**1.2.5 Name Generator - Math, Strings, Casting**

**Introduction**

Our goal is to create a comical ‘name generator’ like the ones at the link below. The name generator will gather input such as your first and last name and return a unique and crazy name based on what you give it.

Concepts covered are:

* Scanner methods
* Static methods
* String class
* index
* substring
* Java Math class
* Math.random()
* Type casting
* Keywords: **import**, **static**,
* Java classes: java.util.Scanner, Math

**Materials**

* Computer with BlueJ

**Activity**

**Part I: String Class Methods**

1. This project will be done with partners - so grab a partner!
2. Explore his link together: [**http://www.namegeneratorfun.com/**](http://www.namegeneratorfun.com/)and find a name generator that is amusing and fun for you. Each generator uses your name as input and generates a pseudo-random name. Choose one and see if you can figure out how it works.
3. Open BlueJ and create a new project called NameGenerator in your Projects folder. Create a new class called NameGenerator. Fill out your Javadoc header for the class.
4. First we need to create a method that gathers user input and stores it in variables. We will use a Scanner object like we did in previous assignments. We need to import Scanner methods at the top of our code.

|  |
| --- |
| * Above the class write your import statement: import java.util.Scanner; * In order to abstract gathering user input we will create a **static** method called getInput() that will ask users for input and store the input. The input will be in the form of a String that the method will return. Make this method. * Make a new Scanner object using this format:   Scanner input = new Scanner(System.in);   * Add this line next. This is the line that gathers String input and returns it:   return input.next(); |

1. Next you need to create an introduction to your NameGenerator class that has some flavor and gives instructions to the user. Since we are only creating one class for this assignment we can use a main() method and have it ask questions and call the getInput() function.

|  |
| --- |
| * Create a main() method. Hint: PSVM * Print several lines for the user explaining what the name generator will do. To print the syntax is System.out.println(String output) * Here's where we ask our first question. Each time you ask a question, you will need to call the getInput() function and store the output in a variable. The code looks like this but you'll have your own variation:      * You'll want to gather:   + First name   + Last name   + City you were born in   + A previous school you attended   + The first name of a relative |

1. Now we need to write a method to manipulate the input strings and generate a new name. We'll need to know more about the String class and methods. Take a look at the String class APIs to get a feel for what you can do with strings:

[**https://docs.oracle.com/javase/7/docs/api/java/lang/String.html**](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html)

|  |
| --- |
| * Create a new method called genFirstName() that returns a String object (your new first name) and takes your first and last name variables as parameters. |

Note: If you want to compare Strings you can't just do an ==.... Here's a link to a tutorial on comparing String object contents:

[**https://docs.oracle.com/javase/tutorial/java/data/comparestrings.html**](https://docs.oracle.com/javase/tutorial/java/data/comparestrings.html)

1. Inside this method we will use two important String class static methods to generate a new first name: String.substring(), String.indexOf(), and String.length().

String.indexOf() returns the index, or position, of the character you pass it as an argument. For example is firstName = "Anthony" then firstName.indexOf('A') returns 0. The key is remembering that indexes start at 0. Here's firstname:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **String:** | A | n | t | h | o | n | y |
| **Index:** | 0 | 1 | 2 | 3 | 4 | 5 | 6 |

String.substring() is an **overloaded** method that can take 1 or 2 arguments. The arguments are a starting index and (optionally) an ending index. If no ending is specified it will return everything from the starting index to the end of the string. This portion of the string is called a **substring**. For example firstName.substring(2) returns "thony".

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **String:** | A | n | t | h | o | n | y |
| **Index:** | 0 | 1 | 2 | 3 | 4 | 5 | 6 |

If an ending index is specified it is not included. For example: firstName.substring(2,6) returns "thon"

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **String:** | A | n | t | h | o | n | y |
| **Index:** | 0 | 1 | 2 | 3 | 4 | 5 | 6 |

String.length() takes no arguments and returns the number of characters in the string. For example firstName.length() returns 7. Notice this is one greater than the greatest index number because indexes start at 0.

|  |
| --- |
| * Using the three methods described above, get the first threeletters of your first name and the first two letters of your last name and combine them to generate your first name. * Test your code. * Modify your genFirstName() method to create a new first name using some other algorithm of your choosing! * Now create a genLastName() method to create a new last name. This will take the city you were born in and a previous school as parameters and return a new last name. * Get the first two letters of the city you were born in. Get the get the first three letters of your previous school. Using concatenation, combine these results to generate your Sci-Fi last name. * Test your code. * Modify your genLastName() method to create a new last name using some other algorithm of your choosing! |

**Part II: Java Math Class**

1. Now we will generate a random place of origin for you. Random numbers are generated using the Math class, which like the String class, contains only **static methods** and is included in Java standard libraries. It does not need to be imported. We've used static methods but let's get a formal definition at this point:
   1. Static methods ***belong to the class*** 
      * Use the **static** keyword that follows the access modifier in the method signature.
      * They are NOT inherited by objects instantiated from the class.
      * They are meant to be utilities and not meant to define an object or change it’s behavior.
      * They are called on the class.
   2. Regular methods ***belong to the object***
      * They are inherited by objects instantiated by the class.
      * They are meant to alter the behavior of objects.
      * They are called on the object.

The Math class has several methods that you can look up. Take a look here and explore a bit:

[**https://docs.oracle.com/javase/8/docs/api/java/lang/Math.html**](https://docs.oracle.com/javase/8/docs/api/java/lang/Math.html)

We will be using the random() method, which returns a randomly generated **double** from 0 to 0.999999999999999. Remember that a **double** is a **float** that can be twice as large. Usually we don't want a number in the range that Math.random() gives, so we need to scale and translate the value generated.

We will type-cast (or simply cast) the double as an integer. This will convert the value and round in the process.

Casting follows this format:

type newVariable (type to cast as) variableName

Example: int num = (int) string;

Here's the whole process of generating a random integer from 1 to 50:

int x =(int) (Math.random()\*50) + 1

|  |
| --- |
| * Create a new method called genOrigin() that takes the first name of a relative and a friend's name as parameters and returns a place of origin. * Generate a random number between 1 and length – 1 of a relative’s first name. You can generate a random number using this format: * Get the last letters of your relative’s first name beginning at the random location through to the end of the string. * Generate a random number between 1 and (length – 1) of another relative’s name. You may choose to use a friend's name in place of a relative's name. * Get the last letters of this name beginning at the random location through to the end of the string. * Using concatenation (string addition), combine the results to generate your Sci-Fi place of origin. * Explore the String methods and find one that will capitalize the first letter of your new names and place of origin. * Print a friendly message such as:   "Hello Carki Chsal of Lesomas. Welcome!" |

**Conclusion**

1. Review the online documentation regarding [**Random Numbers**](http://interactivepython.org/runestone/static/JavaReview/VariableBasics/randomNumbers.html) and complete the “Check your understanding” exercise. How would you call or invoke a **static method** called sqrt(25) that is part of the Math class?
2. Explain how the following call simulates a dice roll.

int diceRoll = (int)(Math.random() \* 12) + 1;

1. In this activity, you generated random numbers between 1 and the (length – 1) of your first relative’s name. Explain why the algorithm specified this length and describe the pattern that resulted.
2. Explain how the following two statements return different results:

int r1 = (int)(Math.random()) \* 10 + 1;

int r2 = (int)(Math.random() \* 10) + 1;

1. Complete all codingbat String-1 exercises:

[**http://codingbat.com/java/String-1**](http://codingbat.com/java/String-1)