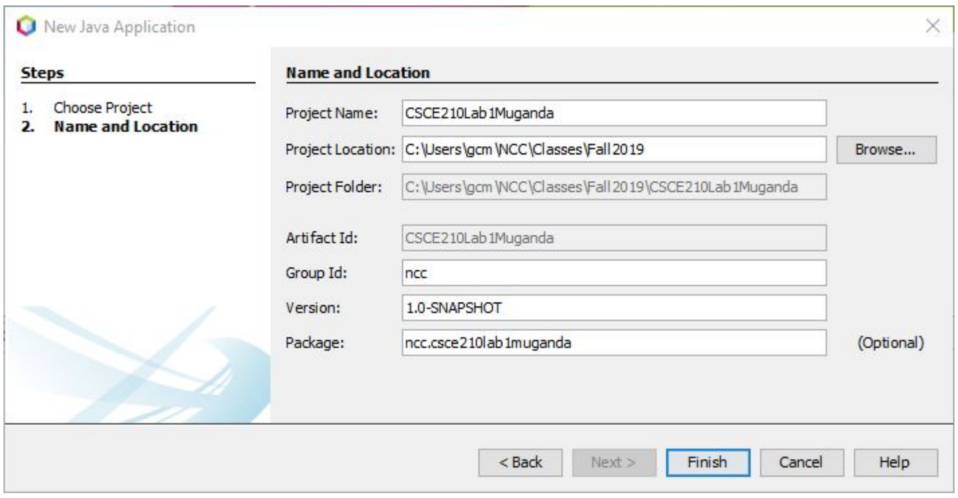
CSCE 210 LAB 1

GODFREY MUGANDA

1. Recursive Reverse Printing

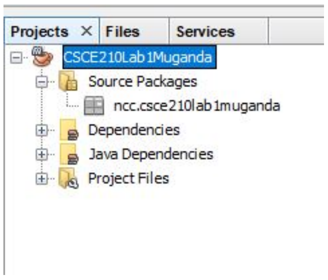
The purpose of this first lab is to get you acquainted with both Java and the Netbeans IDE, and learn how to submit a project for grading.

Open up Netbeans and create a project named CSC210Lab1LastName (use your last name as the suffix afor the project name). In the latest version of Netbeans, the file-creation dialog will look something like this:

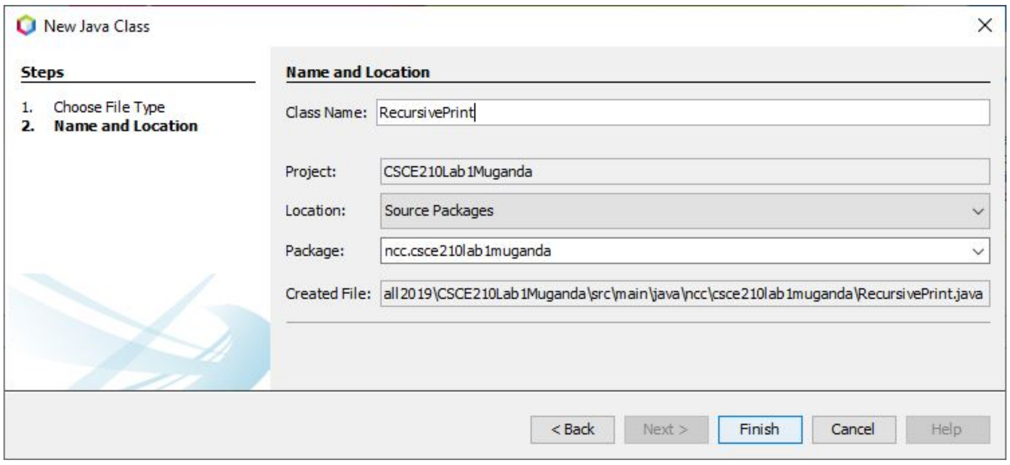


Project Name: CSCE210Lab1Muganda
Project Location: C:\Users\gcm\NCC\Classes\Fall2019
Project Folder: C:\Users\gcm\NCC\Classes\Fall2019\CSCE210Lab1Muganda
Artifact Id: CSCE210Lab1Muganda
Group Id: ncc
Version: 1.0-SNAPSHOT
Package: ncc.csce210lab1muganda (Optional)

Click `Finish` to finish creating the project. Expand the `source packages` node.



Right-click on the package (here, `ncc.csse210Lab1Muganda`) and select `New.. /Java class` to add a Java class called `RecursivePrint` to your project.



Class Name: RecursivePrint
Project: CSCE210Lab1Muganda
Location: Source Packages
Package: ncc.csce210lab1muganda
Created File: all2019\CSCE210Lab1Muganda\src\main\java\ncc\csce210lab1muganda\RecursivePrint.java

When you click `Finish`, the IDE will add a skeleton for a Java class to your code. Delete the useless comments at the top, and you will be left with something that looks like this:

```
package ncc.csce210lab1muganda;

/**
 *
 * @author gcm
 */
public class RecursivePrint
{

}
```

Modify the contents of the file above to look like this:

```
import java.util.Scanner;

public class RecursivePrint
{
    public static void main(String [] args)
    {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter a string: ");
        String str = sc.nextLine();
        System.out.println("Here is the string printed backwards:");

        // print the string backwards with the first recursive method.
        printBackwards1(str, 0);
        System.out.println();

         // print the string backwards with the second recursive method.
        printBackwards2(str, str.length()-1);
    }
    public static void printBackwards1(String str, int lower)
    {

    }

    public static void printBackwards2(String str, int upper)
    {

    }
}
```

Here you have a `main` method that calls on two methods to print a string backward. The two print methods must be recursive. The first one

```
  public static void printBackwards1(String str, int lower)
    {

    }
```

prints the portion `str[lower..str.length()-1]` of the string `str` backwards. Similarly, the second method

```
    public static void printBackwards2(String str, int upper)
    {

    }
```

prints the portion `str[0..upper]` of the string `str` backwards.

When you are done writing the two methods, you should test them by right-clicking on the name of the file and selecting `Run File`. Here is a sample run and output:

```
Enter a string: North Central College
Here is the string printed backwards:
egelloC lartneC htroN
egelloC lartneC htroN
```

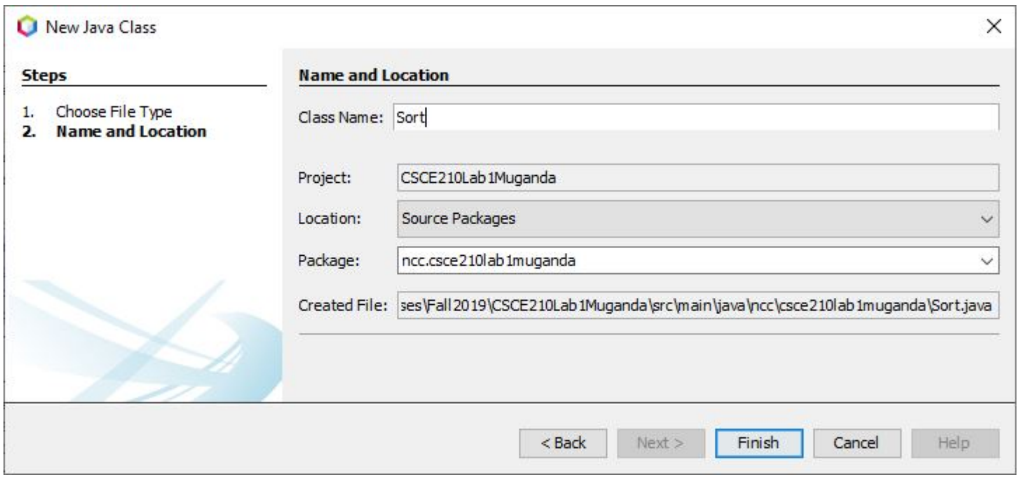
2. Recursive Selection Sorting

Next, we are going to add a file that contains code to enter and sort an array of integers. The sorting will be accomplished by a recursive method

```
void sort(int [] arr, int lower, int upper)
```

which when called, will sort the portion `arr[lower..upper]` of the array `arr` in non-decreasing order.

Begin by right-clicking on the package and selecting `New../ Java Class`, and then add a class `Sort.java`.



Name and Location

Class Name: Sort

Project: CSCE210Lab1Muganda

Location: Source Packages

Package: ncc.csce210lab1muganda

Created File: ses\Fall2019\CSCE210Lab1Muganda\src\main\java\ncc\csce210lab1muganda\Sort.java

Add a main method and stubs for three other methods as follows:

```
public class Sort
{
    public static void main(String []args)
    {
       int size;

       // Read the size of the array to be created
       Scanner sc = new Scanner(System.in);
       System.out.print("Enter the size of the array to sort: ");
       size = sc.nextInt();

       // Create an array of the given size  and fill it with
       // random numbers
       int [] arr = new int[size];
       Random randy = new Random();
       for (int k = 0; k < size; k++)
       {
           arr[k] = randy.nextInt(100);
       }

       // Output the array
       System.out.printf("The random array created is: %s\n", Arrays.toString(arr));

       // Sort the array using the recursive function
       sort(arr, 0, arr.length-1);

       //Output the sorted array.
       System.out.printf("The random array created is: %s\n", Arrays.toString(arr));
       System.out.println();
    }

    /**
     * swaps the values at the positions pos1 and pos2 in the array arr
     * @param arr
     * @param pos1
     * @param pos2
     */
    static void swap(int [] arr, int pos1, int pos2)
    {

    }

    /**
     *
     * @param arr
     * @param lower
     * @param upper
     * @return an array int []{m, M} where m is the position of a minimum value
     *         in the subarray arr[lower..upper], and M is the position of a
     *         maximum value in the same subarray.
     */
    static int [] findMinMaxPositions(int []arr, int lower, int upper)
    {


    }

    /**
     * sorts the portion arr[lower..upper] of an integer array in non-decreasing
     * order.
     * @param arr
     * @param lower
     * @param upper
     */
    static void sort(int [] arr, int lower, int upper)
    {

    }
}
```

The `swap(arr, k, j` method swaps elements `arr[k]` and `arr[j]` of the array.

The method `int [] findMinMaxPositions(int []arr, int lower, int upper)` takes as parameters an array `arr` and two integers that define a range of that array, and return an array with two elements, which are the indices of minimum and maximum values in the subrange of the array. The index of a minimum value is in position 0 of the returned array, and the index of a maximum value is in position 1.

The `sort()` method will use the two other methods.

When you are done, go to the `Run` method on the Netbeans menu and select the `Build Project` item. Once the project has built successfully, right click on the `Sort.java` file item and select `Run file`. Here is a sample input with corresponding output:

```
Enter the size of the array to sort: 10
The random array created is: [26, 80, 5, 95, 12, 62, 42, 81, 77, 49]
The sorted array is: [5, 12, 26, 42, 49, 62, 77, 80, 81, 95]
```

Caution: The case where the array returned by `findMinMaxPos` is {`upper, lower`}, meaning the maximum and minimum values are occupying each other's rightful positions, should be treated as a special case. Why?

3. Submitting the Project for Grading

Locate the entire project folder wherever you stored it, and copy it into your folder on the K drive.

If you are working at home, you may want to zip the project folder before uploading to your K drive space.

4. Due Date

This lab is due Wednesday, August 28, at midnight.