Check the header for the place of your final —depends on last name.

Office hours after 2 Dec. will be changed. Check Piazza frequently for updates. You may make appointments with the staff, as specified on the course staff webpage.

Conflict with the final exam? Complete the Final Conflict assignment on the CMS by the end of 1 December.

Review sessions, beginning 5 Dec in Phillips 101.

Day	Time	Instructor	Topic
Mon	1PM	Jun Ma	Subclasses and abstract classes, including constructors
Mon	2PM	Sidharth Telang	Casting, apparent/real classes; executing sequences of statements involving creating/using objects
Mon	3PM	Steve Marschner	Drawing frames for calls, executing method calls
Tues	1PM	David Gries	Exception handling, GUIs
Tues	2PM	Li Hao	Developing loops from invariants
Tues	3PM	Yexiang Xue	Developing the required algorithms
Wed	2PM	Deepak Bapat	Recursion
Wed	3PM	Prabhjeet Singh	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 the material that was 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. Check your solutions in DrJava. If you have had difficulties on the previous exams, we recommend that you check your solutions with a staff member.

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

In addition to the material covered in the prelims, the following topics may appear 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, together with its 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, we may ask you to write a single loop with an English statement of what its repetend does, rather than a nested loop, as explained *ad nauseam* 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 (except for the algorithms listed in point 1).

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 Exception or RuntimeException, with two constructors. Know when it makes sense to extend which class. Be able to write code to create an instance of an exception 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 in method declarations; you won't need it on the exam. See the chapter on Exception Handling.

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

7. **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 the 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 will not have to write code that deals with GUIs on the exam.