Lab Name: Computing Statistics - 40 points, 6 extra points

Lab description: Review with Class, ArrayList and Coding Algorithms

Resources: All files are located here, and they are also copied to the bottom of this document. it is your own choice to pick out IDE (BlueJ, Replit<links are available on page 5>, or any Java editors/compiler environment)

Rubrics:
Part 1: 10 points (complete Loan Class with loan.java and test it out with LoadTester.java)
Part 2: 20 points (Level1 & Level 2, complete ComputingStatistics class and run the result with ComputingStatisticsRunner.java and kiva_loans_small.csv (in the same location as the tester file).
Part 3: 10 points (video presentation of the running of the code with precise explanation, under 5 minutes)
Extra Credit:
Part 4: 6 points (Level 3 from part 2, calculate variance, standard deviation and empirical rules)

ArrayLists:

This lab focuses on experimenting with the ArrayList data structure and features a brief review of basic object-oriented programming.

Previously, you have used arrays to store and manipulate collections of primitives and objects. However, Java arrays can only represent collections of a fixed size. This is an unfortunate limitation as the programmer often does not know how many items will be added in advance.

Many programming languages, including Java, provide programmers with data structures that are resizable as well as random access and contiguous. In Java, this data structure is ArrayList. ArrayList internally maintains an array of data that is resized to accommodate additional elements. ArrayList also allows the programmer to add/remove items at arbitrary indices and even find the index of a particular item.

Kiva is an international nonprofit, founded in 2005 and based in San Francisco, with a mission to connect people through lending to alleviate poverty. We celebrate and support people looking to create a better future for themselves, their families, and their communities. By lending as little as $25 on Kiva, anyone can help a borrower start or grow a business, go to school, access clean energy or realize their potential. For some, it's a matter of survival, for others it's the fuel for a life-long ambition. We will use some Kiva data that contains some data that we will use to practice some basic descriptive statistics that are commonly used in data science.

Part 1: Loan Class (10 points)

1. Create a new Java class and name the file Loan.java
2. Add the instance variables listed below to the class (make sure the access modifier is private):

Data type	Variable name	Description
`int`	`id`	The loan ID number
`double`	`loanAmount`	The value of the loan
`String`	`country`	The country the loan is from
`int`	`daysToFund`	The number of days it took to fund the loan
`int`	`numLenders`	The number of lenders for the loan

3. Add the initialization constructor to the class:

```
public Loan(int i, double l, String c, int d, int n) {
    id = i;
    loanAmount = l;
    country = c;
    daysToFund = d;
    numLenders = n;
}
```

4. Add the accessor methods listed below to the class (make sure the access modifier is public):

Return type	Method name	Description
`int`	`getId`	Returns the loan Id
`double`	`getLoanAmount`	Returns the loan amount
`String`	`getCountry`	Returns the country the loan is from
`int`	`getDaysToFund`	Returns the number of days it took to fund
`int`	`getNumLenders`	Returns the number of lenders
`String`	`toString`	Returns a string representation of the loan

5. Add the modifier methods listed below to the class (make sure the access modifier is public):

Return type	Method name	Parameter name	Description
`void`	`setId`	`id`	Sets the loan Id
`void`	`setLoanAmount`	`loanAmount`	Sets the loan amount
`void`	`setCountry`	`country`	Sets the country the loan is from
`void`	`setDaysToFund`	`daysToFund`	Sets the number of days it took to fund
`void`	`setNumLenders`	`numLenders`	Sets the number of lenders

6. Test your Loan.java file using the LoanTest.java file. (Uncomment all the testing code)
    a. Fix any errors as this will be used in the next part of the lab.

Output:

```
Id: 12345 loan amount: 280.0 country: Japan days to fund: 5 num lenders:
8
Id: 12345 loan amount: 280.0 country: Japan days to fund: 5 num lenders:
8
54321 == 54321
450.0 == 450.0
USA == USA
12 == 12
4 == 4
Id: 54321 loan amount: 450.0 country: USA days to fund: 12 num lenders:
4
Id: 54321 loan amount: 450.0 country: USA days to fund: 12 num lenders:
4
```

Part 2: ComputingStatistics Class (20 points)

1. Open the ComputingStatistics.java file
2. The instance variable, initialization constructor, and first method have been completed
3. Continue answering the questions posed below
4. Each method should use the method header provided
5. After each method, uncomment the line of code in ComputingStatisticsRunner.java and test your method

Level 1 Questions: (10 points in coding)

1. What is the total amount of money loaned?
   a. `public double totalAmount()`
2. What is the average loan amount?
   a. `public double avgLoan()`
   b. Remember you could use what is already defined, i.e. totalAmount().
3. What is the largest loan?
   a. `public double largestLoan()`
   b. Remember you can use `Integer.MIN_VALUE`
4. What is the smallest loan?
   a. `public double smallestLoan()`
   b. Remember you can use `Integer.MAX_VALUE`
5. What country got the largest loan?
   a. `public String largestLoanCountry()`
6. What country got the smallest loan? (output will be either Philippines or Indonesia)
   a. `public String smallestLoanCountry()`

Level 2 Questions: (10 points in coding)

1. What is the average number of days needed to fund a loan?
   a. `public double avgDaysToFund()`
2. What was the largest loan made to people in Kenya? **(You do: include a test for this method in the runner file, do you know how to do it in the runner/tester?)**
   a. `public double largestLoanKenya()`
   b. Remember you can use `Integer.MIN_VALUE`
3. What is the average amount of loans made to people in the Philippines?
   a. `public double avgLoanPhilippines()`
4. In which country was the loan granted that took the longest to fund?
   a. `public String longestToFundCountry()`
5. Does El Salvador or Kenya have more loans funded?
   a. `public String mostLoansFunded()`

Level 3 Questions: (Extra Credit: 6 points, 2 each)

For our final few questions, we are interested in exploring the distribution of the data. To do this we need to introduce a few more statistical concepts including variance and standard deviation.

Variance looks at a single variable and measures how far the set of numbers are spread out from their average value. However, it is a bit hard to interpret because the units are squared so it is not on the same scale as our original numbers. This is why most of the time we use the standard deviation, which is just the square root of the variance. A large standard deviation tells us that our data is quite spread out while a small standard deviation tells us that most of our data is pretty close to the mean.

$$variance = \frac{\sum (x - \bar{x})^2}{n}$$

$$stdev = \sqrt{variance}$$

Do not let the fancy math get you down, the variance is the sum of the squared values of each value minus the average for that value divided by the number of values. This website has a good explanation of variance and standard deviation.

1. What is the variance of the money loaned?
   a. `public double variance()`
2. What is the standard deviation of the money loaned?
   a. `public double standardDeviation()`
3. The Empirical Rule or 68-95-99.7% Rule reminds us that 68% of the population falls within 1 standard deviation. Does this hold for our data?
   a. `public boolean empiricalRule()`

Sample output (kiva_loans_small):

```
Total amount: 1.0778625E7
Average loan amount: 866.8670580665917
Largest loan amount: 12925.0
Smallest loan amount: 50.0
The country with the largest loan amount: India
The country with smallest loan amount: Philippines
Average days to fund loan request: 11.461476596429145
The largest loan funded in Kenya: 5875.0
Average loan amount in the Philippines: 325.30146425495263
The country with the longest to fund loan: Kenya
Variance: 1075599.4194342843
Standard deviation: 1037.1110931015464
Empirical rule (yes/no): true
```

Now replit project links are available, use them at your own risk
Link made by Payton and Brody
Link made by Deven

```
//start of LoanTester.java--------------------
public class LoanTester {
  public static void main(String[] args) {
//      Loan testOne = new Loan(12345, 280.00, "Japan", 5, 8);
//      System.out.println(testOne);
//      System.out.println("ID: 12345 loan amount: 280.0 country: Japan days to fund: 5 num lenders: 8");
//      testOne.setId(54321);
//      System.out.println(testOne.getId() + " == 54321");
//      testOne.setLoanAmount(450.00);
//      System.out.println(testOne.getLoanAmount() + " == 450.0");
//      testOne.setCountry("USA");
//      System.out.println(testOne.getCountry() + " == USA");
//      testOne.setDaysToFund(12);
//      System.out.println(testOne.getDaysToFund() + " == 12");
//       testOne.setNumLenders(4);
//       System.out.println(testOne.getNumLenders() + " == 4");
//       System.out.println(testOne);
//       System.out.println("ID: 54321 loan amount: 450.0 country: USA days to fund: 12 num lenders: 4");
  }
}

//start of ComputingStatisticsRunner-------------
import java.util.*;
import java.io.*;
import java.text.*;

public class ComputingStatisticsRunner {
  public static void main(String[] args) throws FileNotFoundException {
    // Specifies the data file to be used.
    String file = "kiva_loans_small.csv";

    // Creates an ArrayList to store the data and calls the readData() method.
    ArrayList<Loan> list = readData(file);

    // Loop to print out the existing data to see how it is structured.
    //for(int i = 0; i < list.size(); i++) {
    //   System.out.println(list.get(i));
    //}

    //Use this code to test the methods in the ComputingStatistics class
    ComputingStatistics analysis = new ComputingStatistics(list);
    System.out.println("Total amount: " + analysis.totalAmount());

    //System.out.println("Average loan amount: " + analysis.avgLoan());
    // System.out.println("Largest Loan: " + analysis.largestLoan());
    // System.out.println("Smallest Loan: " + analysis.smallestLoan());
    // System.out.println("Largest Loan Country: " + analysis.largestLoanCountry());
    // System.out.println("Smallest Loan Country: " + analysis.smallestLoanCountry());
    // System.out.println("Average days to fund: " + analysis.avgDaysToFund());
    //System.out.println("Average loan amount in the Philippines: "+ analysis.avgLoanPhilippines());
    //System.out.println("The country with the longest to fund loan:" + analysis.longestToFundCountry());
    // System.out.println("Variance: " + analysis.variance());
    // System.out.println("Standard Deviation: " + analysis.standardDeviation());
    // System.out.println("Empirical Rules (yes/no): " + analysis.empiricalRule());

  }

//Start of ComputeStatisticsRunner.java   ------------------------
  /**
   * Reads in the provided file and creates an ArrayList of the data.
   * @param file the name of the text file containing the data.
   * @return the ArrayList containing the data from the text file.
   */
  public static ArrayList<Loan> readData(String file) throws FileNotFoundException {
    // Scanner used to read in the data from the file.
    Scanner in = new Scanner(new File(file));
    // ArrayList to store the data.
```

```
      ArrayList<Loan> list = new ArrayList<Loan>();
      // Read in the header line so it is not added to the ArrayLists.
      String header = in.nextLine();
      // Check to see if the file still has data to be read in.
      while(in.hasNextLine()) {

        SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");

        // Read in the line of data and
          // use a space as a delimiter to separate the different columns.
        String[] line = in.nextLine().split(",");

        // Local variable containing the ID.
        int ID = Integer.parseInt(line[0]);

        // Local variable containing the amount.
        int amount = Integer.parseInt(line[2]);

        // Local variable containing the country.
        String country = line[5];

        // Local variable containing the lenders.
        int lenders = Integer.parseInt(line[11]);

        // Local variable containing the difference in days.
        int differenceInDays = -1;

        try {
          Date postedDate = sdf.parse(line[8]);

          Date fundedDate = sdf.parse(line[9]);

          long differenceInTime = fundedDate.getTime() - postedDate.getTime();

          differenceInDays = (int)((differenceInTime / (1000 * 60 * 60 * 24)) % 365);
        }

       // Catch the Exception
       catch (ParseException e) {
          e.printStackTrace();
       }
       // Add the loan to the arraylist.
       list.add(new Loan(ID, amount, country, differenceInDays, lenders));

      }
      // Return the completed ArrayLists.
      return list;
    }

}

//Start of ComputeStatistics.java---------
import java.util.ArrayList;
```

```
public class ComputingStatistics {
  /**
  * The ArrayList containing all of the loan data.
  */
  private ArrayList<Loan> data;

  /**
   * Creates a new ComputingStatistics object with an empty ArrayList
   */
  public ComputingStatistics() {
    data = new ArrayList<Loan>();
  }

  /**
   * Creates a new ComputingStatistics object with the data passed in
   */
  public ComputingStatistics(ArrayList<Loan> d) {
    data = d;
  }

  /**
   * Calculates the total amount funded from all of the loans in the file.
   * @return the total loan amount.
   */
  public double totalAmount() {
    double amount = 0.0;
    for(int i = 0; i < data.size(); i++) {
      amount = amount + data.get(i).getLoanAmount();
    }
    return amount;
  }
}
```

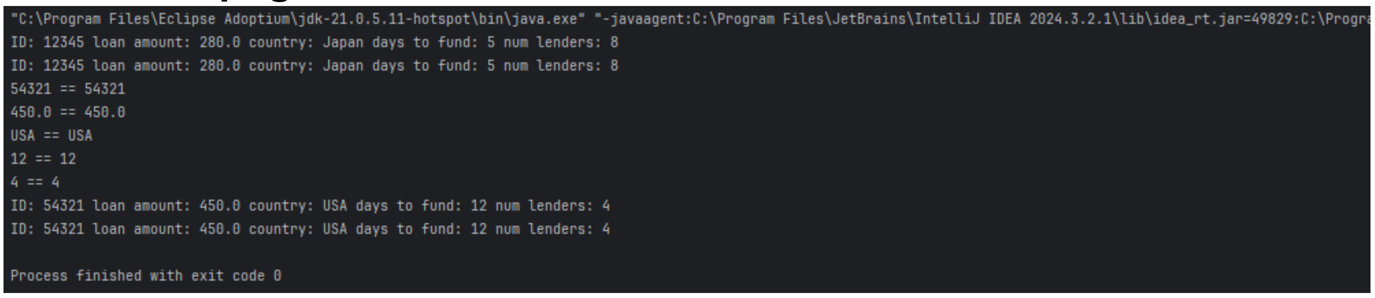
Fill in information below

Your Name:

Christopher Mustard

Copy/Paste your running output snapshots below:

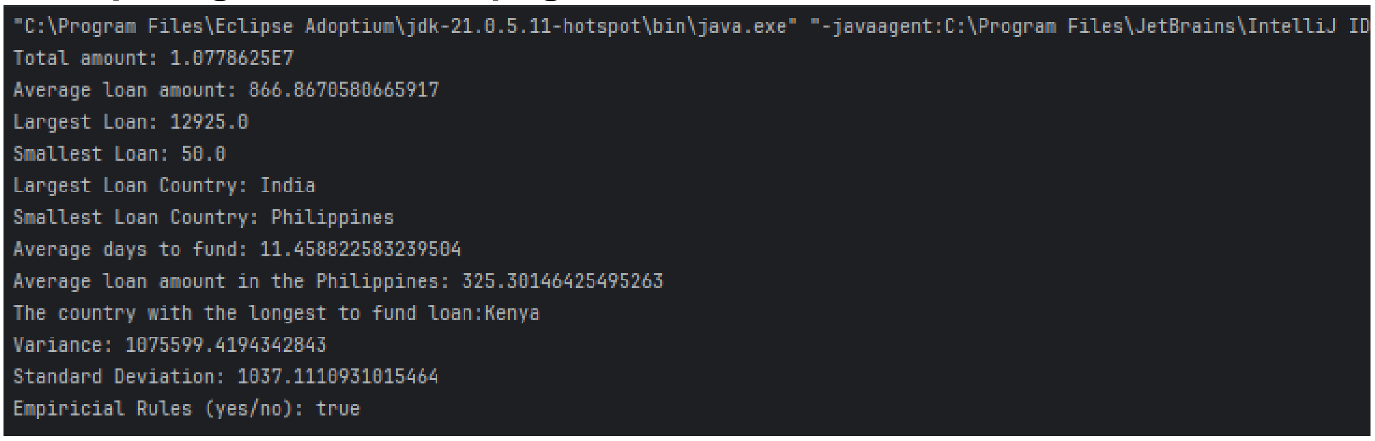
./Loan Test.png



```
"C:\Program Files\Eclipse Adoptium\jdk-21.0.5.11-hotspot\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2024.3.2.1\lib\idea_rt.jar=49829:C:\Progra
ID: 12345 loan amount: 280.0 country: Japan days to fund: 5 num lenders: 8
ID: 12345 loan amount: 280.0 country: Japan days to fund: 5 num lenders: 8
54321 == 54321
450.0 == 450.0
USA == USA
12 == 12
4 == 4
ID: 54321 loan amount: 450.0 country: USA days to fund: 12 num lenders: 4
ID: 54321 loan amount: 450.0 country: USA days to fund: 12 num lenders: 4

Process finished with exit code 0
```

./ComputingStatistic Test.png



```
"C:\Program Files\Eclipse Adoptium\jdk-21.0.5.11-hotspot\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ ID
Total amount: 1.0778625E7
Average loan amount: 866.8670580665917
Largest Loan: 12925.0
Smallest Loan: 50.0
Largest Loan Country: India
Smallest Loan Country: Philippines
Average days to fund: 11.458822583239504
Average loan amount in the Philippines: 325.30146425495263
The country with the longest to fund loan:Kenya
Variance: 1075599.4194342843
Standard Deviation: 1037.1110931015464
Empiricial Rules (yes/no): true
```

Please insert the link to all source codes (must be shared) or copy/paste the original code here if you use BlueJ(only java files):

./src (My Code)
./lib (Provided Code)

Your explanation in video here (must be shared, key terms must be worded out and explained with details, no more than 5 minutes long):

./explanation.png

Please enter the resource from your code!!! (links to the idea or code you have borrowed)

./lib (Provided Code)

Type in your reflection here, for example, did you run into any issues/bugs, how did you solve them? What did you learn from this project, any suggestions for making this project better?

N/A

Do you deserve the extra credits? Why?

I was able to complete the extra credit questions. I documented my code in a professional manner. This project was well organized!