CSSE 220 Day 22

Exam 2 Review
File I/O, Exceptions
LodeRunner Project

Check out *FilesAndExceptions* from SVN

Questions?

Today

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

Exam 2

- Exam is tomorrow evening…

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 at 7 PM!

- Topics from Chapters 1-14 and Sessions 11-21
- Will include:
  - A paper part (two double-sided pages 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 1/4 of the exam, 1/3 of the credit.
  - A programming part (open-computer): a few small programs, possibly including recursion, 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.

Have you done these?

- Reviewed chapters 1 to 14 from Big Java
- Prepared your 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

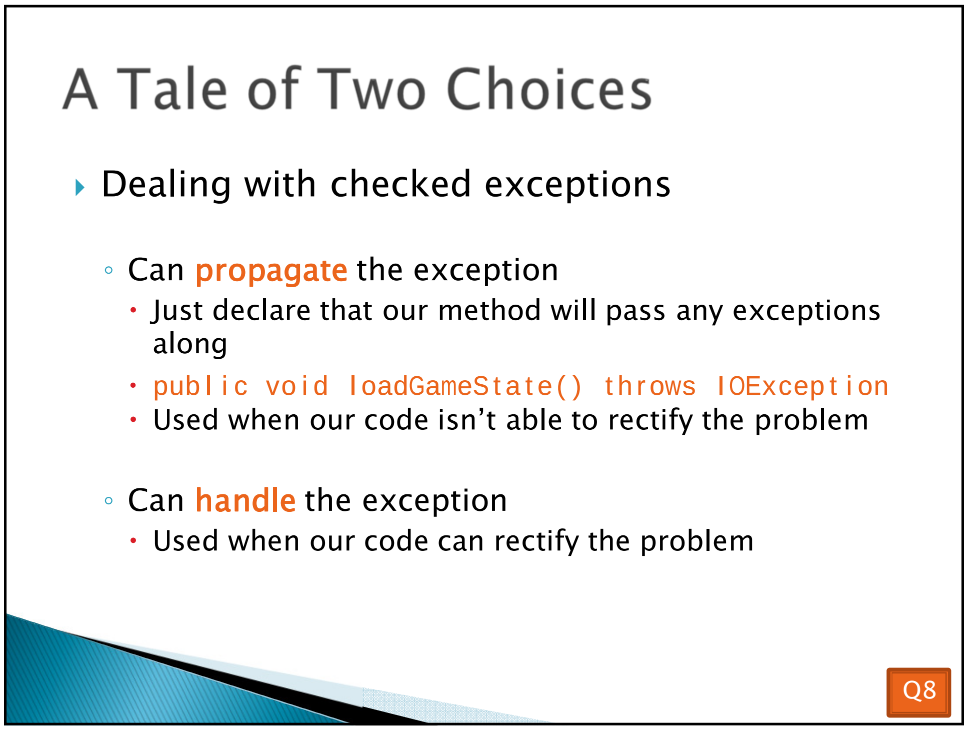
- Used to signal that something went wrong:
  - `throw new EOFException("Missing column");`
- Can be **caught** by **exception handler**
  - Recovers from error
  - Or exits gracefully

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`

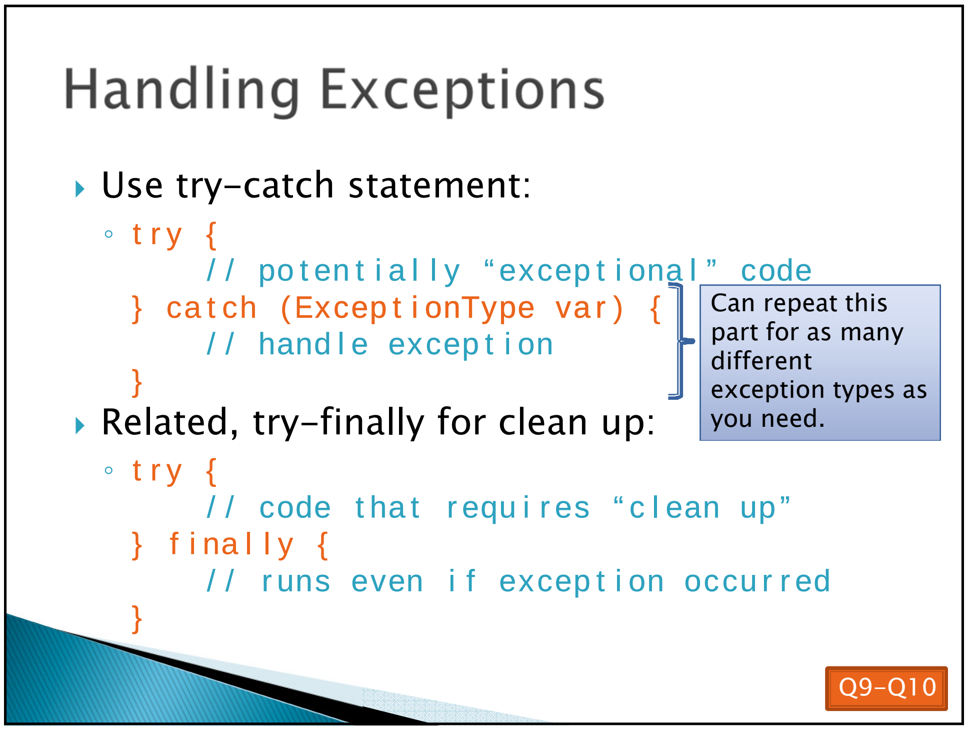
Q6-Q7



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

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 Thursday:
  - User stories
  - CRC cards
  - UML class diagram
  - Begin writing code for development Cycle 1
  - 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

csse220-201220-Lode11,jacksoam,toorha,weirjm

csse220-201220-Lode12,gartzkds,harbisjs,smithgb

csse220-201220-Lode13,conwaygt,satchwsm,wangl2

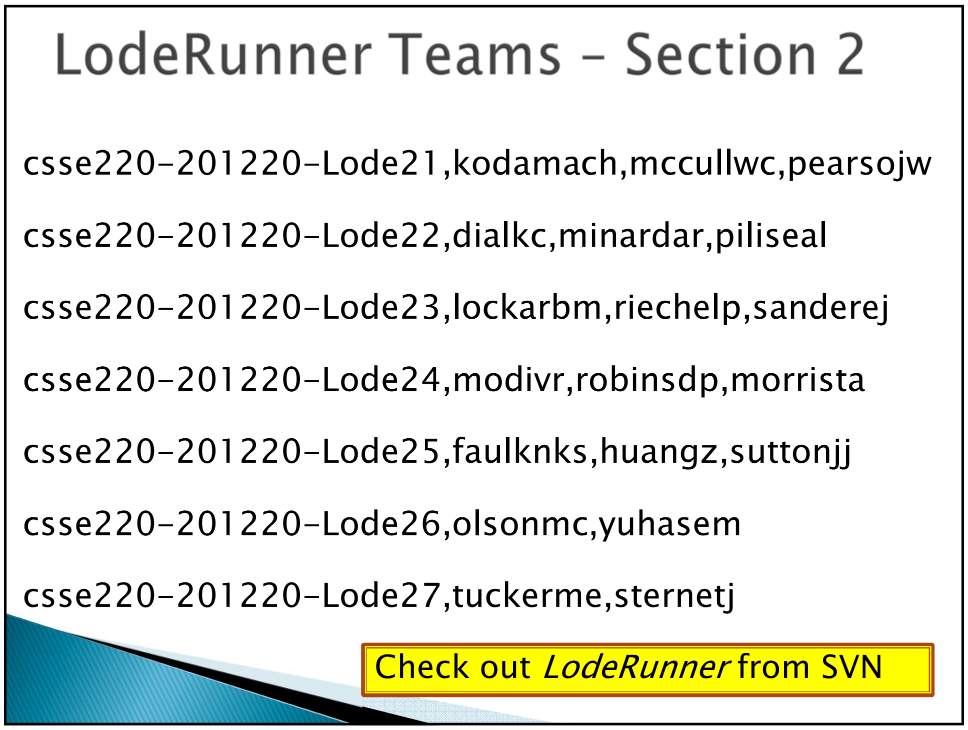
csse220-201220-Lode14,postcn,rujirasl,swenseen

csse220-201220-Lode15,ameslc,dingx,campbeeg

csse220-201220-Lode16,janeiraj,mcculfpe,murphysw

csse220-201220-Lode17,harrissa,koestedj,watterlm

Check out *LodeRunner* from SVN



LodeRunner Teams – Section 2

csse220-201220-Lode21,kodamach,mccullwc,pearsojw

csse220-201220-Lode22,dialkc,minardar,piliseal

csse220-201220-Lode23,lockarbm,riechelp,sanderej

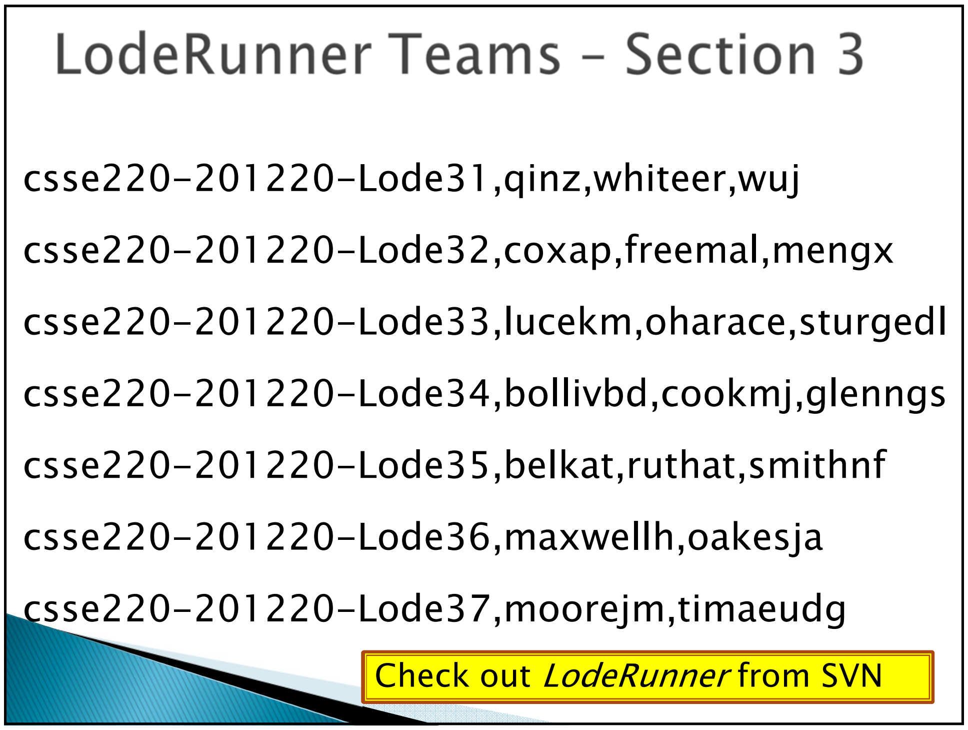
csse220-201220-Lode24,modivr,robinsdp,morrista

csse220-201220-Lode25,faulknks,huangz,suttonjj

csse220-201220-Lode26,olsonmc,yuhasem

csse220-201220-Lode27,tuckerme,sternetj

Check out *LodeRunner* from SVN



LodeRunner Teams – Section 3

csse220-201220-Lode31,qinz,whiteer,wuj

csse220-201220-Lode32,coxap,freemal,mengx

csse220-201220-Lode33,lucekm,oharace,sturgedl

csse220-201220-Lode34,bollivbd,cookmj,glenngs

csse220-201220-Lode35,belkat,ruthat,smithnf

csse220-201220-Lode36,maxwellh,oakesja

csse220-201220-Lode37,moorejm,timaeudg

Check out *LodeRunner* from SVN