CS202 Java Object Oriented Programming

Errors and Exception Handling

Chengyu Sun
California State University, Los Angeles

Examples of Exceptions

- ArrayIndexOutOfBoundException

```
int a[] = new int[10];
for( int i=0 ; i <= a.length ; ++i ) a[i] = i;
```

- NumberFormatException

```
String s = "7.3";
int n = Integer.parseInt( s );
```

Runtime Errors

- Errors happened when the program is running, e.g.
  - Access an element outside array boundary
  - Read a file which doesn't exist
  - User input errors
- Runtime error handling
  - Error codes
  - Exceptions

Error Codes

```
// n >= 1

int factorial( int n )
{
   // return an error code
   if( n < 1 ) return –1;

   int fact = 1;
   for( int i=1 ; i <= n ; ++i )
     fact *= i;

   return fact;
}
```

```
int f = factorial(x);

if( f == -1 ) // error handling
{
   System.err.println( "error!");
   System.exit(1);
}
else          // regular code
{
   // do something with f
}
```

Exception – `throw`

```
int factorial( int n ) throws LessThanOneException
{
   if( n < 1 ) throw new LessThanOneException();

   int fact = 1;
   for( int i=1 ; i <= n ; ++i ) fact *= i;
   return fact;
}
```

Exception – `try` and `catch`

```
try       // regular code
{
   int f = factorial(n);
   // do something with f
}
catch( LessThanOneException e ) // error handling
{
     System.err.println( "error!" );
     System.exit(1);
}
```

Use Exceptions

- When an error occurs, throw an exception object
- A method which throws an exception should be declared so using throws ExceptionClassName
- Enclose regular code in a try block
- Enclose error handling code in a catch block

`Exception` Class

- Pre-defined in Java
- All exception classes must inherit from this class
- Important methods
  - `Exception()`
  - `Exception( String msg )`
  - `String getMessage()`
  - `void printStackTrace()`

User Defined Exception Class

```
public class LessThanOneException extends Exception {

    public LessThanOneException() { super(); }

    public LessThanOneException( String msg )
    {
        super(msg);
    }

}
```

Throw Multiple Exceptions

```
public void foo() throws ExceptionA, ExceptionB, ExceptionC
{
    ...
    throw new ExceptionA("Bad thing A");

    ...
    throw new ExceptionB("Bad Thing B");

    ...
    throw new ExceptionC("Bad Thing C");
}
```

Catch Multiple Exceptions

```
try
{
    foo();
}
catch( ExceptionA ea )
{
    // do something
}
catch( ExceptionB eb )
{
    // do something
}
catch( ExceptionC ec )
{
    // do something
}
```

- Or, if we don't need to distinguish exactly which type of exception occurs

```
try
{
    foo();
}
catch( Exception e )
{
    System.err.println( e.getMessage() );
    System.exit(1);
}
```

A More Complex Example

```
readFile {
        open the file;
        determine its size;
        allocate that much memory;
        read the file into memory;
        close the file;
}
```

Handling Errors with Error Code

```
errorCodeType readFile {
   errorCode = 0;
   open the file;
   If (theFileIsOpen) {
      determine the length of the file;
      if (gotTheFileLength) {
         allocate that much memory;
            if (gotEnoughMemory) {
               read the file into memory;
               if (readFailed) { errorCode = -1; }
            }
            else { errorCode = -2; }
      } else { errorCode = -3; }
      close the file;
      if (theFileDidntClose && errorCode == 0) { errorCode = -4; }
      else { errorCode = errorCode and -4; }
   }
   else { errorCode = -5; }
   return errorCode;
}
```

Handling Errors with Exceptions

```
readFile {
   try
   {
      open the file;
      determine its size;
      allocate that much memory;
      read the file into memory;
      close the file;
   }
   catch(fileOpenFailed) { doSomething; }
   catch (sizeDeterminationFailed) { doSomething; }
   catch (memoryAllocationFailed) { doSomething; }
   catch (readFailed) { doSomething; }
   catch (fileCloseFailed) { doSomething; }
}
```

Advantages of Exceptions

- Separate error handling code with from regular code
- Group error types and error differentiation
- Propagate errors up the call stack