre

def return_movies_by_genre(movies, genre)
  result = movies.select(){|movie| movie[:genre].include?(genre)}
  return result
end
```

IT5_arrays.rb IT6_hashes.rb

```ruby

puts "The current movies within my movie collection are:"
puts my_movies
puts
puts "I want now the program to return only my drama movies"
puts
puts "DRAMA MOVIES"
puts "----------------"
puts return_movies_by_genre(my_movies, :drama)
```

```
→  PDA_evidences git:(master) ✗ 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) ✗
```

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

Edible.java — ~/codeclan_work/PDA_evidence/code

```java
public interface Edible{
    float getCaloriesPerKilo();
    float getWeightInKg();
    float getTotalCalories();
    String getName();
}

```

Project — ~/codeclan_work/PDA_evidence/code

```java
public class Carrot implements Edible {

    private float caloriesPerKilo;
    private float weightInKg;
    private String name;


    public Carrot(float weightInKg){
        this.caloriesPerKilo = 400.0f;
        this.weightInKg = weightInKg;
        this.name = "Carrot";
    }

    public float getCaloriesPerKilo(){
        return this.caloriesPerKilo;
    }

    public float getWeightInKg(){
        return this.weightInKg;
    }

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

    public float getTotalCalories(){
        return (this.weightInKg * this.caloriesPerKilo);
    }

}
```

Ed... Beef.java Ca... Re... Ru... Ed... Re...

```java
public class Beef implements Edible {

  private float caloriesPerKilo;
  private float weightInKg;
  private String name;

  public Beef(float weightInKg){
    this.caloriesPerKilo = 2500.0f;
    this.weightInKg = weightInKg;
    this.name = "Beef";
  }

  public float getCaloriesPerKilo(){
    return this.caloriesPerKilo;
  }

  public float getWeightInKg(){
    return this.weightInKg;
  }

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

  public float getTotalCalories(){
    return (this.weightInKg * this.caloriesPerKilo);
  }

}
```

}

Ed... Be... Ca... Recipe.java Ru... Ed...

```java
import java.util.ArrayList;

public class Recipe{

  private ArrayList<Edible> ingredientList;

  public Recipe(){
    this.ingredientList = new ArrayList<>();
  }

  public ArrayList<Edible> getEdibleList(){
    return this.ingredientList;
  }

  public void addEdible(Edible ingredient){
    this.ingredientList.add(ingredient);
  }

  public float getTotalCalories(){
    float totalCalories = 0f;
    for (Edible ingredient: ingredientList){
      totalCalories += ingredient.getTotalCalories();
    }
    return totalCalories;
  }

}
}
```

Runner.java — ~/codeclan_work/PDA_evidence/code

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

```java
public class Runner{

  public static void main (String[] args){

    Recipe recipe = new Recipe();
    Carrot carrot = new Carrot(0.3f);
    Beef beef = new Beef(0.5f);
    recipe.addEdible(carrot);
    recipe.addEdible(beef);

    System.out.println("Beef total calories: " +
    beef.getTotalCalories());
    System.out.println("Carrot total calories: " +
    carrot.getTotalCalories());


    System.out.println("Recipe total calories: " +
    recipe.getTotalCalories());
  }
}
```

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