Overview

Our work this week includes a class called Country and a class called CountryDisplayer. Country holds information about a given country object, including a HashMap that stores its name and several pieces of data, like population, energy consumption, and more. It also includes methods such as getName (gets name of country), getCountryInfo (returns a requested piece of data about a country as a string). CountryDisplayer interacts with the user to display certain information about a country object. The user can enter what data file they would like to get the information from, and it includes methods such as getCountryByName (returns a country object that matches the name the user entered) and getCountryMaxIndicator (returns a country object with the max value for a data piece that the user requests).

Example of Running Program

To run the program, type:

```
$ javac CountryDisplayer.java
$ java CountryDisplayer
```

The program will then ask which data file you would like to use. You will probably want to say "CountryDataset.csv"

Next, you will be asked which country you want information about. Type a country name, such as:

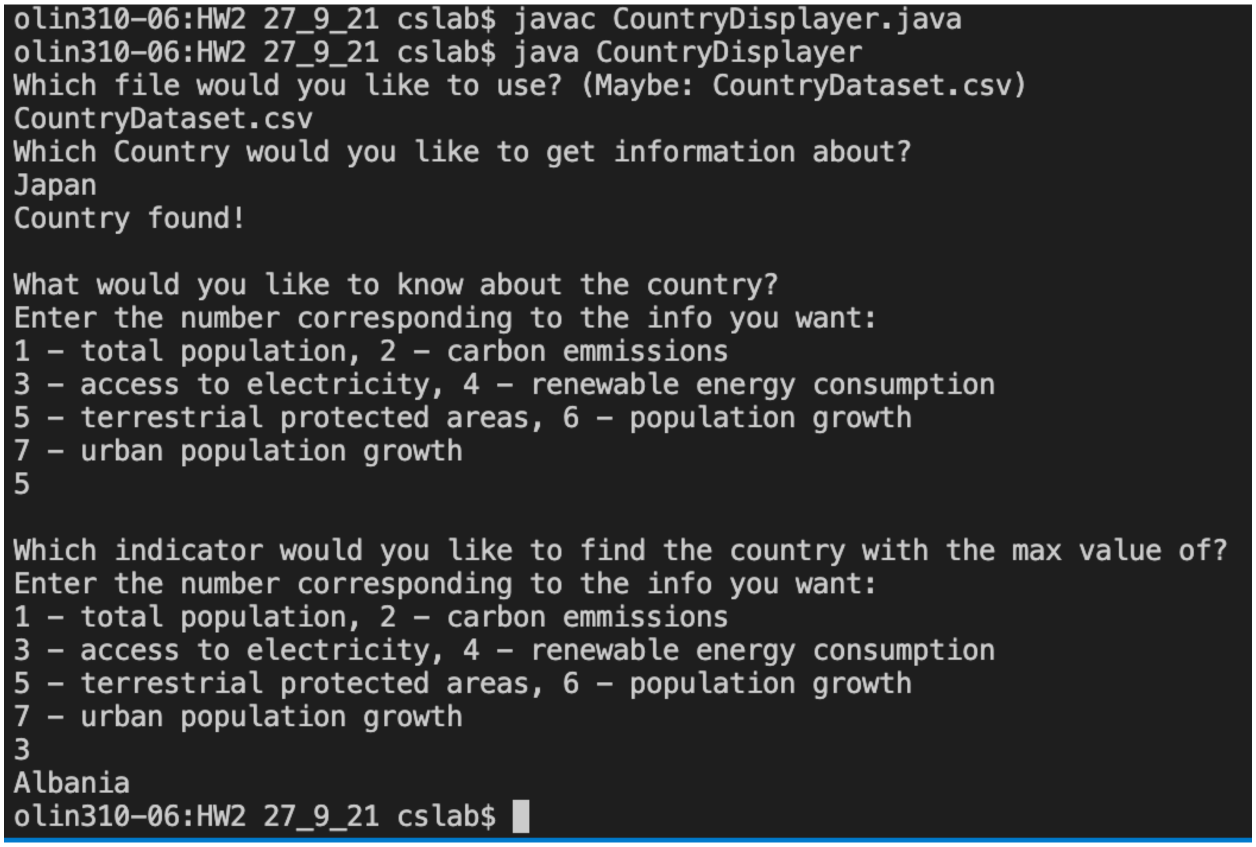
"Angola", "Yemen" or "Japan"

Then, the program will ask you what you would like to know about this country. Enter a number 1 through 7 that corresponds to the information you would like. For example:

"1", "3" ,"7"

Finally, the program will ask you which indicator you would like to find the country with the maximum value of. Again, enter a number 1-7, for example:

"2" "6" "4"



```
olin310-06:HW2 27_9_21 cslab$ javac CountryDisplayer.java
olin310-06:HW2 27_9_21 cslab$ java CountryDisplayer
Which file would you like to use? (Maybe: CountryDataset.csv)
CountryDataset.csv
Which Country would you like to get information about?
Japan
Country found!

What would you like to know about the country?
Enter the number corresponding to the info you want:
1 - total population, 2 - carbon emmissions
3 - access to electricity, 4 - renewable energy consumption
5 - terrestrial protected areas, 6 - population growth
7 - urban population growth
5

Which indicator would you like to find the country with the max value of?
Enter the number corresponding to the info you want:
1 - total population, 2 - carbon emmissions
3 - access to electricity, 4 - renewable energy consumption
5 - terrestrial protected areas, 6 - population growth
7 - urban population growth
3
Albania
olin310-06:HW2 27_9_21 cslab$ 
```

Prompts

Why did you choose the user interaction process that you did? Do you have everything on the command line or interactive input or a mix?

We have everything that involves user interaction as interactive input. We chose this approach because it’s easier and more understandable for the user to simply be asked questions and given instructions by the program printing them out and to type in their answer. We thought it would be harder for them to have to figure out how to enter command line arguments. Also, we chose the interactive input approach because it helps build connections between the user and the program, since the program can say friendly and helpful things to them. We thought this would improve the user’s overall experience, as they would feel like they had a cheerful and easy time using the program.

Why does it make sense to make this program object-oriented? You didn’t have a choice, but you should reflect on the benefits of an object-oriented approach to this task instead of simply looking through the data file or lists, for example.

Making this program object-oriented is beneficial because it's a very organized way to program. Different cases, or instances, in which the program would be run are organized into different objects, which keeps things neat and understandable. Object-oriented code is easy to modify, use, and re-use in a variety of different situations. It is more abstract than creating a program that only works with a very specific data file or list and only performs a very specific set of actions with it. Also, it helps for a beginning programmer to be able to think about uses of their program and different situations in terms of objects, since everyone's experience with the world is pretty object-oriented. In other words, it helps the programmer conceptualize their program more easily. One last reason is that it helps the person reading the code understand what's going on, since it clearly labels different objects and how they're used.

Rubric Items

README thorough and detailed

Here it is :)

User can specify data file

```
Scanner scanner = new Scanner(System.in);

System.out.println("Which file would you like to use? (Maybe: CountryDataset.csv)");

String fileName = scanner.nextLine();

CountryDisplayer file1 = new CountryDisplayer(fileName);
```

This code shows the program asking the user which data file they would like to use. It then creates a new instance of the class CountryDisplayer, taking in that data file. For example, the user can enter “CountryDataset.csv” and the program will take all the information from that file.

User can look up a specific indicator about a specific country

Method getCountryByName in CountryDisplayer that returns a Country object at user’s request:

```
public Country getCountryByName(String name) {

       for(Country temp: countries){

           if(temp.getName().equals(name)){

               System.out.println("Country found!");

               return temp;

           }

       }

       System.out.println("Sorry, that country is not in the list. Please run the
program again.");

       System.exit(0);

       return null;
```

```
    }
```

Method getCountryInfo in Country that returns a string that represents the data from a certain category for a certain Country at user’s request:

```
public String getCountryInfo(String countryName, String category){

        if(isInteger(category)!=true){

            System.out.println("Sorry, that indicator is not in the list.");

            System.out.println("Please run the program again and enter a number.");

            return "";

        }

        int indicatorInt = Integer.parseInt(category);

        if(indicatorInt<1 || indicatorInt>7){

            System.out.println("Sorry, that indicator is not in the list. Please run the
program again.");

            return "";

        }

        return this.countryData.get(countryName).get(Integer.parseInt(category));
```

```
   }
```

Code in CountryDisplayer's main that calls the pervious 2 methods:

```
System.out.println("Which Country would you like to get information about?");

String countryRequested = scanner.nextLine();

Country userCountry = file1.getCountryByName(countryRequested);

System.out.println("\nWhat would you like to know about the country?");

System.out.println("Enter the number corresponding to the info you want:");

System.out.println("1 - total population, 2 - carbon emissions");

System.out.println("3 - access to electricity, 4 - renewable energy consumption");

System.out.println("5 - terrestrial protected areas, 6 - population growth");

System.out.println("7 - urban population growth");

String infoRequested = scanner.nextLine();

String info = userCountry.getCountryInfo(countryRequested,infoRequested);

if(info.equals("")){

       System.out.println(info);
}
```

This code shows how the program asks the user which country they would like information about, then finds the country object with that country name, or tells them that they were unable to find that country (this part is written inside the getCountryByName method). Then, it asks the user what they would like to know about the country, and uses the getCountryInfo method to return information based on the specific indicator that the user requests. Again, if the user requests an indicator that does not exist, the program will tell them to try again.

User can look up country with max or min of a specific indicator

Method getCountryMaxIndicator in CountryDisplayer that returns the Country with the max value for a certain data category:

```
public Country getCountryMaxIndicator(String indicator) {

       Country max = countries.get(0);



       if(isInteger(indicator)!=true){

           System.out.println("Sorry, that indicator is not in the list.");

           System.out.println("Please run the program again and enter a number 1-7.");

           return null;

       }
```

```
        int indicatorInt = Integer.parseInt(indicator);
        if(indicatorInt<1 || indicatorInt>7){
            System.out.println("Sorry, that indicator is not in the list. Please run the program again.");
            return null;
        }
        for(Country temp: countries){
            String stringTemp = temp.getCountryInfo(temp.getName(), indicator);
            double doubleTemp = Double.parseDouble(stringTemp);
            String maxName = max.getName();
            String maxInfo = max.getCountryInfo(maxName, indicator);
            double maxDouble = Double.parseDouble(maxInfo);
            if(doubleTemp > maxDouble){
                max = temp;
            }
        }
        return max;
```

```
    }
```

Code in CountryDisplayer’s main that calls the getCountryMaxIndicator method:

```
System.out.println("\nWhich indicator would you like to find the country with the max
value of?");

System.out.println("Enter the number corresponding to the info you want:");

System.out.println("1 - total population, 2 - carbon emissions");

System.out.println("3 - access to electricity, 4 - renewable energy consumption");

System.out.println("5 - terrestrial protected areas, 6 - population growth");

System.out.println("7 - urban population growth");

String maxIndRequested = scanner.nextLine();

Country maxCountry = file1.getCountryMaxIndicator(maxIndRequested);

if(maxCountry != null){

       System.out.println(maxCountry.getName());

}
```

This code shows how the program asks the user which indicator they would like to find a country with the max of. If they enter anything that isn’t a number from 1 to 7, the program will let them know that that doesn’t work and encourage them to try again.

Handles user typing incorrect country name or indicator

If the user types the wrong indicator because they entered a word instead of a number:

```
if(isInteger(category)!=true){

            System.out.println("Sorry, that indicator is not in the list.");

            System.out.println("Please run the program again and enter a number.");

            return "";

        }

```

If the user types the wrong indicator because they entered a number but it wasn't between 1 and 7:

```
if(indicatorInt<1 || indicatorInt>7){

            System.out.println("Sorry, that indicator is not in the list. Please run the
program again.");

            return "";

        }
```

- This one returns an empty string because it's in the method getCountryInfo, which returns a string.

```
if(indicatorInt<1 || indicatorInt>7){

```

```
           System.out.println("Sorry, that indicator is not in the list. Please run the
program again.");

           return null;

       }
```

- This one returns null because it's in the method getCountryMaxIndicator, which returns a Country object.

If the user enters a name of a country that is not included in the spreadsheet:

```
public Country getCountryByName(String name) {

       for(Country temp: countries){

           if(temp.getName().equals(name)){

               System.out.println("Country found!");

               return temp;

           }

       }

       System.out.println("Sorry, that country is not in the list. Please run the
program again.");

       System.exit(0);

       return null;
```

```
    }
```

These snippets of code show how the program handles the user typing the wrong thing in several different methods. The general pattern is that the program tells the user why they weren't able to return the requested info, and asks them to try again.

Program is object-oriented as specified

```
Scanner scanner = new Scanner(System.in);

        System.out.println("Which file would you like to use? (Maybe:
CountryDataset.csv)");

        String fileName = scanner.nextLine();

        CountryDisplayer file1 = new CountryDisplayer(fileName);
```

This code shows how we create an instance of CountryDisplayer in the main method of the class.

Sufficient documentation

```
/**

     * gets country with the maximum data value in a specific category
```

```
    * tells user if category they requested is not in the data

    * @param indicator (String) category of data that the user wants to know about

    * @return the Country object with max value in the requested  data category

    */

   public Country getCountryMaxIndicator(String indicator){
```

This is an example of our JavaDocs documentation, that explains the method getCountryMaxIndicator.

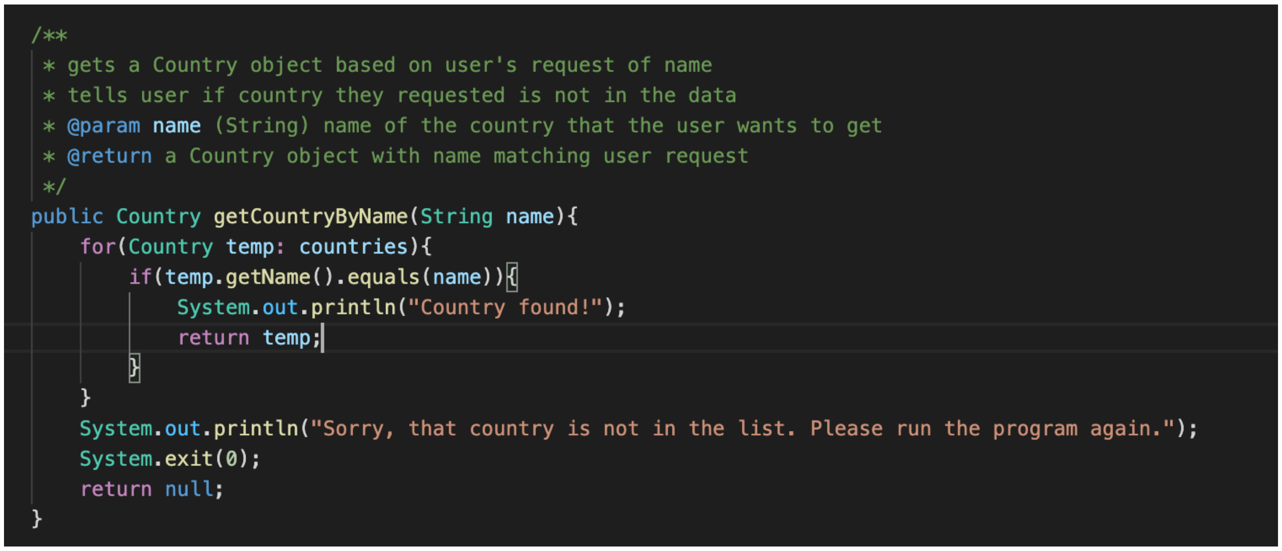
Style

```
/**

    * tests to see if a string the user entered is a number

    * @param userString the string the user entered

    * @return (boolean) true if entry is a number, false if not
```



```
/**
 * gets a Country object based on user's request of name
 * tells user if country they requested is not in the data
 * @param name (String) name of the country that the user wants to get
 * @return a Country object with name matching user request
 */
public Country getCountryByName(String name){
    for(Country temp: countries){
        if(temp.getName().equals(name)){
            System.out.println("Country found!");
            return temp;
        }
    }
    System.out.println("Sorry, that country is not in the list. Please run the program again.");
    System.exit(0);
    return null;
}
```

This example shows that our brackets, indentation, etc. are all in the right place, following Java Style Guide.