Lab 14

1. Write a method to check if a `String` is a palindrome or not. Use Scanner to ask the user for a `String` and call your method.

- **Hint** A palindrome is a word that is the same forwards and backwards. You'll need two variables to go through your String to compare the characters. One variable will start at the beginning of the String and the other will start at the end of the String. You'll need to use a loop to compare the characters at each position. If the characters are not the same, the word is not a palindrome. If you get through the entire word and all the characters are the same, the word is a palindrome.

```
public static boolean isPalindrome(String str) {
  int i = 0;
  int j = str.length() - 1;

  while (i < j) {
    if (str.charAt(i) != str.charAt(j)) {
      return false;
    }
    i++;
    j--;
  }
  return true;
}
```

2. Write a method that takes a `String` and returns a new `String` that is the reverse of the original `String`. Use Scanner to ask the user for a `String` and call your method.

```
public static String reverseString(String str) {
  String reversed = "";
  for (int i = str.length() - 1; i >= 0; i--) {
    reversed += str.charAt(i);
  }
  return reversed;
}
```

3. Write a method that checks if a SSN is in a valid format. Use Scanner to ask the user for a SSN and call your method. A valid SSN is in the format `###-##-####` where each `#` is a digit. The SSN must be 11 characters long including the dashes. Remember

- **Hint** Remember that the `String` class has a method called `charAt` that will return the character at a specific index. You can use this method to check if the characters at the correct positions are digits or dashes.

```
public static boolean isValidSSN(String ssn) {
  if (ssn.length() != 11) {
    return false;
  }

  for (int i = 0; i < ssn.length(); i++) {
    if (i == 3 || i == 6) {
      if (ssn.charAt(i) != '-') {
        return false;
      }
    } else {
      if (!Character.isDigit(ssn.charAt(i))) {
        return false;
      }
    }
  }
  return true;
}
```

4. Read in the data from the file `in.csv`. Store the ID, first name, last name, and account balance in separate ArrayLists. You should use the correct data type for each ArrayList depending on the data in the file.

```
1, John, Craft, 5561.1
2, Jane, Lane, 66002.55
3, Joan, Dion, 3123.45
4, Jack, Kent, 1541.56
5, Jenn, Kimble, 82145.32
6, Jina, Hagert, 73690.25
7, Jake, Smith, 9345.23
8, Jose, Diaz, 2323.23
9, Jade, Doe, 65890.39
```

```
import java.io.*;
import java.util.*;

public static void main(String[] args) throws IOException {

    ArrayList<Integer> ids = new ArrayList<>();
    ArrayList<String> firstNames = new ArrayList<>();
    ArrayList<String> lastNames = new ArrayList<>();
    ArrayList<Double> accountBalances = new ArrayList<>();

    File file = new File("in.csv");
    Scanner input = new Scanner(file);
    while (input.hasNextLine()) {
```

```
      String line = input.nextLine();
      String[] data = line.split(", ");
      ids.add(Integer.parseInt(data[0]));
      firstNames.add(data[1]);
      lastNames.add(data[2]);
      accountBalances.add(Double.parseDouble(data[3]));
    }
    input.close();
  }
}
```

- Create a print method that will print out the data in the ArrayLists in a formatted way. For example:

```
ID: 1
First Name: John
Last Name: Craft
Account Balance: 5561.1
```

```
public static void printData(ArrayList<Integer> ids, ArrayList<String>
firstNames, ArrayList<String> lastNames, ArrayList<Double>
accountBalances) {
  for (int i = 0; i < ids.size(); i++) {
    System.out.println("ID: " + ids.get(i));
    System.out.println("First Name: " + firstNames.get(i));
    System.out.println("Last Name: " + lastNames.get(i));
    System.out.println("Account Balance: " + accountBalances.get(i));
    System.out.println();
  }
}
```

- Create a method to find the person with the highest account balance and print out their information.

```
public static void findPersonWithHighestBalance(ArrayList<Integer> ids,
ArrayList<String> firstNames, ArrayList<String> lastNames,
ArrayList<Double> accountBalances) {
  double maxBalance = accountBalances.get(0);
  int maxIndex = 0;
  for (int i = 1; i < accountBalances.size(); i++) {
    if (accountBalances.get(i) > maxBalance) {
      maxBalance = accountBalances.get(i);
      maxIndex = i;
    }
  }
  System.out.println("Person with the highest account balance:");
  System.out.println("ID: " + ids.get(maxIndex));
```

```
  System.out.println("First Name: " + firstNames.get(maxIndex));
  System.out.println("Last Name: " + lastNames.get(maxIndex));
  System.out.println("Account Balance: " + accountBalances.get(maxIndex));
}
```

- Create a method to find the person with the lowest account balance and print out their information.

```
public static void findPersonWithLowestBalance(ArrayList<Integer> ids,
ArrayList<String> firstNames, ArrayList<String> lastNames,
ArrayList<Double> accountBalances) {
  double minBalance = accountBalances.get(0);
  int minIndex = 0;
  for (int i = 1; i < accountBalances.size(); i++) {
    if (accountBalances.get(i) < minBalance) {
      minBalance = accountBalances.get(i);
      minIndex = i;
    }
  }
  System.out.println("Person with the lowest account balance:");
  System.out.println("ID: " + ids.get(minIndex));
  System.out.println("First Name: " + firstNames.get(minIndex));
  System.out.println("Last Name: " + lastNames.get(minIndex));
  System.out.println("Account Balance: " + accountBalances.get(minIndex));
}
```

- Write to a file called `out.csv` the ID, first name, last name, and account balance of the person with the highest and lowest account balance.

```
public static void writeToFile(String fileName, ArrayList<Integer> ids,
ArrayList<String> firstNames, ArrayList<String> lastNames,
ArrayList<Double> accountBalances, int index) throws IOException {
  File file = new File(fileName);
  PrintWriter writer = new PrintWriter(file);
  writer.println("ID: " + ids.get(index));
  writer.println("First Name: " + firstNames.get(index));
  writer.println("Last Name: " + lastNames.get(index));
  writer.println("Account Balance: " + accountBalances.get(index));
  writer.close();
}
```

- Add a method that will allow the user to search for a person by their last name. If the person is found, print out their information. If the person is not found, print out a message saying that the person was not found.

- **Hint** You can use the `equals` method to check if a `String` equals another `String`.

```
public static void main(String[] args) throws IOException {
  ArrayList<Integer> ids = new ArrayList<>();
  ArrayList<String> firstNames = new ArrayList<>();
  ArrayList<String> lastNames = new ArrayList<>();
  ArrayList<Double> accountBalances = new ArrayList<>();

  File file = new File("in.csv");
  Scanner file = new Scanner(file);

  while (file.hasNextLine()) {
        String line = file.nextLine();
        String[] data = line.split(", ");
        ids.add(Integer.parseInt(data[0]));
        firstNames.add(data[1]);
        lastNames.add(data[2]);
        accountBalances.add(Double.parseDouble(data[3]));
    }
  Scanner input = new Scanner(System.in);
  System.out.print("Enter last name to search: ");
  String lastName = input.nextLine();
  searchPersonByLastName(ids, firstNames, lastNames, accountBalances,
lastName);
}

public static void searchPersonByLastName(ArrayList<Integer> ids,
ArrayList<String> firstNames, ArrayList<String> lastNames,
ArrayList<Double> accountBalances, String lastName) {
  boolean found = false;
  for (int i = 0; i < lastNames.size(); i++) {
    if (lastNames.get(i).equals(lastName)) {
      System.out.println("Person found:");
      System.out.println("ID: " + ids.get(i));
      System.out.println("First Name: " + firstNames.get(i));
      System.out.println("Last Name: " + lastNames.get(i));
      System.out.println("Account Balance: " + accountBalances.get(i));
      found = true;
      break;
    }
  }
  if (!found) {
    System.out.println("Person not found.");
  }
}
```

- Write the Program so you are using one `ArrayList` with a `string` array for each line of the file. Then you can use the `split` method to separate the data into the correct `ArrayLists`.

```
public static void main(String[] args) throws IOException{
  ArrayList<String[]> data = new ArrayList<>();
```

```
  File file = new File("in.csv");
  Scanner file = new Scanner(file);

  while (file.hasNextLine()) {
    String line = file.nextLine();
    String[] splitData = line.split(", ");
    data.add(splitData);
  }

}
```

- Add a method to print an entry by the Id.

```
public static void printEntryById(ArrayList<String[]> data, int id) {
  for (int i = 0; i < data.size(); i++) {
    if (Integer.parseInt(data.get(i)[0]) == id) {
      System.out.println("ID: " + data.get(i)[0]);
      System.out.println("First Name: " + data.get(i)[1]);
      System.out.println("Last Name: " + data.get(i)[2]);
      System.out.println("Account Balance: " + data.get(i)[3]);
      break;
    }
  }
}
```

- Now pass the `ArrayList` of `String[]` to the methods that return the ID of the person with the highest and lowest account balance.

```
public static void findPersonWithHighestBalance(ArrayList<String[]> data)
{
  double maxBalance = Double.parseDouble(data.get(0)[3]);
  int maxIndex = 0;
  for (int i = 1; i < data.size(); i++) {
    if (Double.parseDouble(data.get(i)[3]) > maxBalance) {
      maxBalance = Double.parseDouble(data.get(i)[3]);
      maxIndex = i;
    }
  }
  return maxIndex;
}

public static void findPersonWithLowestBalance(ArrayList<String[]> data) {
  double minBalance = Double.parseDouble(data.get(0)[3]);
  int minIndex = 0;
  for (int i = 1; i < data.size(); i++) {
    if (Double.parseDouble(data.get(i)[3]) < minBalance) {
```

```
      minBalance = Double.parseDouble(data.get(i)[3]);
      minIndex = i;
    }
  }
  return minIndex;
}
```

- Now put together the full program that loads `in.csv` into an `ArrayList` of `String[]`, finds the person with the highest and lowest account balance, returns there Id, and passes it to the `printEntryById` method. Then add a search function where the user can enter a last name and the program will print out the information of the person with that last name.

```
public static void main(String[] args) {
  ArrayList<String[]> data = new ArrayList<>();

  File file = new File("in.csv");
  Scanner file = new Scanner(file);

  while (file.hasNextLine()) {
    String line = file.nextLine();
    String[] splitData = line.split(", ");
    data.add(splitData);
  }

  int maxIndex = findPersonWithHighestBalance(data);
  int minIndex = findPersonWithLowestBalance(data);

  printEntryById(data, maxIndex);
  printEntryById(data, minIndex);

  Scanner input = new Scanner(System.in);
  System.out.print("Enter last name to search: ");
  String lastName = input.nextLine();
  searchPersonByLastName(data, lastName);
}

public static void searchPersonByLastName(ArrayList<String[]> data, String
lastName) {
  boolean found = false;
  for (int i = 0; i < data.size(); i++) {
    if (data.get(i)[2].equals(lastName)) {
      System.out.println("Person found:");
      System.out.println("ID: " + data.get(i)[0]);
      System.out.println("First Name: " + data.get(i)[1]);
      System.out.println("Last Name: " + data.get(i)[2]);
      System.out.println("Account Balance: " + data.get(i)[3]);
      found = true;
      break;
    }
  }
```

```
    if (!found) {
      System.out.println("Person not found.");
    }
  }
```