PA1: Series Object

Due: April 10th, Thursday, 11:59 pm

The latest up-to-date **corrections** can be found here.

This assignment is an **individual** assignment. You may ask Professors/TAs/Tutors for some guidance and help, but you can't copy code. You may discuss the assignment conceptually with your classmates, including bugs that you ran into and how you fixed them. However, you need to write your submission solely by yourself.

You are not allowed to import any additional package for this assignment.

This assignment has **1 submission** on Google Form and **1 submission** on CompassX. Make sure you submit all of them.

Part 0: Integrity of Scholarship Agreement (0 points)

Before starting the homework, please carefully read and fill out the integrity of the scholarship agreement form. You will not receive scores in this class until you submit the form.

Part 1: Setup IntelliJ

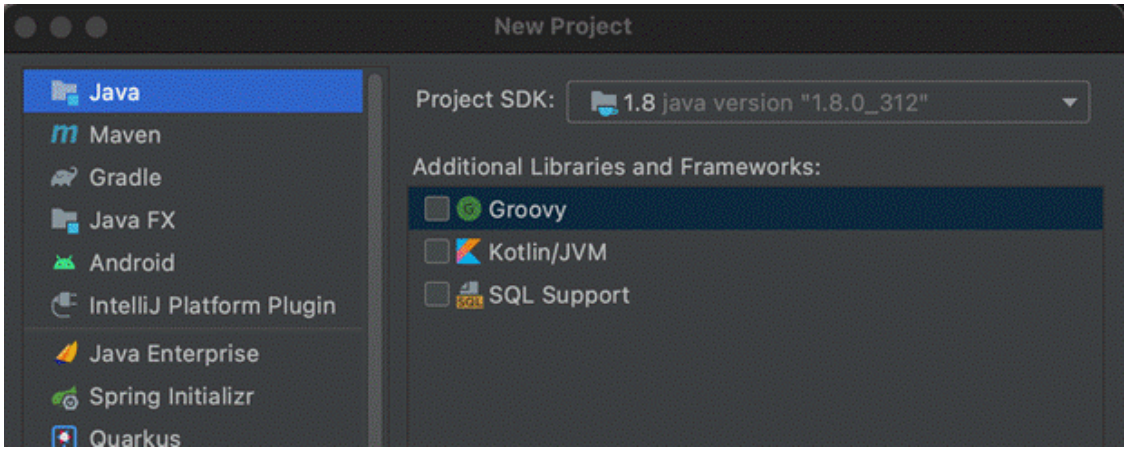
Install IntelliJ **community edition** from JetBrains. You can keep the default settings during the installation.

Part 2: Complete Your First Java Program and Submit on CompassX

Then, follow the steps below to finish your first ever java program. We are using the legacy UI in the following steps as an example. In your local machine, feel free to pick whatever UI you would like.

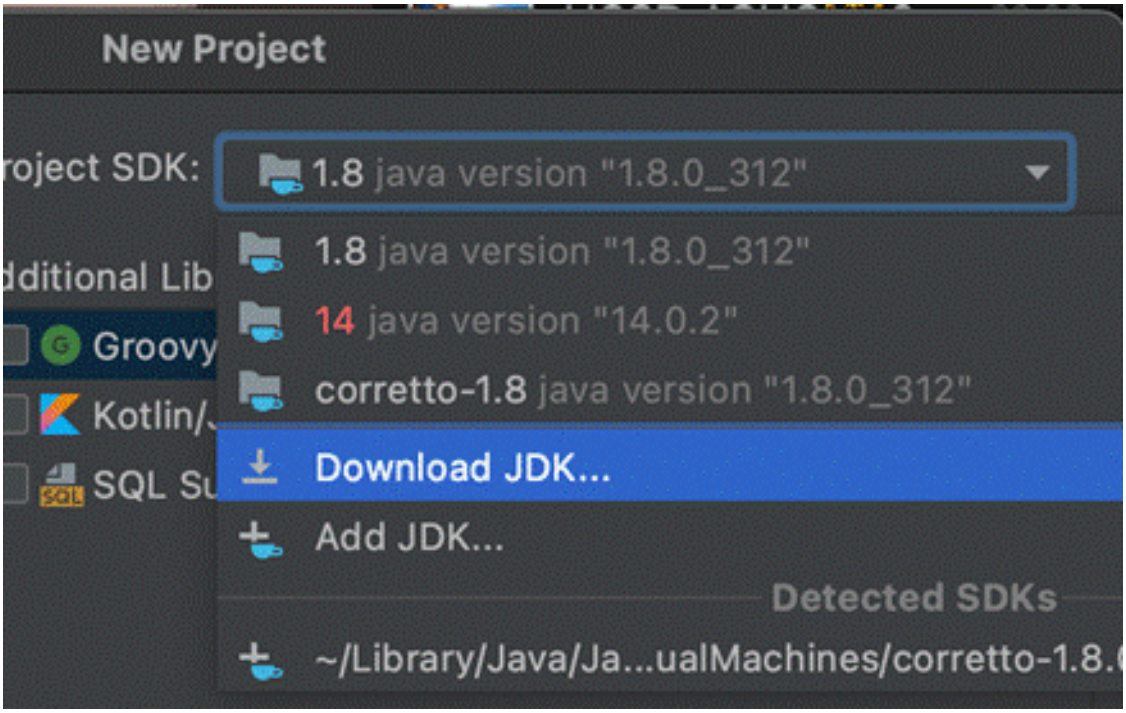
1. Now we are going to create our first Java program using IntelliJ. First, select **Create New Project.**

2. Choose any version higher than 1.8 as your Java SDK.



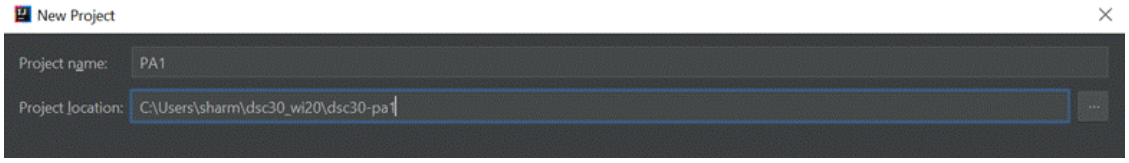
   – If your IntelliJ found the Project SDK for you (displays the java version instead of <No SDK>), ignore this step and click **Next**.

- If you haven't downloaded java on your computer, click download JDK.

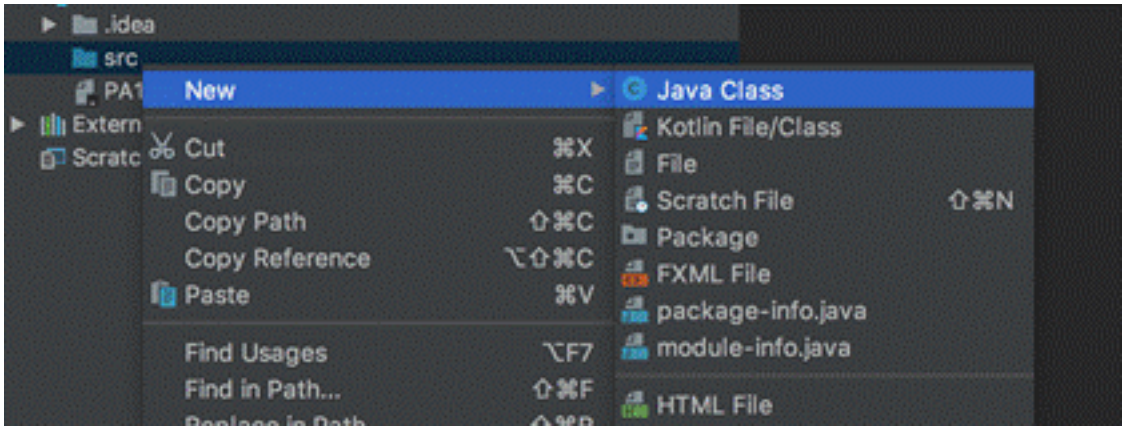


- And choose any version higher than 1.8 as your version. IntelliJ will download the JDK for you.

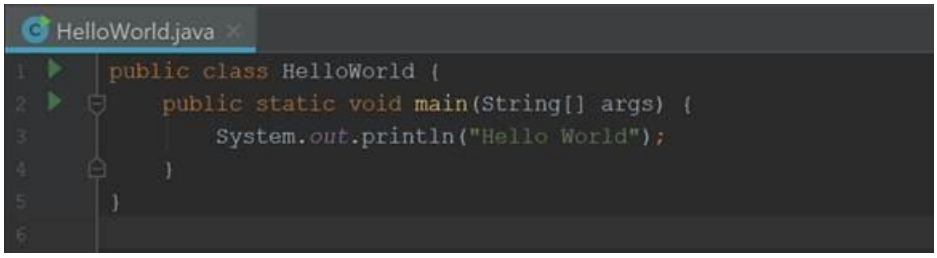
3. You don't need to use a template, so click **Next** again. Name the project **PA1** or **dsc30-pa1** (generally, the name of the project is up to you).



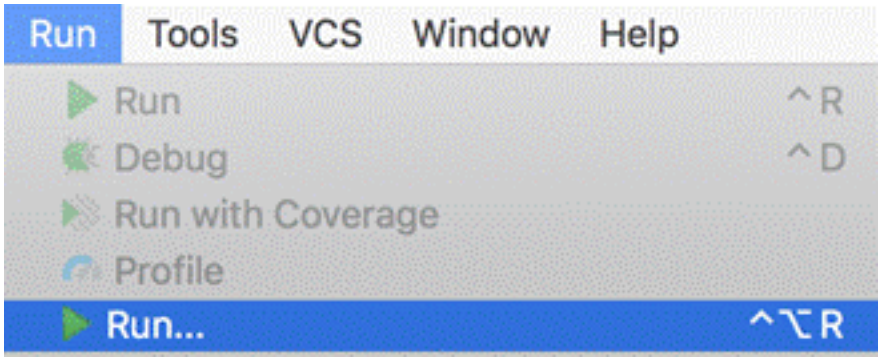
- Pro Tip: Be sure to organize your PA into a folder you know how to find it. Otherwise this project might be lost in your computer when you try to work on it later.
- Now click **Finish** and click **OK** in the pop-up window. Then open up the project view (there should be a shortcut prompt to do so on the screen).
- Now create a Java Class in your **src** (short for source) folder named **HelloWorld**.



4. Now type out the following code:

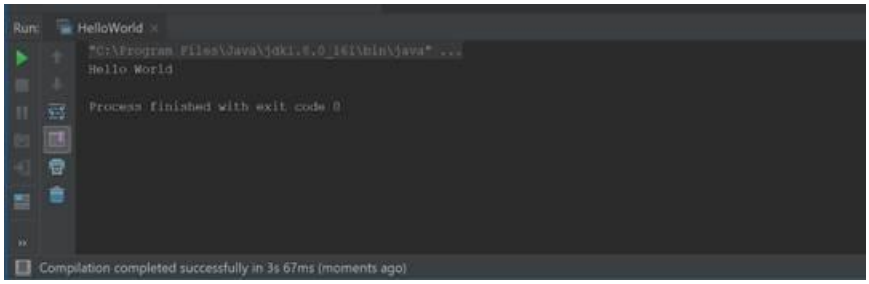


5. You are almost there. Now run your project.



- You will need to select HelloWorld as your run target. This is the file that contains your main method.

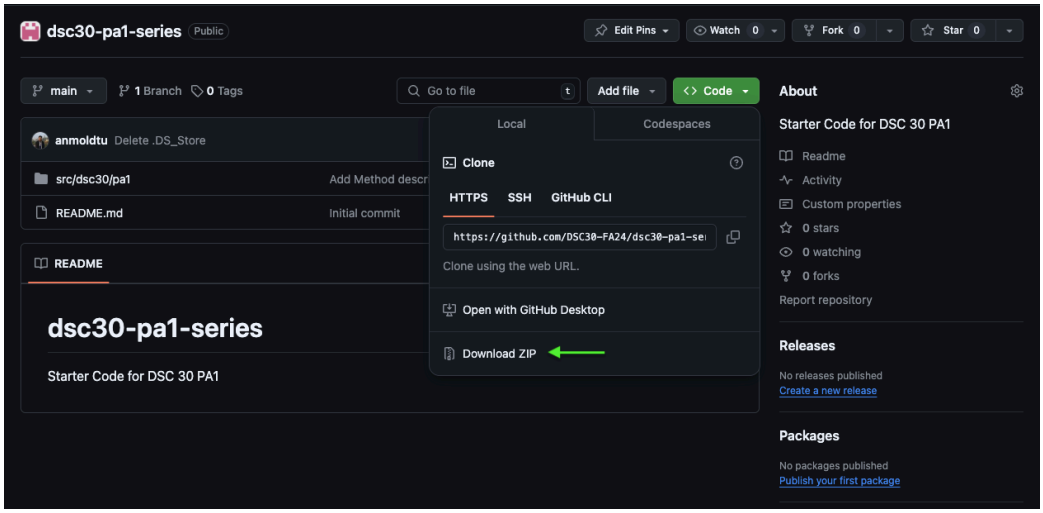
- Now your program will compile and run. At the bottom of your screen, you will see program output. If you see errors, please fix them and try again.



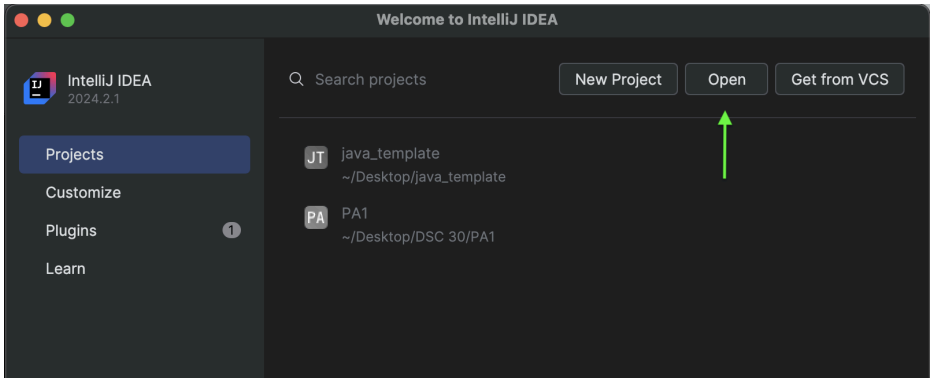
At this point, you have successfully created and executed your first java file. Now let's work on a *real* PA1.

Part 3: Using the Starter Code

Starter code for this assignment is available here. You can download the ZIP file through navigating to Code → Download ZIP as shown in the screenshot below.



Next, you need to open this project in IntelliJ. Navigate to Open in IntelliJ and open the downloaded folder: ***dsc30-pa1-series***



Part 4: Pandas Series (44 points)

This quarter, you are going to build a preliminary version of Python Pandas in Java. Specifically you will implement a Java object that works similarly to Series and DataFrame.

Series.java in the starter code includes two instance variables and nine incomplete methods.

Instance variables

```
private String[] rowNames;
```

Row names of the series.

```
private int[] data;
```

The integer array that contains the list of data that constitutes a Series object.

Methods

```
public Series(String[] _rowNames, int[] _data)
```

Inputs

_rowNames: an array of row names
_data: an array of int data

Constructor. If _rowNames is null, use index numbers as the default row names. For example in this scenario, the first data will have a row name “0”.

Note: Once the row name is initialized, it cannot be changed from outside the class. Specifically, the following code should not change the row name of the series s.

```
int[] d = {1, 200, 3};
String[] rn = {"a", "b", "c"};
Series s = new Series( rn , d);

// the second row name of the series s should remain as “b”
rn[1] = "z”;
```

`public String toString()`

Output
A string that shows a table of row names and data values, with the following formatting. Please note that the space separation between the rowName and data in each row is “\t”.
 	
`========`
`Printing Series...`
 	
`a  1`
`b  200`
`c  3`

Use this method whenever you want to check the current data of a Series instance.

`public int getLength()`
Output The length of the series object

`public String[] getRowNames()`
Output The rowNames array of this object

`public String[] getData()`
Output A **String** array that has the list of the data values.

The integer values must be converted to strings.

`public void append(String rn, int d)`
Inputs rn: a row name d: a data

Add a new pair of rowName - data at the end of the Series object. Assume there are no two rows that share the same rowName. In cases when rn is null or an empty string, then use index number (where rn will be appended) as the default rowName

`public int loc(String rn)`
Input rn: a row name
Output The retrieved data value. Return -1 if the label rn does not exist.

Read a data value, given a rowName.

`public int[] loc(String[] rn)`
Input rn: an array of row names

Output
An array of the retrieved data values. If a certain label within rn[ ] does not exist, fill -1 in the output array. If rn is null, just return null.

Read a bulk of data values, given a bulk of rowNames. You may utilize the other loc() above.

`public int iloc(int ind)`
Input ind: an integer index
Output The retrieved data value, located at the input integer index. Return -1 if the ind does not exist.

Read a data value, based on an integer index.

`public boolean drop(String rn)`
Input rn: a rowname
Output Return true if the drop was successful. Otherwise false.

Remove a pair of rowname-data, given a rowname.

Part 5: Code Submission

- Submit `Series.java` to CompassX under **PA1**.
- Make sure the names of the files are matching.

Your score will be determined by the **latest** submission on CompassX.

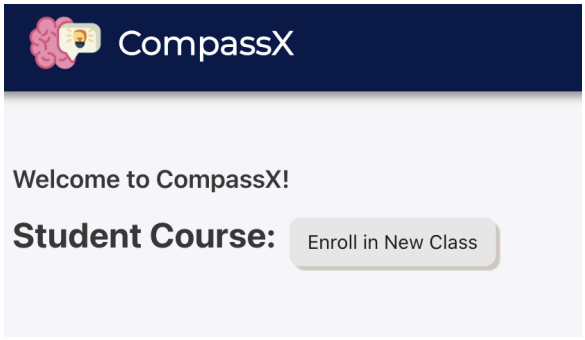
Note:

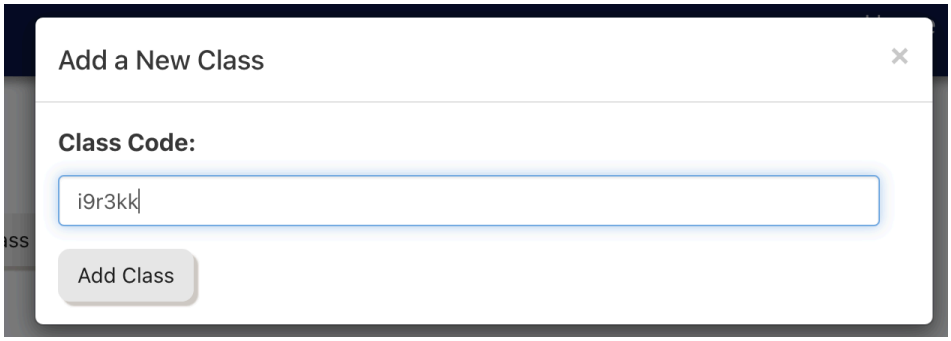
- Make sure that you have all the methods and required files before uploading the code to the autograder. You will get an error and no credit if you have any method missing.
- Make sure you have implemented getRowNames and getData before submitting on CompassX, as they need to be correct in order for us to validate your code

Instructions for CompassX setup

Log in through your UCSD **student email** for CompassX. This is how we keep track of your score for PAs.

Enroll in the DSC30 class using the class code: **i9r3kk**





Now you can see a class appearing in the student course.

There are 4 parts in total for each PA, and you need to do it in order.

You can only submit the plan and evaluation form once. Please think carefully before you submit them.

Once you have finished some parts, they will turn green.

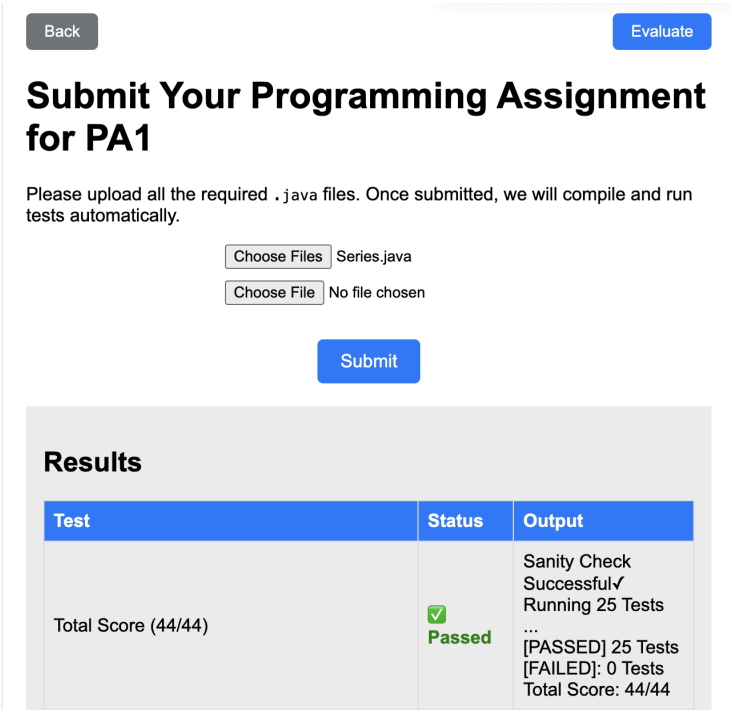
Once you submit the plan, there will be the PA write up in another tab.

Instructions for Code Submission on CompassX

Once you upload your files on CompassX you will also receive feedback from our testing script. Feel free to submit your code periodically to check your progress, as you work on PAs. It is your responsibility to submit **all the required** files listed in the Code Submission section, in order to receive full credits for this assignment.

Log into CompassX, and select our course DSC30. On the class dashboard, select the current assignment `PA1`. Upon clicking submit, a window will prompt for the submission method. Select **Submit** as your submission method.

Select your `Series.java` file and upload it. The result will inform you whether your file passes all test cases or not.



Submission Message:

1. If your file passes all our test cases, you will see a report as following:

Test	Status	Output
Total Score (44/44)	✅ Passed	Sanity Check Successful✓ Running 25 Tests ... [PASSED] 25 Tests [FAILED]: 0 Tests Total Score: 44/44

2. If your file fails some of our test cases, you will see a report looking like this:

Test	Status	Output
Total Score (26/44)	✗ Failed	Sanity Check Successful✓ Running 25 Tests ... [PASSED] 15 Tests [FAILED]: 10 Tests Total Score: 26/44

3. If you do not submit the required file, you will see a report looking like this:

Results

Test	Status	Output
File Series.java not found! Re-upload your submission to fix this!		