Exceptions in Java

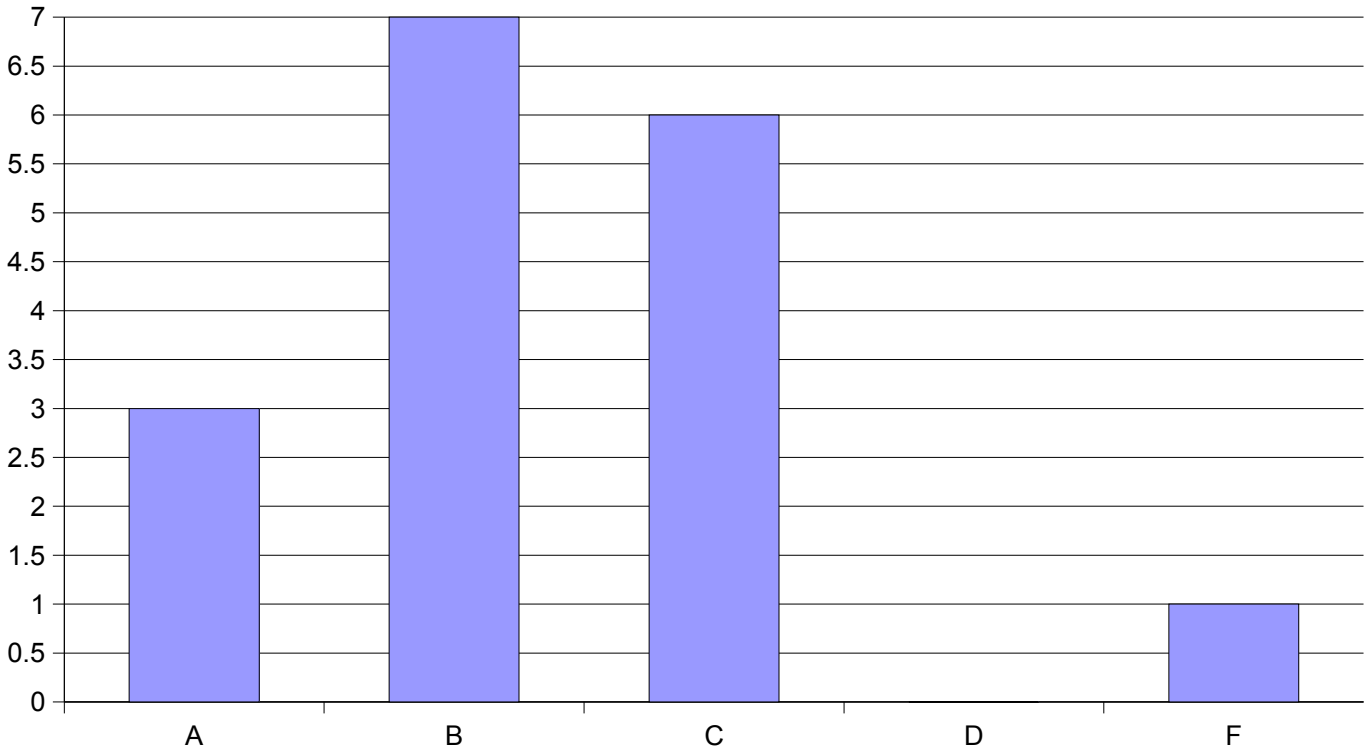
3-23-2004

Opening Discussion

- Do you have any questions about the quiz?
- What did we talk about last class?
- Do you have any questions about the assignment?
- When you have a function in which something goes wrong, how does it tell the rest of the program? Do you always put in code to make sure the functions you call execute properly?

Midterm Results

- The median grade on the exams was exactly an 80.



Testing Code

- Something closely related to today's topic is that of testing code. That is something that you should be doing. In some cases, just running the game does a fairly good test, but it isn't always the easiest one to debug.
- Remember that in Java you can put a main in any class. You should put a main in things like your linked list class where all you do is test and print out stuff.

Error Handling

- In the code that you are probably used to, there are two ways that a function can tell you if an error occurred.
  - Return an error code.
  - Set a flag that should be checked.
- Both of these are very easy to ignore which can lets the errors “propagate” and makes debugging much more difficult.

Enter Exceptions

- An alternate approach to error handling is the use of exceptions. As their name implies, exceptions are things used for exceptional events.
- When an error occurs, the code will "throw" an exception. When an exception is thrown control pops up the stack until code is found to handle it. If no handler is found that thread exits completely.

try, catch, and finally Blocks

- There are 4 syntactic components of dealing with exceptions. The ones you will write most are try and catch blocks.
- When you have a section of code that can have an exception thrown in it that you want to handle, you put it in a try block.
- After the try block you can have one or more catch blocks. Each one specifies a type and catches anything of that type (including subtypes).
- A finally block is like a default. In addition, it ALWAYS happens.

throws and throw

- Sometimes a method can have an exception occur in it, but it doesn't know how to handle it. In this situation, the exception should be added to the throws clause of the method declaration.
- If you want to throw your own exceptions you use the throw statement. It is followed by an object that is the exception to be thrown.

Checked vs. Unchecked Exceptions

- There are two broad categories of exceptions.
  - Checked exceptions must either be caught, or they must appear in the throws clause of the method.
  - Unchecked exceptions don't have to do this and often shouldn't be handled. If an unchecked exception arises it typically signifies a major problem and the code should crash.
- Subtypes of RuntimeException are unchecked.

Benefits of Exceptions

- Exceptions are nice because they can't be ignored. They tell you there is a problem immediately instead of letting the code run on until it has a serious problem or just leaving logic errors.
- The Exception class also has handy methods like printStackTrace() that can be used to help with the debugging process as well.

Code

- Let’s go look in the javadocs at some of the function calls that can throw exceptions then write some code that deals with those exceptions.

Minute Essay

- The line new FileInputStream(fileName) can throw two exceptions. Look those up in the javadocs, and write code that would catch them and print out semi-meaningful messages.
- Assignment #4 is due today.