# Required Structure - READ THIS!

The TestClass.java and Class.java structure along with the try-catch blocks in the TestClass.java and the exception handling and isValid() methods in the Class.java are required for this course. Not building the Rectangle and ... and other programs in this way will not get credit.

<https://docs.oracle.com/javase/8/docs/api/index.html>

<https://introcs.cs.princeton.edu/java/31datatype/StockQuote.java.html>

<http://www.cs.cornell.edu/courses/cs1110/2010fa/handouts/demos/10sep28stockquote/StockQuote.java>

Assignment 4

For those of you doing MIS flavor problems this assignment contains the second part of an important MIS concept called sort/merge.  A sort program was included in the previous assignment; this one calls for a merge program followed by a sort.

For both editions, this assignment begins the study of class creation. While studying the appropriate chapters and sections from your version of the text (Chapters 9.1-9, 10.1-5, and 12.1-9) you should also check the attached videos and examine the Building.java file for a heavily commented example of how I required the classes and their associated test classes in an application to be constructed for robust execution. The accompanying .rar file contains the Building.java class definition along with its TestBuilding.java class, a child class called Garage.java, and its test program, as well as a bunch of Exception classes.  Examine these files carefully for an example of how a class and its test class should be structure for input error checking using exception handling.

[CS501ClassStructure1.mp4](https://sit.instructure.com/courses/21994/files/2443511/download), [CS501ClassStructure2a.mp4](https://sit.instructure.com/courses/21994/files/2443510/download), [JavaExceptionHandling.mp4](https://sit.instructure.com/courses/21994/files/2443510/download)

[BuildingGarage.rar](https://sit.instructure.com/courses/21994/files/2443524/download)

The collection of classes in the Java 8.0 Standard Edition is at: [http://docs.oracle.com/javase/8/docs/api/ (Links to an external site.)Links to an external site.](http://docs.oracle.com/javase/8/docs/api/).  There might be a comparable page for your version of Java.

This assignment begins with building a class, its test class that will contain the main() method, and the use of exception handling.

Exceptions are situations where some unexpected event happens, as when input is expected to be a number but is a word. Java provides several structures and key words that allow the program to become aware of the exception and handle it so that the program continues to operate. It is expected that after an exception due to invalid input the **user is notified and the program returns to the input section to allow the user to enter valid data**. Alternatively, the program may examine the error and if capable will either fix it or ignore it, and go on.

The point of exception handling is that the program does **not crash**, as is necessary for 24/7 operations.  As shown in the video, it is recommended that validation be done in isValidxxxx() methods in the class which returns True or False. If False then the methods that invoked the isValidxxxx() method, usually a set() method, which is coded to check any changes to be made in the class' parameters, should throw an exception, which is caught by a catch block in the calling class, in this case the test class. Observe this structure in the Building.java and TestBuilding.java classes.

Note also that all Java classes, whether from the JDK or written by individuals, are arranged into a hierarchy headed by a **root class called Object**. Every other class, either supplied by the Java compilers or created by programmers, is a child class of either the Object class or one of its child classes. Chapter 11, Inheritance and Polymorphism in both editions, discusses this in greater detail. You are welcome to refer back to the Building.java class as you get into these chapters.

While the Building.java class is extensively documented with comments, CS501 exercises may not need such frequent nor voluminous comments and methods, but it is a good idea to get into the habit of creating classes along the pattern of Building.java.

Although not requested in the assignments, whenever you build a class in Java it is a good idea to build it with all its expected methods:

* default and paramaterized constructors,
* all get() and set() methods possibly with validations checks called by the set() methods,
* any special methods, like for calculating area() as in the rectangle class, and
* standard methods that override the ones included automatically from the Object class at the root of the class hierarchy such as toString(), (see Section 11.6) and equals(), (see Section 11.10).

All the methods automatically included in the Object class can be found by scrolling down to the Object class in [http://docs.oracle.com/javase/8/docs/api/ (Links to an external site.)Links to an external site.](http://docs.oracle.com/javase/8/docs/api/).  Some of these are obviously not often required but the toString(), equals(), and getClass() are frequently used in practice.  For debugging you might want to include a print() method that displays the values of all the parameters.

This assignment asks you to create programming classes and objects representing geometric and business objects. One of the goals is to have these classes be reused in possibly many different kinds of applications. Since these classes often do not know what applications they will be a part of when they were created, they have to be created in a general manner. Best is that each class be in its own file and its associated test class be in a different file. The main() method is in the test class; there can be a main() methods in the class itself but they are usually reserved for testing. My recommendation is to put a main() method only in the test classes. The advantage of this separation is that when a class becomes part of larger application with many other classes all the test code does not have to be dragged into the application. Follow this distinction as shown in the chapter.

For 9.1 and 9.11/9.12 allow the user to enter the numbers.

|  |  |  |
| --- | --- | --- |
| Assignment 4 | 11th Edition | 10th Edition |
| CS | 9.1 w/ exception handling from Ch12,  9.5,  and  9.11/9.12 as single program | 9.1 w/ exception handling from Ch12,  9.5, and  9.11/9.12 as single program |
| MIS | 7.31,  10.9 w/ exception handling from Ch12, and  9.5 | 7.31,  10.9 w/ exception handling from Ch12,  and  9.5 |

Your response to Exercises 8.1 or 10.9 should include try-catch blocks and exception handling.  Have the user enter the data for the three students and check at least the grades to be A, B, etc.  Exceptions, involving *try*, *catch*, *throw*, *throws*, and *finally* commands, are described in Chapter 12. Write the program so that after the user enters the input it is validated in the Class and if valid the appropriate results are shown. If not valid, the Class should throw an exception to the catch block in the Test class which notifies the user with a message about the error and then the program should return to the input portions. See the relevant videos.

For the Course class (Section 10.5) in Exercise 10.9 add a String grade[] array and have the user enter grades for students which should be validated to be either A, B, C, D, or F using exception handling.  Have the students be compiled into the test class as in Listing 10.5 (so the user does not have to enter student names) and run a loop to ask the user to enter the grade for each student in each course.  Be sure to show the user the names of the students.

Remember:

1. All programs need a brief introduction printed for the user saying what the program will do.
2. The prompts to the user for the inputs required for the program need to say what each input is (*g.,*letter, word, number phrase), their units, and how it should be formatted (*e.g.,* 3 number separated by spaces).