## Lab Name: Computing Statistics with Kiva Data

### Credit:

### 

Thanks to the creators of/contributors to the Foundations of Python Programming book (Brad Miller, Paul Resnick, Lauren Murphy, Jeffrey Elkner, Peter Wentworth, Allen B. Downey, Chris)

### Lab description:

#### 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:

[Kiva](http://kiva.org/) 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

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 |

1. Add the initialization constructor to the class:

|  |
| --- |
| **public Loan(int ID, double loanAmount, String country, int daysToFund,**  **int numLenders) {**  **this.ID = ID;**  **this.loanAmount = loanAmount;**  **this.country = country;**  **this.daysToFund = daysToFund;**  **this.numLenders = numLenders;**  **}** |

1. 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 |

1. 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 |

1. Test your Loan.java file using the LoanTest.java file.
   1. 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

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:

1. What is the total amount of money loaned?
   1. public double totalAmount()
2. What is the average loan amount?
   1. public double avgLoan()
3. What is the largest loan?
   1. public double largestLoan()
   2. Remember you can use Integer.MIN\_VALUE
4. What is the smallest loan?
   1. public double smallestLoan()
   2. Remember you can use Integer.MAX\_VALUE
5. What country got the largest loan?
   1. public String largestLoanCountry()
6. What country got the smallest loan?
   1. public String smallestLoanCountry()

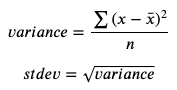
##### Level 2 Questions:

1. What is the average number of days needed to fund a loan?
   1. public double avgDaysToFund()
2. What was the largest loan made to people in Kenya?
   1. public double largestLoanKenya()
   2. Remember you can use Integer.MIN\_VALUE
3. What is the average amount of loans made to people in the Philippines?
   1. public double avgLoanPhilippines()
4. In which country was the loan granted that took the longest to fund?
   1. public String longestToFundCountry()
5. Does El Salvador or Kenya have more loans funded?
   1. public String mostLoansFunded()

##### Level 3 Questions:

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.



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](https://www.mathsisfun.com/data/standard-deviation.html) has a good explanation of variance and standard deviation.

1. What is the variance of the money loaned?
   1. public double variance()
2. What is the standard deviation of the money loaned?
   1. 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?
   1. 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 |

#### Resources:

[Computing Statistics with Kiva Data source](https://runestone.academy/runestone/books/published/fopp/Projects/kiva_statistics.html)

[Kiva data source](https://www.kaggle.com/gcdatkin/kiva-loan-type-prediction/)

[GitHub Starter Repository](https://github.com/ASIJ-Mustangs/ComputingStatistics)