Office hours after 6 May will be change. Please check the staff page on the course website frequently for updates. You may also make appointments with the staff, as specified on the course staff webpage.

Anyone with a conflict with the final exam (see page header for the date and time) already have registered on the course CMS by the end of 2 May.

Review sessions, beginning 6 Dec in Phillips 101.

Day	Time	Instructor	Topic
Mon	1PM	Ternquist	Subclasses and abstract classes, including constructors
Mon	2PM	Park	Casting, apparent/real classes; executing sequences of statements involving creating/using objects
Mon	3PM	Park/ Mamouras	Drawing frames for calls, executing method calls
Tues	1PM	Bapat	Exception handling, GUIs
Tues	2PM	Mamouras	Developing loops from invariants
Tues	3PM	Lloyd	Recursion
Wed	2PM	Gries	Developing the required algorithms
Wed	3PM	Gries	Arrays, vectors, strings, wrapper classes

The final is cumulative, *covering all topics in the course* except as described below. So, you have to know everything that was covered in the two prelims in addition to material presented after those two prelims. See the handouts on the two prelims (on the course web page).

Study by doing the practice exams on the course page. You may wish to check your solutions in DrJava. If you have had difficulties on the previous exams, we recommend you check your solutions with a staff member

These topics are not on the final: reading a file or the keyboard, applications, and applets, and interfaces.

In addition to the material covered in the prelims, this material may be on the final.

1. **Several algorithms**. Know the algorithms given below this paragraph. We may simply write "show binary search", or "Show us the partition algorithm", and you have to give the precondition, postcondition, and loop invariant and then develop the algorithm. Or, we may give you the header of the method and you have to write the precondition and postcondition that go with it and then develop the rest. We expect that: the loop with initialization is developed from the invariant; a loop that has nothing to do with the invariant gets little credit. Everyone should get full credit on this question because it is simply a matter of (1) memorizing specifications and then (2) practicing developing known algorithms from their specs. For selection and insertion sort, write only a single loop, not a nested loop, as explained *ad nauseum* in course material.

Binary search, Dutch National Flag, Partition algorithm, Selection sort, Insertion sort.

2. **Developing an algorithm: stepwise refinement**. We have used stepwise refinement in class many times, attempting to solve a little bit of a problem at a time. Read Sec. 2.5 on p. 82, and you might also study Sec. 9.2, p. 304, which discusses the development of several problems that deal with arrays.

3. Loops. (1) Be able to develop a for-loop that processes a range of integers, as on prelim 2. This includes developing the invariant. (2) For while-loops, be able to write a loop given the invariant, but you will not be asked to create the invariant for a problem you have not seen before.

4. **Array**. 1- and multi-dimensional arrays —rectangular and ragged. This includes knowing (1) how to access the number of columns in a row, (2) how to create a rectangular array or a ragged array, (3) how arrays are stored as objects, and (4) how to draw a Java array.

5. **Exception handling**. Be able to write a class that extends Throwable, Exception, or RuntimeException, with two constructors. Be able to write code to create an instance of such a class and throw it. Be able to write a simple try-statement with a single catch-clause. Understand what happens when an exception is thrown. Do not concern yourself with the *throws* clause; you won't need it. See the chapter on Exception Handling.

6. **Abstract classes**. Know the purpose of an abstract class abstract and its syntax. Know the purpose of an abstract method and its syntax. See Sec. 4.7 of the text and lesson page 4-5 of the ProgramLive CD.

7. **Interfaces**. **You are not responsible for these.** [Know the syntax for an interface definition. Know how to have a class implement an interface and what that means. Understand fully interface Comparable in package java.lang. Be able to: (1) write methods that have parameters of type (class) Comparable, (2) write a procedure that sorts an Comparable array, write a class that implements Comparable. See lectures slides and the appropriate section in the text.]

8. **Placement of components in a GUI**. Know the default layout managers for JFrame, JPanel, and Box and how the manager arranges components in it. Know these basic components: `JButton`, `JLabel`, `JTextField`, `JTextArea`. Know three things you have to do to listen to an event. Be able to understand programs that place components in a GUI and the code for listening to an event. You do not have to write code.