Exceptions

Dept of Computer Science
University of Maryland College Park

This material is based on material provided by Ben Bederson, Bonnie Dorr, Fawzi Emad, David Mount, Jan Plane
Overview

- Division by zero
- Exceptions
Division By Zero In Java

- In Java floating point division by zero generates infinity
- In Java integer division by zero returns an error (exception)
- **Example:** Division.java
- Why are they treated differently:
  - Because with float division, the answer can be precisely represented and the IEEE standard mandates it
Program Errors – Run Time

- Run-time errors
  - Illegal/Impossible operations to execute
  - Detected during program execution
    - But not detectable at compile time
  - Treated as exceptions in Java

- Example Errors
  - Division by zero
  - Using null reference
  - Illegal format conversion

  **Example:** Division.java (integer option)
What to Do When Errors Occur

- Cry!! 😊
- Alternatives
  - Ignore Error
  - Print error message and terminate
  - Have the method handle the error in the code where the problem lies as best as you can
  - Have the method pass it off to someone else to handle
    - Usually an error code is returned to whoever called the method
  - Modern language approach: Exception!
  - Note: in language like C, a “core dump” takes place
Exceptions

- Rare event outside normal behavior of code
  - Usually a run-time error
- **Examples**
  - Division by zero
  - Access past end of array
  - Out of memory
  - Number input in wrong format (integer vs. float)
  - Unable to write output to file
  - Missing input file
  - Application specific
    - (e.g., attempt to remove a nonexistent customer from a database)
Exceptions

- Notice that you can also define your own exceptions and generate them when your code detect errors.
- When any of the previous errors occur we say that “an exception occurred”.
- We use the phrase “a program throws an exception” to indicate a program generates an exception.
- In Java an exception is represented by an object.
What to Do When an Exception Occurs?

- **Do nothing**
  - Usually your program will be terminated by the JVM
  - Stack trace is printed showing where exception was generated (red and blue in Eclipse window)
  - **Example:** MilesPerGallon.java

- **Do something about it**
  - In some applications aborting the program is not an option
    - Email processor, web/database server
  - We can “handle” the exception by “catching the exception” and defining code to be executed
    - Use `try { }` to enclose code that can potentially throw an exception
    - Use `catch(EXCEPTIONTYPE e) {}` to “catch” the exception and define the code to execute when the exception occurs
    - **Example:** MilesPerGallonV2.java
What to Do When an Exception Occurs?

- Notice that when an exception occurs control “jumps” to the catch clause and code after the point where the exception occurred is not executed.
- If the exception is not thrown the code of the catch clause is ignored.
- Exception Object
  - When an exception is thrown, a new exception object is created, which encodes information about the nature of the exception.
- Every Exception object supports the following methods:
  - `Exception(String message)` Constructor taking an explanation as an argument.
  - `String getMessage()` → Returns a message describing the exception.
  - `void printStackTrace()` → Prints the contents of the Java call stack at the time of the exception.
Exception Propagation

- When an exception occurs, Java looks in the current method for a catch clause that matches the exception. If one is found, the exception is handled; otherwise exception propagation takes place.

- Exception Propagation
  - Java uses exception propagation to look for exception handlers.
  - When an exception occurs, Java pops back up the call stack to each of the calling methods to see whether the exception is being handled in a catch block of the method. This is exception propagation.

- The **first method** it finds that catches the exception will have its catch block executed. **Execution resumes normally** in the method after this catch block.

- If we get all the way back to main and no method catches this exception, Java catches it and **aborts** your program.

- **Example:** Propagation.java
There are different types of exceptions, depending on the error.

- `ArithmeticException` → divide by zero
- `NullPointerException` → attempt to access an object with a null reference
- `IndexOutOfBoundsException` → array or string index out of range
- `ArrayStoreException` → attempting to store an object of type X in an array of type Y
- `EmptyStackException` → attempt to pop an empty Stack (java.util).
- `IOException` → attempt to perform an illegal input/output operation (java.io)
- `NumberFormatException` → attempt to convert an invalid string into a number (e.g., when calling `Integer.parseInt()`)

... 

**Exception**: The most generic type of exception.

- All of these are Objects in Java’s class library (most in java.lang)
Finally Block

- Code that is always run (finally → ALWAYS)
- Run no matter what
  - When no exception has been thrown
  - When exception is thrown
  - If you exit the method via return
    - For example if the return occurs before the finally clause, the clause will be executed and then we will return

- Example: Finally.java