CSSE 220 Day 21

File I/O, Exceptions
LodeRunner Project

Check out *FilesAndExceptions* from SVN

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

1. 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

1. Can handle the exception
   - Used when our code can rectify the problem

Q7

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

LodeRunner 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

Team Member Survey

- Fill out Partner Survey on Moodle
  - 3 Person Teams
- Read the specification for LodeRunner
- Teams will be posted by Monday's class

Plan, then do

- There are milestones due most class days:
- For Wednesday's 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