CS310 Data Structures Java Programming Assignment 1 Rubric

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Java Program: |  | | | |
| Rating Rating Category | Exemplary | Partially Proficient | Basic (needs work) | Not Demonstrated |
| Documentation | The program includes all required documentation and all documentation is clear. | The program includes all required documentation, but some of the documentation lacks clarity. | Some of the required documentation is missing, or most of the documentation lacks clarity. | No documentation was included. |
| Pts | 10 | 9 - 7 | 6 - 1 | 0 |
| Donor Class: attributes, constructors, getters, setters | Correctly implemented class with all required attributes, constructor, getters, and setters. | There are one or two minor errors in definition of the class. | There were multiple minor, or major errors in class definition. | Class was not defined. |
| Pts | 8 | 7 - 5 | 4 - 1 | 0 |
| Donor Class:  equals, toString, email validation | Correctly implemented toString, and equal methods, and method to validate email address. | There are one or two minor errors in definition of the methods. | There were multiple minor, or major errors in method definitions. | None of these methods were defined. |
| Pts | 8 | 7 - 5 | 4 - 1 | 0 |
| Donation Class: attributes, constructors, getters, setters | Correctly implemented class with all required attributes, constructor, getters, and setters. | There are one or two minor errors in definition of the class. | There were multiple minor, or major errors in class definition. | Class was not defined. |
| Pts | 8 | 7 - 5 | 4 - 1 | 0 |
| Donation Class: equals, toString, check validation | Correctly implemented toString, and equal methods, and method to validate check numbers. | There are one or two minor errors in definition of the methods. | There were multiple minor, or major errors in method definitions. | None of these methods were defined. |
| Pts | 8 | 7 - 5 | 4 - 1 | 0 |
| main class methods: SetDonorAttributes  GetDonorAttributes | Correctly implemented methods. | There are one or two minor errors in definition of the methods. | There were multiple minor, or major errors in method definitions. | Neither of these methods was defined. |
| Pts | 8 | 7 - 5 | 4 - 1 | 0 |
| main class methods: SetDonationAttributes  GetDonationAttributes | Correctly implemented methods. | There are one or two minor errors in definition of the methods. | There were multiple minor, or major errors in method definitions. | Neither of these methods was defined. |
| Pts | 8 | 7 - 5 | 4 - 1 | 0 |
| main class:  Test Set 1 | Coded both tests correctly and validated that tests executed successfully. | One necessary test was missing, or one test was not successfully executed. | Neither test was successfully executed. | No tests were created. |
| Pts | 8 | 7 - 5 | 4 - 1 | 0 |
| main class:  Test Set 2a | Coded test correctly and validated that test executed successfully. | Minor errors creating object, reading data from file, using setters to set attributes, validating email or using getters to display attributes. | Major errors in test suite coding, missing necessary parts of the test, or did not validate that test executed successfully. | Test not created. |
| Pts | 8 | 7 - 5 | 4 - 1 | 0 |
| main class:  Test Set 2b | Coded test correctly and validated that test executed successfully. | Minor errors creating object, reading data from file, calling method to set attributes, validating email or calling method to display attributes. | Major errors in test suite coding, missing necessary parts of the test, or did not validate that test executed successfully. | Test not created. |
| Pts | 8 | 7 - 5 | 4 - 1 | 0 |
| main class:  Test Set 3 | Coded both tests correctly and validated that tests executed successfully. | One necessary test was missing, or one test was not successfully executed. | Neither test was successfully executed. | Neither test was created. |
| Pts | 8 | 7 - 5 | 4 - 1 | 0 |
| Java Constructs /  Readability /  Miscellaneous | Code is organized and easy to follow, with no issues, and there is no unnecessary code. | There are minor construct, readability or other miscellaneous issues. | There are major construct, readability or other miscellaneous issues. | The code is unreadable. |
| Pts | 10 | 9 - 7 | 6 - 1 | 0 |
| Java Program SubTotal (any late points deducted will be based upon this subtotal) | | | | |
| Deductions |  | | | |
| Delivery | Submitted on time | Submitted 1-2 days late  (2% deducted per day late) | Submitted 3-5 days late  (2% deducted per day late) | Not submitted within 5 days of due date |
| Late Deductions | 0 | –2% to –4% | –6% to –10% | –100% (no credit) |
| Unacceptable Program | Compiles without syntax errors. |  |  | Does not compile without errors. |
| Unacceptable Deductions | 0 |  |  | –100% (no credit) |

MISCELLANEOUS

Read section 1.7 on how to use NetBeans to do some of this work for you. Work smarter, not harder.

MAIN –

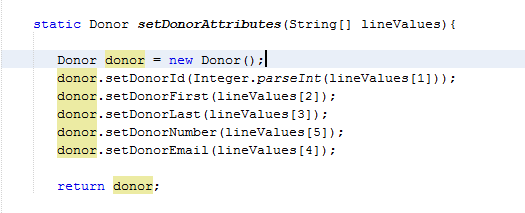
The setDonor/DonationAttributes and the getDonor/DonationAttributes methods should be in the main method.

Think of the domain objects as “nouns”, where the domain class describes the class. The Donor has names, phone numbers, etc. The method interact with the attributes, getters/setters, validations, etc. These four methods act on the class, and could be considered verbs, which is why they would not be in the domain class. I realize it is a fine line this week, but you will see why more and more as we continue. For next week, please move these methods back into your main class.

When you are calling the setDonorAttributes and setDonationAttributes, you were including a Donor/Donation object in your parameters. Each method then returned the processed Donor/Donation. You do not need to do that. To call the method, you can use



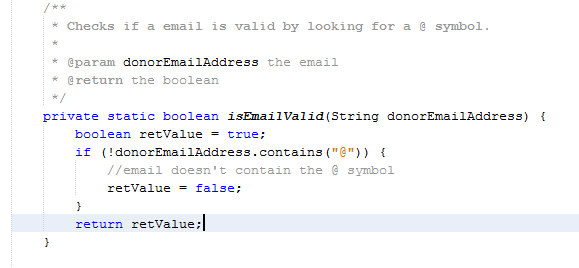
You then instantiate an object in the respective setDxxxAttribute method, and then return that. This will make your app a bit more efficient. Also note I changed the name of the donation object. In OO, you want to “decouple” methods and classes from each other. Now this method can be used anywhere, while the calling method can use a more specific name.



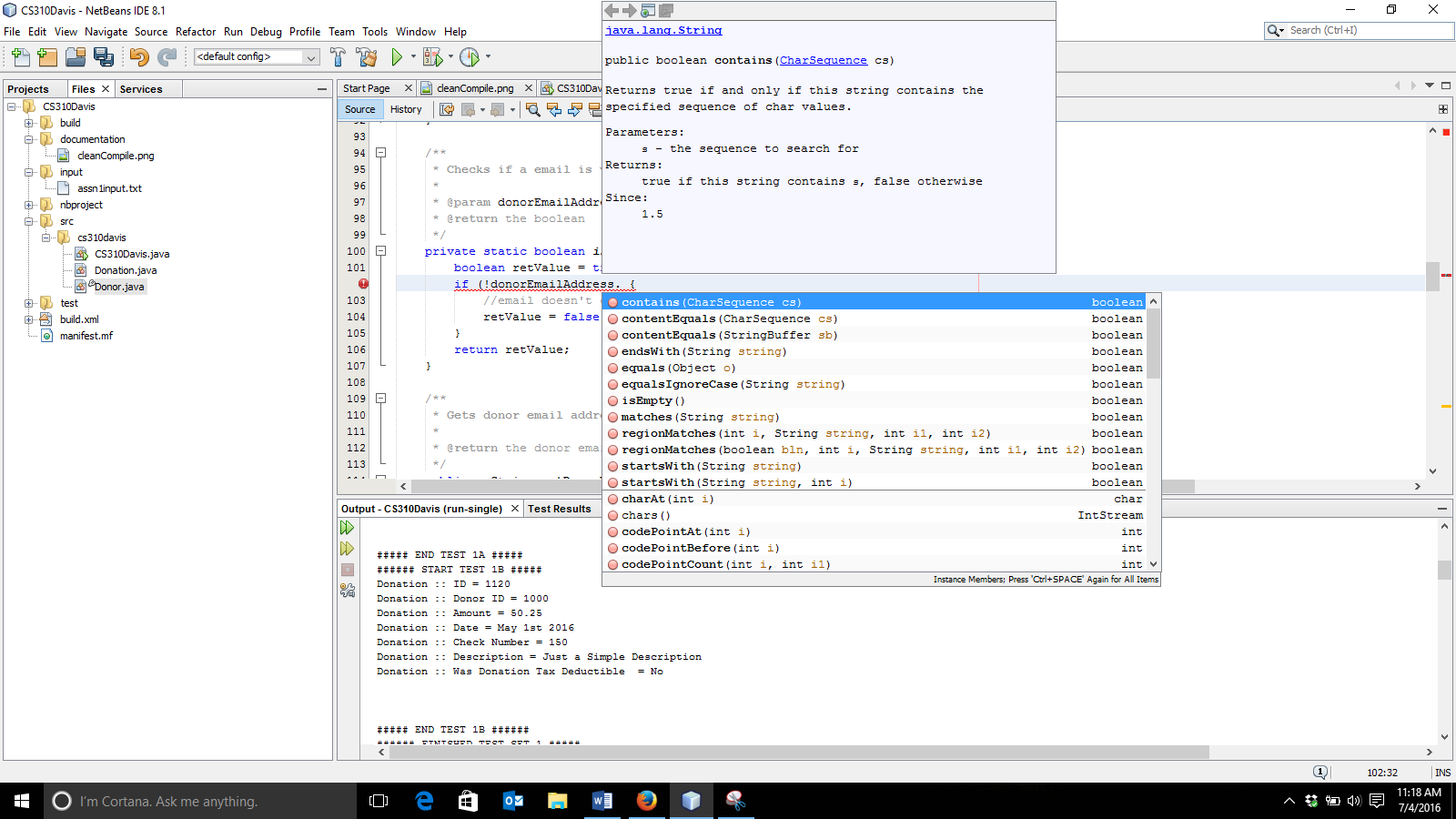
JAVA CONSTRUCTS – it is good practice to only have one return statement in a method. This makes it much easier to debug code later on for someone who needs to maintain the code, or if you are like me, you forget what you were doing six months from now. You do not need to worry about a return statement stuck in the middle of a large method that you may overlook.



In order to only have one return statement, you can use the following. Note when you initialize the return value, you want it to be the value that is most likely to be returned. Once the if statement has been processed, and it does not need to go any further, you are saving cycles.



You can even take this a step further to simplify your method. Looking at the documentation for the contains method, you will see it returns a boolean (again read section 1.7 to see how I did this). You can simply just return that without the need to assign it to a local variable.



The method can now look like:

