Exceptions

CS151

Chris Pollett

Sept. 26, 2005.

Outline

- What is an exception
- Sources of Exceptions
- Hierarchy of Exceptions
- Exception Syntax

What is an Exception?

- Exceptions are unexpected conditions in programs.
- Java provides a mechanism to facilitate recovery from such unexpected collisions.
- This mechanism disrupts the normal flow of execution and goes to a block of code specifically for handling the exception.

Sources of Exceptions

- When the normal flow of execution is interrupted because of an exception, we say an exception has been *thrown*.
- Exceptions originate from two sources:
  - At run-time. This might happen for instance if one tries to dereference a null pointer.
  - In a Java program, when an unexpected condition occurs, an exception can be explicitly thrown with the *throw* statement.

Hierarchy of Exceptions

- Exceptions are modeled as objects of different exception classes.
- All error and exceptions are subclasses of **Throwable**.
- **Error** is a subclass of Throwable for throwing serious of fatal problems with a program. Errors are thrown by the JVM and are not typically handled by regular programs. Some subclasses include **AssertionError** and **OutOfMemoryError**.
- **Exception** is a subclass of Throwable for problems which might be thrown by a typical program. All-user defined exceptions should be a subclass of Exception. Some notable subclasses are IOException, CloneNotSupportedException, and InterruptedException
- **RuntimeException** is a subclass of Exception which are cause by illegal operation and are thrown by the JVM. Some examples are: ArithmeticException, ClassCastException, IndexOutOfBoundsException, IllegalArgumentException, NullPointerException, and NumberFormatException.

Exception Syntax

- To throw an exception the command is:

  throw *ExceptionName*;

  For example: throw new MyException();
- Any exceptions not caught within a method, but which might be thrown by that method must be listed in the method declaration. For example,

  public void myMethod() throws IOException;
- To handle exception use try-catch block:

  try{/* code which might cause exception*/}

  catch(MyException_1 e1){/* what to do for this type of exception*/}

  ….

  catch(MyException_n en){/* what to do for this type of exception*/}

  finally{/*what to do in all cases including no exception */}
- One common thing to do when an exception occurs to to print the list of stack calls: e.printStackTrace();

Example

```
public class PurchaseOrder
{
    public double calculateItemTotal(double unitPrice, int quantity)
    {
        if(quantity < 0) throw new IllegalArgumentException("negative
          quantity");//exception case
        //normal case
        return unitPrice*quantity;
    }
    // rest of class
}
// Code which might use above:
PurchaseOrder anOrder;
try{double total= anOrder.calculateItemTotal(…);
} catch(IllegalArgumentException e){/* handle exception*/}
```