Encapsulation Lab 1

Lab 1

This lab will focus on building a journal to rate different coffees. The journal will use encapsulation. Users enter the name of the roaster, the country of origin, the region and how many stars (*) they rate the coffee. Entries are saved to a CSV file. Users can display coffees already in the journal, or they can add new coffees to the journal. There are two main parts to this project, the `CoffeeJournal` class and the command line interface.

The `CoffeeJournal` Class

Since this project will be reading and writing to a CSV file, we need to first import the CSV module. After that we are going to create the constructor for the `CoffeeJournal` class. The `file` attribute is the file path the CSV file. The next four attributes are strings related to the coffee. Finally, the `coffee` attribute is going to be the contents of the CSV file stored as an ArrayList of arrays of strings. Even if no coffees have been saved, the CSV file contains headers for each column.

```
//add class definitions below this line

class CoffeeJournal {
  private String file;
  private String roaster;
  private String country;
  private String region;
  private String stars;
  private ArrayList<String[]> coffee;

  public CoffeeJournal(String f) {
    file = f;
    roaster = "";
    country = "";
    region = "";
    stars = "";
    coffee = loadCoffee();
  }
}

//add class definitions above this line
```

`loadCoffee` is a helper method that returns the contents of the CSV file. The file is read one line at a time. Each line is an array of strings, which is then added to the ArrayList `savedCoffee`. Once the file has been read, the method returns `savedCoffee`.

```
  private ArrayList<String[]> loadCoffee() {
    ArrayList<String[]> savedCoffee = new ArrayList<String[]>();
    try {
      CSVReader reader = new CSVReader(new FileReader(file));
      for (String[] row : reader) {
        savedCoffee.add(row);
      }
      reader.close();
    } catch (Exception e) {
      System.out.println(e);
    }
    return savedCoffee;
  }
```

Testing Your Code

For the sake of legibility, the path to the CSV file is stored in its own variable. Instantiate a `CoffeeJournal` object and print this object. Java will print the object type followed by its location in memory. You should see something like this: `CoffeeJournal@378bf509`. Your memory location will be different.

```
    //add code below this line

    String file = "code/encapsulation/testJournal1.csv";
    CoffeeJournal journal = new CoffeeJournal(file);
    System.out.println(journal);

    //add code above this line
```

▼ **Self-Check**

This is what your code should look like.

```
import java.io.*;
import com.opencsv.*;
import org.apache.commons.lang3.ObjectUtils;
import java.util.ArrayList;
import java.util.Scanner;

//add class definitions below this line

class CoffeeJournal {
  private String file;
  private String roaster;
  private String country;
  private String region;
  private String stars;
  private ArrayList<String[]> coffee;

  public CoffeeJournal(String f) {
    file = f;
    roaster = "";
    country = "";
    region = "";
    stars = "";
    coffee = loadCoffee();
  }

  private ArrayList<String[]> loadCoffee() {
    ArrayList<String[]> savedCoffee = new ArrayList<String[]>
      ();
    try {
```

```
      CSVReader reader = new CSVReader(new FileReader(file));
      for (String[] row : reader) {
        savedCoffee.add(row);
      }
      reader.close();
    } catch (Exception e) {
      System.out.println(e);
    }
    return savedCoffee;
  }
}

//add class definitions above this line

public class EncapsulationLab1 {
  public static void main(String[] args) {

    //add code below this line

    String file = "code/encapsulation/testJournal1.csv";
    CoffeeJournal journal = new CoffeeJournal(file);
    System.out.println(journal);

    //add code above this line
  }

  //add method definitions below this line



  //add method definitions above this line
}
```

Encapsulation Lab 2

Lab 2

The next step is to add getter and setter methods for the `roaster`, `country`, `region`, and `stars` attributes. These methods should go after the `loadCoffee` method but still be a part of the `CoffeeJournal` class.

```
public String getRoaster() {
  return roaster;
}

public void setRoaster(String newRoaster) {
  roaster = newRoaster;
}

public String getCountry() {
  return country;
}

public void setCountry(String newCountry) {
  country = newCountry;
}

public String getRegion() {
  return region;
}

public void setRegion(String newRegion) {
  region = newRegion;
}

public String getStars() {
  return stars;
}

public void setStars(String newStars) {
  stars = newStars;
}
```

Testing Your Code

Use the newly created setters to add a value for each of the four attributes. Then print these attributes with the getter methods.

```
  //add code below this line

  String file = "code/encapsulation/testJournal2.csv";
  CoffeeJournal journal = new CoffeeJournal(file);
  journal.setRoaster("Peace River");
  journal.setCountry("Rawanda");
  journal.setRegion("Remera");
  journal.setStars("***");

  System.out.println(journal.getRoaster());
  System.out.println(journal.getCountry());
  System.out.println(journal.getRegion());
  System.out.println(journal.getStars());

  //add code above this line
```

▼ **Self-Check**

This is what your code should look like.

```
import java.io.*;
import com.opencsv.*;
import org.apache.commons.lang3.ObjectUtils;
import java.util.ArrayList;
import java.util.Scanner;

//add class definitions below this line

class CoffeeJournal {
  private String file;
  private String roaster;
  private String country;
  private String region;
  private String stars;
  private ArrayList<String[]> coffee;

  public CoffeeJournal(String f) {
    file = f;
    roaster = "";
    country = "";
    region = "";
    stars = "";
```

```
    coffee = loadCoffee();
  }

  private ArrayList<String[]> loadCoffee() {
    ArrayList<String[]> savedCoffee = new ArrayList<String[]>
      ();
    try {
      CSVReader reader = new CSVReader(new FileReader(file));
      for (String[] row : reader) {
        savedCoffee.add(row);
      }
      reader.close();
    } catch (Exception e) {
      System.out.println(e);
    }
    return savedCoffee;
  }

  public String getRoaster() {
    return roaster;
  }

  public void setRoaster(String newRoaster) {
    roaster = newRoaster;
  }

  public String getCountry() {
    return country;
  }

  public void setCountry(String newCountry) {
    country = newCountry;
  }

  public String getRegion() {
    return region;
  }

  public void setRegion(String newRegion) {
    region = newRegion;
  }

  public String getStars() {
    return stars;
  }

  public void setStars(String newStars) {
    stars = newStars;
```

```
  }
}

//add class definitions above this line

public class EncapsulationLab2 {
  public static void main(String[] args) {

    //add code below this line

    String file = "code/encapsulation/testJournal2.csv";
    CoffeeJournal journal = new CoffeeJournal(file);
    journal.setRoaster("Peace River");
    journal.setCountry("Rawanda");
    journal.setRegion("Remera");
    journal.setStars("***");

    System.out.println(journal.getRoaster());
    System.out.println(journal.getCountry());
    System.out.println(journal.getRegion());
    System.out.println(journal.getStars());

    //add code above this line
  }

  //add method definitions below this line



  //add method definitions above this line
}
```

Encapsulation Lab 3

Lab 3

We will finish up the `CoffeeJournal` class with some methods that add a coffee to the journal, prints the journal, and saves the updated journal to the CSV file. Printing the journal will use string formatting to make the output look organized. Only when the user quits the program will the new information be saved to the CSV file. All of these methods are public.

Methods

The `addCoffee` method stores the `roaster`, `country`, `region`, and `stars` attributes in an array and then adds it to the `coffee` ArrayList. This method should go after the last setter method but remain in the `CoffeeJournal` class.

```
public void addCoffee() {
  String[] newCoffee = new String[4];
  newCoffee[0] = roaster;
  newCoffee[1] = country;
  newCoffee[2] = region;
  newCoffee[3] = stars;
  coffee.add(newCoffee);
}
```

The `save` method opens the CSV file in write mode. It then iterates over the `coffee` attribute. Each array of strings is written to the file. **Note**, eventually the `save` method will be linked to stopping the program. So saving should be the last thing you do before exiting the program. This method should go after `addCoffee` but remain in the `CoffeeJournal` class.

```
public void save() {
  try {
    CSVWriter writer = new CSVWriter(new FileWriter(file));
    for (String[] c : coffee) {
      writer.writeNext(c);
    }
    writer.close();
  } catch (IOException e) {
    System.out.println(e);
  }
}
```

The `showCoffee` method takes into account two cases. First, there is no information about a coffee in either the `coffee` attribute. If the size of `coffee` is one there is no coffee information (the one row is the headers). In this case, prompt the user to enter information. In all ther cases, iterate through `coffee` and print each element in the array of strings. `String.format` allows you to add spacing to the output. This method should go after `save` but remain in the `CoffeeJournal` class.

```
public void showCoffee() {
  if (coffee.size() == 1) {
    System.out.println("Enter a coffee");
  } else {
    for (String[] row : coffee) {
      String info = String.format("%-13s %-13s %-13s %-13s",
      row[0], row[1], row[2], row[3]);
      System.out.println(info);
    }
  }
}
```

Testing Your Code

Test 1

Let's make sure your class is working as expected. The first test is going to try and print the coffee journal with no information in it.

```
    //add code below this line

    String file = "code/encapsulation/testJournal3.csv";
    CoffeeJournal journal = new CoffeeJournal(file);
    journal.showCoffee();

    //add code above this line
```

You should see the following output:

```
Enter a coffee
```

Test 2

Change your testing code to look like the code below. The second test is going to add a coffee to the journal and print its contents.

```
    //add code below this line

    String file = "code/encapsulation/testJournal3.csv";
    CoffeeJournal journal = new CoffeeJournal(file);
    journal.setRoaster("Peace River");
    journal.setCountry("Rawanda");
    journal.setRegion("Remera");
    journal.setStars("***");
    journal.addCoffee();
    journal.showCoffee();
    journal.save();

    //add code above this line
```

You should see the following output:

```
Roaster         Country         Region          Stars
Peace River     Rawanda         Remera          ***
```

Test 3

Change your testing code to look like the code below. The third test is going to add a coffee to the journal and print the coffee information already stored in the CSV file (Test 2) plus the newly entered coffee.

```
    //add code below this line

    String file = "code/encapsulation/testJournal3.csv";
    CoffeeJournal journal = new CoffeeJournal(file);
    journal.setRoaster("Peace River");
    journal.setCountry("Ethiopia");
    journal.setRegion("Sidoma");
    journal.setStars("****");
    journal.addCoffee();
    journal.showCoffee();
    journal.save();

    //add code above this line
```

You should see the following output:

```
Roaster         Country         Region          Stars
Peace River     Rawanda         Remera          ***
Peace River     Ethiopia        Sidoma          ****
```

▼ Self-Check

This is what your code should look like.

```
import java.io.*;
import com.opencsv.*;
import org.apache.commons.lang3.ObjectUtils;
import java.util.ArrayList;
import java.util.Scanner;

//add class definitions below this line

class CoffeeJournal {
  private String file;
  private String roaster;
  private String country;
  private String region;
  private String stars;
  private ArrayList<String[]> coffee;

  public CoffeeJournal(String f) {
    file = f;
    roaster = "";
    country = "";
    region = "";
    stars = "";
```

```
    coffee = loadCoffee();
  }

  private ArrayList<String[]> loadCoffee() {
    ArrayList<String[]> savedCoffee = new ArrayList<String[]>
      ();
    try {
      CSVReader reader = new CSVReader(new FileReader(file));
      for (String[] row : reader) {
        savedCoffee.add(row);
      }
      reader.close();
    } catch (Exception e) {
      System.out.println(e);
    }
    return savedCoffee;
  }

  public String getRoaster() {
    return roaster;
  }

  public void setRoaster(String newRoaster) {
    roaster = newRoaster;
  }

  public String getCountry() {
    return country;
  }

  public void setCountry(String newCountry) {
    country = newCountry;
  }

  public String getRegion() {
    return region;
  }

  public void setRegion(String newRegion) {
    region = newRegion;
  }

  public String getStars() {
    return stars;
  }

  public void setStars(String newStars) {
    stars = newStars;
```

```
  }

  public void addCoffee() {
    String[] newCoffee = new String[4];
    newCoffee[0] = roaster;
    newCoffee[1] = country;
    newCoffee[2] = region;
    newCoffee[3] = stars;
    coffee.add(newCoffee);
  }

  public void save() {
    try {
      CSVWriter writer = new CSVWriter(new FileWriter(file));
      for (String[] c : coffee) {
        writer.writeNext(c);
      }
      writer.close();
    } catch (IOException e) {
      System.out.println(e);
    }
  }

  public void showCoffee() {
    if (coffee.size() == 1) {
      System.out.println("Enter a coffee");
    } else {
      for (String[] row : coffee) {
        String info = String.format("%-13s %-13s %-13s %-13s",
      row[0], row[1], row[2], row[3]);
        System.out.println(info);
      }
    }
  }
}

//add class definitions above this line

public class EncapsulationLab3 {
  public static void main(String[] args) {

    //add code below this line

    // Test 1
    String file = "code/encapsulation/testJournal3.csv";
    CoffeeJournal journal = new CoffeeJournal(file);
    journal.showCoffee();
```

```
    // Test 2
    String file = "code/encapsulation/testJournal3.csv";
    CoffeeJournal journal = new CoffeeJournal(file);
    journal.setRoaster("Peace River");
    journal.setCountry("Rawanda");
    journal.setRegion("Remera");
    journal.setStars("***");
    journal.addCoffee();
    journal.showCoffee();
    journal.save();

    // Test 3
    String file = "code/encapsulation/testJournal3.csv";
    CoffeeJournal journal = new CoffeeJournal(file);
    journal.setRoaster("Peace River");
    journal.setCountry("Ethiopia");
    journal.setRegion("Sidoma");
    journal.setStars("****");
    journal.addCoffee();
    journal.showCoffee();
    journal.save();

    //add code above this line
  }

  //add method definitions below this line



  //add method definitions above this line
}
```

Encapsulation Lab 4

Lab 4

The `CoffeeJournal` is now complete. This lab focuses on how using an object from this class and building a command line interface around it. The following code should be added to the method definition section at the end.

Command Line Interface

Start by creating some helper methods that are not a part of the `CoffeeJournal` class. We want a menu-driven interface. The user will enter a loop and be presented with a list of choices, which is the `mainMenu` method. The `performAction` method takes the user choice and performs the appropriate action. **Note**, the code examples uses the parameter `journal` which is an instance of the `CoffeeJournal` class.

The menu provides three options. The first one can be handled by the `CoffeeJournal` object. The other two options require the `CoffeeJournal` object but need some additional assistance. The `enterCoffee` method prompts the user to enter information about the new coffee, while the `quit` method saves the new information and exits the loop. Add these methods after `performAction`. **Note**, `run` is a boolean variable that controls the loop.

```
//add method definitions below this line

public static int mainMenu() {
  Scanner sc = new Scanner(System.in);
  System.out.println();
  System.out.println("Coffess of the World");
  System.out.println("\t1. Show Coffee");
  System.out.println("\t2. Add Coffee");
  System.out.println("\t3. Save and Quit");
  return sc.nextInt();
}

public static boolean performAction(CoffeeJournal journal, int
      choice) {
  if (choice == 1) {
    System.out.println();
    journal.showCoffee();
  } else if (choice == 2) {
    enterCoffee(journal);
  } else if (choice == 3) {
    quit(journal);
    return false;
  }
  return true;
}

//add method definitions above this line
```

The `enterCoffee` method asks the user to input information about the coffee. Setters are used to update the attributes of the `CoffeeJournal` object. Finally, these attributes are added to the `coffee` attribute. The `quit` method saves the information to the CSV file, and stops the loop by setting `run` to `false`.

```
public static void enterCoffee(CoffeeJournal journal) {
  Scanner sc = new Scanner(System.in);
  System.out.println();
  System.out.println("Enter the name of the roaster: ");
  String newRoaster = sc.nextLine();
  journal.setRoaster(newRoaster);
  System.out.println("Enter the country of origin: ");
  String newCountry = sc.nextLine();
  journal.setCountry(newCountry);
  System.out.println("Enter the region: ");
  String newRegion = sc.nextLine();
  journal.setRegion(newRegion);
  System.out.println("Enter the number of stars '*' (*, **,
      ***, or ****): ");
  String newStars = sc.nextLine();
  journal.setStars(newStars);
  System.out.println();
  journal.addCoffee();
}

public static void quit(CoffeeJournal journal) {
  System.out.println("Closing the coffee journal");
  journal.save();
}
```

Testing Your Code

To test our code, we are going to set up a loop that controls the command line application. Create a `CoffeeJournal` object as before. In addition, create the variable `run` which controls the loop, and `choice` which represents the user input. The loop should run as long as `run` is `true`. Present the user with the menu of options. Finally perform the desired action.

```
  //add code below this line

  String file = "code/encapsulation/coffeeJournal.csv";
  CoffeeJournal journal = new CoffeeJournal(file);
  boolean run = true;
  int choice;

  while (run) {
    choice = mainMenu();
    run = performAction(journal, choice);
  }

  //add code above this line
```

Use the following information to enter two new coffees.
Roaster	Country	Origin	Stars
:——	:——	:——	:—:
Ritual	Guatemala	Antigua	***
Oak Cliff	Peru	Oxapampa	**

▼ **Using the Terminal**
When using a `Scanner` object, you must type something in the terminal. That is why there is a terminal below the IDE.

You should see the following output:

```
Roaster         Country         Region          Stars
Peace River     Rawanda         Remera          ***
Ritual          Guatemala       Antigua         ***
Oak Cliff       Peru            Oxapampa        **
```

▼ **Self-Check**

This is what your code should look like.

```
import java.io.*;
import com.opencsv.*;
import org.apache.commons.lang3.ObjectUtils;
import java.util.ArrayList;
import java.util.Scanner;

//add class definitions below this line

class CoffeeJournal {
  private String file;
```

```
  private String roaster;
  private String country;
  private String region;
  private String stars;
  private ArrayList<String[]> coffee;

  public CoffeeJournal(String f) {
    file = f;
    roaster = "";
    country = "";
    region = "";
    stars = "";
    coffee = loadCoffee();
  }

  private ArrayList<String[]> loadCoffee() {
    ArrayList<String[]> savedCoffee = new ArrayList<String[]>
      ();
    try {
      CSVReader reader = new CSVReader(new FileReader(file));
      for (String[] row : reader) {
        savedCoffee.add(row);
      }
      reader.close();
    } catch (Exception e) {
      System.out.println(e);
    }
    return savedCoffee;
  }

  public void setRoaster(String newRoaster) {
    roaster = newRoaster;
  }

  public void setCountry(String newCountry) {
    country = newCountry;
  }

  public void setRegion(String newRegion) {
    region = newRegion;
  }

  public void setStars(String newStars) {
    stars = newStars;
  }

  public void addCoffee() {
    String[] newCoffee = new String[4];
```

```
    newCoffee[0] = roaster;
    newCoffee[1] = country;
    newCoffee[2] = region;
    newCoffee[3] = stars;
    coffee.add(newCoffee);
  }

  public void save() {
    try {
      CSVWriter writer = new CSVWriter(new FileWriter(file));
      for (String[] c : coffee) {
        writer.writeNext(c);
      }
      writer.close();
    } catch (IOException e) {
      System.out.println(e);
    }
  }

  public void showCoffee() {
    if (coffee.size() == 1) {
      System.out.println("Enter a coffee");
    } else {
      for (String[] row : coffee) {
        String info = String.format("%-13s %-13s %-13s %-13s",
      row[0], row[1], row[2], row[3]);
        System.out.println(info);
      }
    }
  }
}

//add class definitions above this line

public class EncapsulationLab4 {
  public static void main(String[] args) {

    //add code below this line

    String file = "code/encapsulation/coffeeJournal.csv";
    CoffeeJournal journal = new CoffeeJournal(file);
    boolean run = true;
    int choice;

    while (run) {
      choice = mainMenu();
      run = performAction(journal, choice);
    }
```

```
    //add code above this line
}

//add method definitions below this line
public static int mainMenu() {
  Scanner sc = new Scanner(System.in);
  System.out.println();
  System.out.println("Coffess of the World");
  System.out.println("\t1. Show Coffee");
  System.out.println("\t2. Add Coffee");
  System.out.println("\t3. Save and Quit");
  return sc.nextInt();
}

public static boolean performAction(CoffeeJournal journal,
    int choice) {
  if (choice == 1) {
    System.out.println();
    journal.showCoffee();
  } else if (choice == 2) {
    enterCoffee(journal);
  } else if (choice == 3) {
    quit(journal);
    return false;
  }
  return true;
}

public static void enterCoffee(CoffeeJournal journal) {
  Scanner sc = new Scanner(System.in);
  System.out.println();
  System.out.println("Enter the name of the roaster: ");
  String newRoaster = sc.nextLine();
  journal.setRoaster(newRoaster);
  System.out.println("Enter the country of origin: ");
  String newCountry = sc.nextLine();
  journal.setCountry(newCountry);
  System.out.println("Enter the region: ");
  String newRegion = sc.nextLine();
  journal.setRegion(newRegion);
  System.out.println("Enter the number of stars '*' (*, **,
    ***, or ****): ");
  String newStars = sc.nextLine();
  journal.setStars(newStars);
  System.out.println();
  journal.addCoffee();
}
```

```
  public static void quit(CoffeeJournal journal) {
    System.out.println("Closing the coffee journal");
    journal.save();
  }

  //add method definitions above this line
}
```

Lab Challenge

Lab Challenge

Problem
Write a class named `Person` that has attributes for name, age, and occupation. These attributes should be private. Create getters and setters for each attribute following Java conventions.

Expected Output
* Declare the instance `new Person("Citra Curie", 16, "student")`
* The method `getName()` returns `Citra Curie`
* The method `setName("Rowan Faraday")` changes the name attribute to "Rowan Faraday"
* The method `getAge()` returns 16
* The method `setAge(18)` changes the age attribute to 18
* The method `getOccupation()` returns `student`
* The method `setOccupation("plumber")` changes the occupation attribute to "plumber"

Testing Your Code
Use the button below to test your code before submitting it for evaluation.