CSSE 220 Day 22

Exam 2 Review
File I/O, Exceptions
Vector Graphics Project

Check out *FilesAndExceptions* from SVN

Questions?

Today

- Exam 2 review
- File I/O and Exceptions
- Team Project kickoff

Exam 2

- Exam is in class tomorrow …

Possible Exam Topics

- Recursion
- Sorting and the Comparable interface
- Algorithm analysis and big-Oh notation
- Function objects (mainly Comparator)
- Immutable objects *vs.* side effects
- UML class diagrams
- Interfaces
- Inheritance
- Polymorphism
- Swing event handling
- OO design

- Nothing from today's class will be on the exam.

Exam Tomorrow in class!

- Topics from Chapters 1-14 and Sessions 11-20
- Will include:
  - A paper part (textbook plus one page of notes): short answer, fill-in-the-blank, trace-code-by-hand, draw box-and-pointer diagrams, find-errors-in-code, write short chunks of code, etc. About 1/3 of the exam.
  - A programming part (open-computer): a few small programs, possibly including recurrence, GUIs and event-handling, interfaces, inheritance.
- Review in class today
  - What questions did you bring?
  - What topics would you like to review?
  - I didn't prepare anything but I'm happy to cover whatever you want, including working examples.
  - **If I have a Q & A session 10th hour, how many of you are likely to come?**

Have you done these?

- Reviewed chapters 1 to 14 from Big Java
- Prepared a sheet of notes to help you summarize what you consider important
- Reviewed the slides, in-class quizzes, homework from sessions 1 to 21
- Practiced programming, unit testing, documenting your code, & using the Java API
- You can ask questions by email to the csse220-staff mailing list or your instructor

Files and Exceptions

Reading & writing files
When the unexpected happens

Review of Anonymous Classes

- Look at GameOfLifeWithIO
  - GameOfLife constructor has 2 listeners, one *local inner* class and one *local anonymous* class
  - ButtonPanel constructor has 3 listeners which are *local anonymous* classes
- Feel free to use as examples for your project

File I/O: Key Pieces

- Input: `File` and `Scanner`
- Output: `PrintWriter` and `println`
- Be kind to your OS: `close()` all files
- Letting users choose: `JFileChooser` and `File`
- Expect the unexpected: `Exception` handling
- Refer to examples when you need to…

Q1–Q4

Exceptions

- Used to signal that something went wrong:
  - `throw new EOFException("Missing column");`
- Can be **caught** by **exception handler**
  - Recovers from error
  - Or exits gracefully

Q5

A Checkered Past

- Java has two sorts of exceptions
- **Checked exceptions**: compiler checks that calling code isn't ignoring the problem
  - Used for **expected** problems
- **Unchecked exceptions**: compiler lets us ignore these if we want
  - Used for **fatal** or **avoidable** problems
  - Are subclasses of `RunTimeException` or `Error`

Q6–Q7

A Tale of Two Choices

- Dealing with checked exceptions
  - Can **propagate** the exception
    - Just declare that our method will pass any exceptions along
    - `public void loadGameState() throws IOException`
    - Used when our code isn't able to rectify the problem
  - Can **handle** the exception
    - Used when our code can rectify the problem

Q8

Handling Exceptions

- Use try-catch statement:

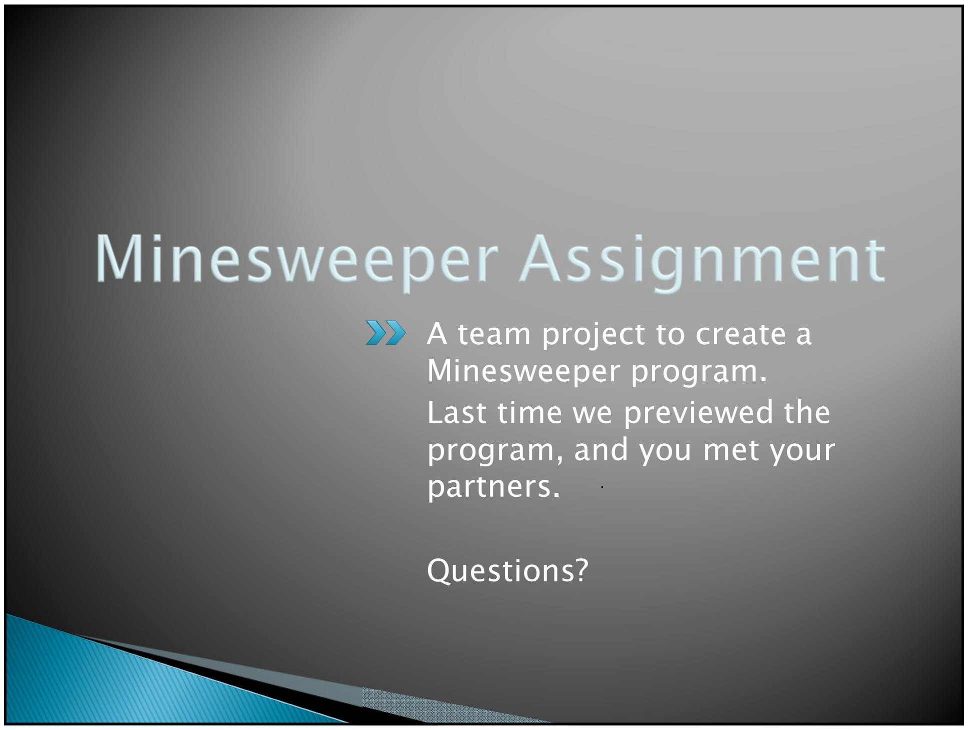
```
try {
    // potentially "exceptional" code
} catch (ExceptionType var) {
    // handle exception
}
```

Can repeat this part for as many different exception types as you need.

- Related, try-finally for clean up:

```
try {
    // code that requires "clean up"
} finally {
    // runs even if exception occurred
}
```

Q9-Q10

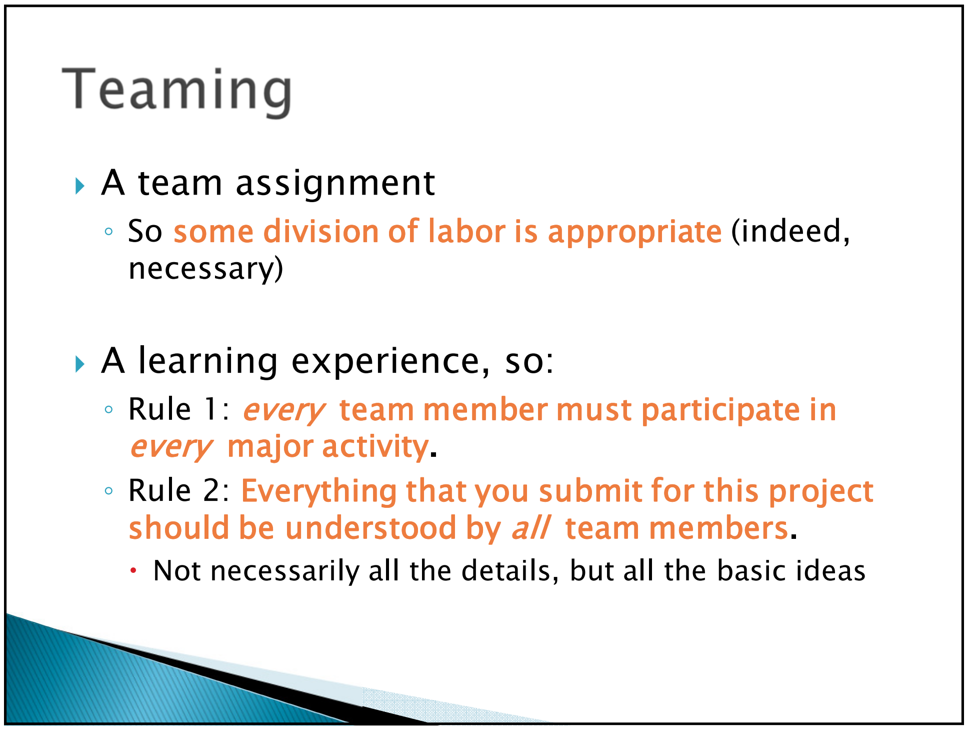


Minesweeper Assignment

- A team project to create a Minesweeper program.

  Last time we previewed the program, and you met your partners.

  Questions?



Teaming

- A team assignment
  - So some division of labor is appropriate (indeed, necessary)
- A learning experience, so:
  - Rule 1: *every* team member must participate in *every* major activity.
  - Rule 2: Everything that you submit for this project should be understood by *all* team members.
    - Not necessarily all the details, but all the basic ideas

Work time now

- Read the specification if you haven't done so
- Start working on your milestone 0 **due tomorrow**
  - Try to get it done in class today so you can:
    - Get some feedback in class before it's graded.
    - Focus on studying for the exam tonight.

Plan, then do

- If you complete these, show me:
  - CRC cards
  - UML – as complete as you can – will help coding later.
  - User stories for cycle 1
    - For some user stroy examples and ideas, see http://en.wikipedia.org/wiki/User_story
- Ask questions as needed!
- Work on the rest, cycle one due Tuesday.
  - There will be a required partner evaluation at end of project
  - When you are done cycle 0, you have my blessing to start coding cycle 1
  - Use any reasonable combination of:
    - group meetings and/or
    - dividing up the work