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

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

Q7

Handling Exceptions

- Use try-catch statement:
  - ```
    try {
        // potentially "exceptional" code
    } catch (ExceptionType var) {
        // handle exception
    }
    ```
- Related, try-finally for clean up:
  - ```
    try {
        // code that requires "clean up"
    } finally {
        // runs even if exception occurred
    }
    ```

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

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

Plan, then do

- There are milestones due most class days:
- For Friday:
  - 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

LodeRunner Teams – Section 1

csse220-201230-Lode11, patterda, armacoce, wintoncc

csse220-201230-Lode12, andersjr, kohlscd, weissna

csse220-201230-Lode13, shomerrt, bearder, rodriga

csse220-201230-Lode14, padillbt, dionkm, mccormjt

csse220-201230-Lode15, andrewca, thomaszk, alvareap

csse220-201230-Lode16, fagglr, heidlapt, johnsom2

csse220-201230-Lode17, yeomanms, yoons1, antleyp

csse220-201230-Lode18, jonescd, beckerja

csse220-201230-Lode19, moortr, meltonej

Check out *LodeRunner* from SVN

LodeRunner Teams – Section 2

csse220-201230-Lode21, petryjc, turnerrs, darttrf

csse220-201230-Lode22, almisbmn, brophywa, lashmd

csse220-201230-Lode23, phillijk, fritzdn, maibacmw

csse220-201230-Lode24, brokllh, abadbg, huangf

csse220-201230-Lode25, iversoda, solomovl, finneysm

csse220-201230-Lode26, depratc, earlesja, jennedj

csse220-201230-Lode27, wellsdb, brindldc, bromenad

csse220-201230-Lode28, yadavy, kowalsdj, hallami

Check out *LodeRunner* from SVN