CMSC131 Fall 2021 (https://www.cs.umd.edu/class/fall2021/cmsc131-01XX-03XX/)

Project #4, Passport Class (Due Monday, Oct 18, 11:55 pm)

Objectives

To practice basic class definition.

Overview

For this project you will implement a **Passport** class that represents a person's Passport. A passport has the full name of a person (first, last and middle name) along with the stamps of countries the person has visited. At the end you will find a driver that illustrates some of the functionality of the this class.

For this project you can discuss the project with classmates, but you may not exchange code nor write code together. Your goal is to learn so you can be successful in internships and class exams. You do not need to include the names of people your discuss the project with.

This project has been used in previous installments of the course. Some students may find implementations for this project online. If you use any such implementation you will be violating academic integrity rules and an appropriate academic case will be submitted which may result in an XF in the course.

Grading

- (14%) Public Tests
- (62%) Release Tests
- (10%) Secret Tests
- (10%) Student Tests
- (4%) Style (Good indentation)

Code Distribution

The project's code distribution is available at PassportClass.zip (https://www.cs.umd.edu/class/fall2021/cmsc131-01XX-03XX/prot/projects/zipFiles/PassportClass.zip) The code distribution provides you with the following:

- A file named **Passport.java** - This is the class you need to implement.
- A file named **PublicTests.java** - This represents the public tests for the project.
- A file named **StudentTests.java** - Where you will write student tests.
- A file named **Driver.java** - Illustrates how to use some of the methods of the class.
- **expectedResults** - A folder that has the expected results for each of the public test.
- **results** - A folder that the results generated by your code for each of the public tests.

The Passport.java class can be found in the **programs** package. Remember that a package is similar to a folder. Just click on the Eclipse triangle you will see next to the **programs** package and you will see the Passport.java file. The PublicTests.java can be found in the **tests** package.

Specifications

Your task is two implement the **Passport** class. This class represents a person's passport. It has three instance variables representing the first, last and middle name (all are string variables). A character instance variable representing a separator (to be used for formatting purposes) is also part of the class. In addition, a StringBuffer instance variable is used to keep track of passport stamps. A description of each method you need to implement can be found at javadoc (doc/index.html). Remove the entry "throw new ..." with the code expected for a method. We add this entry so the code compiles and a white square is generated in the submit server when a method has not been implemented.

Other

- For your project, the initialization of instance variables should take place in the constructors and not at declaration time.
- Your tests will fail if your toString() method is not generating the correct string.
- Although you could define them, there is no need to define additional methods.
- Remember that hard-coding your code so you pass tests is considered an academic integrity violation.
- When you run public tests, files with the "Results.txt" extension (e.g., pub1ConsResult.txt) will be created if no exceptions where thrown by your code (an exception is an error). To see the file(s) created after running public tests, you need to Refresh the Eclipse project (right-click on the project and select Refresh).
- You have 5 release tokens that regenerated every 22 hours.

How to Start the Project

- Write the constructor that has several parameters.
- Write a **toString** method that allows you to see the content of the object.
- Continue with other methods of the class.

Student Tests

Test your code as you develop it, by writing tests in the **StudentTests.java** file. See the information we provided in that file. Using a JUnit test method is equivalent to writing a main method that calls the methods you are implementing. The JUnit test method will help you as you don't have to write the main method (just the code that appears inside). For this project you don't have to worry about using assertions (if you don't know what they are don't worry) with your student tests. Each test will use System.out.println to display results in the Eclipse console. **You may not be able to get help during office hours unless you have student tests. Some students implement their project by running public tests over and over; this is fine at the beginning, but now you have to start writing your own tests, as public tests may not be present in future projects.**

To grade your student tests, we will check that you have a test for each **public** method of the **Passport** class, regardless of what the test is testing (this will change in future projects). You don't have to worry about providing a test for the private method as it is tested implicitly via public methods.

You should write your tests as you write your methods (that is the correct approach). Remember that in the real world you will not have public tests; you have to learn how to write your own tests. Also, we have secret tests, so if you want to pass them, you need to thoroughly test your code. **By the way, sharing of student tests is not allowed.**

We will grade student tests for style (i.e., you should be using good variable names and good indentation). Also, although you could test more than one Passport class method on each student test method, it is preferable to have each method of StudentTests.java test one or two Passport methods.

During project grading it will help TAs if you name each test (method in StudentTest.java) after the method(s) you are testing. For example, test01Constructor, test02DefaultConstructor, test03PepsiTest, etc.

Sample Driver

The following driver relies on the class you need to implement. You will find this driver in the code distribution.

Driver

```
public class Driver {

	public static void main(String[] args) {
		Passport.resetNumberOfPassportObjects();

		Passport passport1 = new Passport("claudia", "I.", "smith");
		System.out.println(passport1);

		Passport passport2 = new Passport("John", "RoberTs");
		System.out.println(passport2);

		Passport passport3 = new Passport();
		System.out.println(passport3);
		System.out.println("=============");

		passport1.addStamp("Spain");
		passport1.addStamp("London");
		System.out.println("Stamps for " + passport1 + "-->" + passport1.getStamps());

		passport1.setSeparator('#');
		System.out.println(passport1);
		Passport passport4 = new Passport("CLAUDIA", "I.", "smith");
		System.out.println(passport1 + " same to " + passport4 + "? " + passport1.equals(p

		System.out.println("=============");
		System.out.println("Comparing");
		System.out.println("Comp1: " + (passport1.compareTo(passport2) > 0));
		System.out.println("Comp2: " + (passport2.compareTo(passport1) < 0));
		System.out.println("Comp3: " + (passport1.compareTo(passport4) == 0));

		System.out.println("=============");
		System.out.println("Normalizing");
		Passport normalized1 = Passport.normalize(passport1, true);
		System.out.println("Normalize #1: " + normalized1);
		System.out.println("Normalize #2: " + Passport.normalize(passport1, false));
		System.out.println("Normalize #3: " + Passport.normalize(passport4, false));

		System.out.println("Number of objects: " + Passport.getNumberOfPassportObjects());
	}
}
```

```
Output
Smith,Claudia,I.
Roberts,John
Samplelastname,Samplefirstname,Samplemiddlename
=============
Stamps for Smith,Claudia,I.-->SpainLondon
Smith#Claudia#I.
Smith#Claudia#I. same to Smith,Claudia,I.? true
=============
Comparing
Comp1: true
Comp2: true
Comp3: true
=============
Normalizing
Normalize #1: SMITH#CLAUDIA#I.
Normalize #2: smith#claudia#i.
Normalize #3: smith,claudia,i.
Number of objects: 7
```

Submission

Submit your project as usual. Make sure you check the results in the submit server. Remember that you need to do release testing to see release tests results.

Academic Integrity

Please make sure you read the academic integrity section of the syllabus so you understand what is permissible in our programming projects. We want to remind you that we check your project against other students' projects and any case of academic dishonesty will be referred to the University's Office of Student Conduct (https://www.studentconduct.umd.edu/).

Web Accessibility (https://www.umd.edu/web-accessibility/)