# COMS W1007x: Honors Intro to Computer Science
# Fall, 2012
# Assignment 5: Due Tuesday, December 11, 2012, 5:00pm, 622 CEPSR

## Part 1: Theory (36 points)

The following problems are worth the points indicated. They are generally based on the book's exercises at the ends of Chapters 10 and 11, under the section heading "Exercises" and "Programming Projects". They cover concepts involving polymorphism, interfaces, sort/search, choosers, exceptions, and a few additional GUI widgets.

"Paper" programs are those which are written on the same sheet that the rest of the theory problems are written on, and are not to be text-edited, compiled, executed, or debugged. For fairness, there will be no extra credit for doing any work beyond the design and coding necessary to produce on paper the objects or object fragments that are asked for, so computer output will have no effect on your grade. The same goes for the Theory Part in general: clear handwriting is sufficient, and computer-generated documents will earn no extra credit.

1) (6 points) Based on EX 10.1 to 10.3: Draw and annotate a class hierarchy of at least three levels and at least six classes that represents various types of restaurants. (The supermost node would be Restaurant.) Show what characteristics (fields) would be represented in the various classes of the hierarchy. Explain how polymorphism could play a role in guiding your various restaurant activities (methods): your classes should obey the Liskov substitution principal.

2) (6 points) Based on EX 10.5: Explain how a call to the addItemListener method represents a polymorphic situation.

3) (6 points) Can try-catch statements be nested? If so, how should you determine the order of the nesting (that is, which goes inside of which)?

4) (6 points) Draw the first four levels of the Java Throwable hierarchy. Your diagram must have at least five classes at each level (you can show the rest by using dots, like ". . .").

5) (6 points) Based on EX 11.4: look up the following exception classes in the online Java API documentation and describe their purpose: IllegalArgumentException, ArrayStoreException, ClassCastException, FontFormatException, TooManyListentersException.

6) (6 points) Give two arguments in favor, and two arguments in opposition, to Java's use of checked exceptions. If you use the internet, you must cite your sources.

## Part 2: Programming (64 points)

This assignment is intended to exercise some of the more advanced concepts in sorting, searching, exceptions, and some of the more sophisticated GUI widgets. You build some databases, and integrate a number of classes to work together. This is probably the most typical real-world Java application in the course.

It is based partly on Listings 10.9 to 10.13 plus 11.7 to 11.13. Given that because of the hurricane we are sliding into the Reading Period, the following programming assignment has been made lighter: it is heavy on cut-and-paste programming, except for the creativity section.

Please recall that all programs must compile, so keep a working version of each Step before your proceed to the next.

**Step 1** (12 points): Getting started.

Implement 10.8, 10.9, and 10.10. (Just do it!) Show the sorted output using *both* sorts. Then, use System.nanoTime() to show how long each one runs.

**Step 2** (13 points): Add a file chooser and a scroll window.

Integrate into your system 10.13, so that you can choose from among several (at least three) different files that you have created. Each file contains a small database: each line of a file holds a last name, a first name, and a phone number that you can read using Scanner, instead of hardcoding it like in Listing 10.8. You will have to make some reasonable assumptions of how you can tell where the last name ends (what about "von der Pol"?) and the first name ends (what about "Mary Ann")? Display your outputs (from both sorts) in a ScrollPane, using Listing 11.13 to add scrolling functionality to Listing 10.13. Timing information can continue to be dumped to System.out.println().

In order to make this Step meaningful, you will probably have to generate files mechanically using a text editor or a small Java program; these files do not have to have actual names or phone numbers.

**Step 3** (13 points): Add buttons with appropriate tool tips and mneumonics.

As in Listing 11.10, make it so that the user can select just one of the sort methods after selecting the files. Have four buttons: selection sort ascending, selection sort descending, insertion sort ascending, insertion sort descending. There is a simple way to modify compareTo to accommodate sort direction.

**Step 4** (13 points): Add searching and a combo box.

Integrate the two search methods given in Listing 10.12. Make them be controlled by using a combo box as in Listing 11.12. The user selects either linear search or binary search, enters the target by using a text box, and gets output in some sort of text area, based on Listing 10.11. By now you will have enough GUI widgets that you should probably pay some attention to how to lay them out using a manager of some sort. Again, do timing information, but dump it to System again.

**Step 5** (13 points): Creativity step: add interpolation search and exceptions.

Find out how interpolation search works and code it up. Cite any sources you use. Since interpolation search can produce a division by zero (and you must devise a file to test this!), handle that exception by using a try-catch. Extend your combo box to include it as a third method. You must then show a file and a search target in which linear wins, another file and target combination in which binary wins, and a third in which interpolation wins.

## General Notes:

The general rules for the previous assignments still apply: If you do a Step you do not have to submit separate output from the previous Steps. Outputs here consist of the usual hard and soft copy of code, softcopy of javadoc, and enough screen shots to show that the system does in fact demonstrate the functionality of the Steps you attempted.

For each Step, design and document the system, text edit its components into files, compile them, debug them, and execute them on your own test data. When you are ready, electronically submit the text of its code, a softcopy of your Javadoc output, a copy of your testing (including any screenshots), and whatever additional documentation is required. Make sure that the source code is clear and any user interface is comprehensive and informative.

Make sure you have properly attributed any code that is legal to incorporate from other sources, including the book's online companion.

Write your classes clearly, with a large block comment at the beginning describing what the class does, how it does it, and justify how it will be tested. Put the comments about testing in a Tester class; if possible, show some test data in the comment also. Document each constructor and method.

Clear programming style will account for a substantial portion of your grade for both the Theory Part and the Programming Part of this assignment. For one reasonable set of suggested practices and stylistic guidelines, see the book's Appendix F. But it is far more important to be consistent: PoLS, everybody!

## Checklist:

Here is a checklist for submission. Please note that this is designed to be a *general* guide, and you are responsible for *all* things asked for in the text of the assignment above, whether or not they appear below.

For theory: Hard copy, handed in to the Prof (or slid under his door) by the given time. No extra points for using a text editor. Neatly stapled, separately from programming hard copy! Name and UNI attached.

For programming: Hard copy, handed in to Prof (or slid under his door) by the given time. All code, all testing runs (cut and pasted from console, and/or captured screenshots), all Javadoc if hardcopy Javadoc is required in hardcopy (for this assignment, it is not). Neatly stapled, and separately stapled from the theory! Name and UNI attached.

Also for programming: Soft copy, submitted to the Assignments page of Courseworks by the beginning of class, in a tarball created by CUIT Unix tar command, given below. Note that "myUNI" should be replaced with your UNI. *Please* use this convention as it makes the TAs' job much easier:

```
tar -cvzf myUNI_HW5.tar.gz whateverMyDirectoryIs
```

Also for programming: Include the softcopy of Javadoc html. Name and UNI included as comments in *every* class. The code must compile. Your last electronic submission before deadline is the official one that will be executed if needed.