CSSE 220 Day 23

File I/O, Exceptions
LodeRunner Project

Check out *FilesAndExceptions* from SVN

Questions?

Today

- File I/O and Exceptions
- Team Project kickoff

Files and Exceptions

- Reading & writing files
- When the unexpected happens

Review of Anonymous Classes

- Look at GameOfLifeWithIO
  - GameOfLife constructor has 2 listeners, two *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–Q3

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

Q4

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`

Q5–Q6

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

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
}
```

Q8-Q9

LoadRunner Assignment

- Demonstrate the program

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.
    - E.g., you are not allowed to have someone do graphics but no coding,
  - 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 next class**
  - Try to get it done in class today so you can:
    - Get some feedback in class before it’s graded.

Plan, then do

- There are milestones due most class days:
- For next class:
  - User stories
  - CRC cards
  - UML class diagram
  - See the project description for details

  - Suggestion:
    - Plan to implement a considerable amount of functionality in Cycle 1
    - It is the longest cycle that you will have