Administrivia

- Reminder: Midterm exam next Tuesday. Review sheet on Web.

- Reminder: Homework 4 design due next Tuesday.

Slide 1

A Little About the Midterm

- Review class notes, example programs from class, minute essays, and quizzes (solutions online).

- Most questions will likely be more difficult (or at least longer) than quiz questions, but similar in format. Might be a few short-answer / multiple-choice questions too.

Slide 2

- Open book, open notes, some access to Web.

- If you want extra non-game "practice problems" to try, send me mail.

Homework 4 Overview

- Start writing code for your game entities. Similar to what you did for player last time.

- Review/revise how you're creating layout for your game. Several options.

Slide 3

- Write replacement for framework `GameEntityList`. This will be a linked list, based on discussion in class. You may find it helpful to draw pictures.

Error Handling — The Problem

- When you have a function in which something goes wrong, how do you tell the rest of the program?

- Examples:

  - Calling a square-root function with a negative number.

Slide 4

  - Trying to open (for reading) a file that doesn't exist.

  - Trying to convert a string to an integer, when the string doesn't contain something appropriate.

Error Handling — "Ostrich Approach"

Slide 5

- Idea — hope it doesn't happen.

- Might sort of work if you tell users in your documentation, and maybe use assertions.

- But users make mistakes, and what then? e.g., out-of-bounds array access.

- And it may not always be easy to tell what inputs will produce errors (e.g., file access).

Error Handling — Return Codes

Slide 6

- Idea — have method return an error code if something goes wrong.

- Works well in situations where it might be hard to avoid sometimes causing the error.

- But requires that users of the method check for the "error" return value — tedious and error-prone.

- And what about methods that want to return a value? is it always possible to designate some value as "this means an error"?

Error Handling — Setting Flags

Slide 7

- Idea — have method set a flag somewhere if something goes wrong.

- Also useful in situations where it might be hard to avoid sometimes causing the error.

- Again, though, users have to check.

- Requires either an extra parameter (and changing it may be tricky in Java) or a "global" variable somewhere.

Error Handling — Exceptions

Slide 8

- Idea — when something goes wrong, "throw an exception". What then?

- Aside — as program runs, we can think of it keeping a stack of nested method calls ("push" when we call a method, "pop" when one returns).

- When an exception is thrown, runtime system works its way up this stack until it finds something to "catch" the exception. If it never finds anything, it terminates the program (actually the thread).

- *Mostly* this is what Java library classes use to indicate errors — but some use return codes, so read documentation carefully.

Slide 9

Dealing With Exceptions

- Catching an exception — "try block":

```
try { ....  }
catch (TypeOfException e) { ....  }
catch (OtherTypeOfException e) { ....  }
finally { .... } // optional
```

- Letting an exception "bubble up":

```
void foo() throws WeirdException { .... }
```

- `Exception` class has some useful methods, e.g.,
  `printStackTrace`.

Slide 10

Checked Versus Unchecked Exceptions

- "Checked exceptions" — ones that sensible programs are supposed to do
  something about (e.g., file not found).

  Must either catch these, or declare that your method lets them bubble up (and
  then callers must do likewise).

- "Unchecked exceptions" — ones for which maybe the reasonable thing to do
  is to just let the program crash.

  Can catch these, or let them bubble up (with or without declaration), possibly
  eventually crashing the program.

Throwing Exceptions

Slide 11

- Throwing an exception:

  `throw new TypeOfException(....)`

- Usually best to try to find an existing `Exception` class that fits, but can declare your own.

- Example — `withdraw` method in our bank account class. (Revisit this next time.)

Exceptions Versus Other Approaches

Slide 12

- What's the attraction?
  - Nice mechanism for dealing with errors and unexpected events.
  - Unlike return codes, can't just be ignored.

- But checked exceptions can be annoying to deal with . . .

Slide 13

Minute Essay

- Here's a line of code that can throw an exception:

```
String s;
double x;
/* code to get a value from the user and put in s omitted */
x = Double.parseDouble(s);
```

Write a few lines of code to catch the kind of exception most likely to be

thrown (look at the documentation for Double.parseDouble() and

print out a meaningful error message.

Slide 14

Minute Essay Answer

- parseDouble can throw a NumberFormatException, so you
  could write:

```
try {
    x = Double.parseDouble(s);
}
catch (NumberFormatException e) {
    System.out.println(s + " is not a number");
}
```