Some additional rules and syntax

More about writing code related to exceptions

You can have more than one catch

If some code might throw more than one type of Exception, you are allowed to (and encouraged to) write a separate handler for the various cases:

```
try{
  // some code which might throw IOException
  // some code which might throw NumberFormatException
}catch(IOException ioe){
  // Handle I/O problem
}catch(NumberFormatException nfe){
  // Handle the fact that some text couldn’t be converted to a number
}finally{
  // If you need to, you can always add a finally clause
}
```

Writing a handler for related exceptions

You can also catch more than one type in one catch-clause (if you have good reason to):

```
try{
  //Some code which can throw an SQLException
  //Some code which can throw an IOException
}catch(SQLException|IOException e){
  // Code which can handle both cases
  // (n.b. they should have the same solution!)
}
```

Exception and overriding

When you override a method in a subclass, you cannot declare that the method throws broader exceptions than the version in the superclass you are overriding.

If class A has a method open() which throws FileNotFoundException, you cannot override open() and declare that your version throws something more general (like any superclass to FileNotFoundException).

If this were allowed, then the handlers for calls to the superclass type references wouldn’t work if the actual object throws something else!

Try with resources

It is possible to use a special syntax for the try-block, if you are dealing with resources which implement the AutoCloseable interface (or the subinterface Closeable), you can take advantage of the fact that you know there's a close() method implemented. This will automatically call close() regardless of whether an exception is thrown, just like the finally clause:

```
try(initialize the autocloseable resource){
  // use the resource
}catch(Exception e){
  // handle the exception
}
```

AutoCloseable try-with-resources example

```
import java.io.*;
public class Resources{
  public static void main(String[] args){
    String firstLine = getFirstLine();
    System.out.println("First line of this file: " + firstLine);
  }
  static String getFirstLine(){
    try( BufferedReader in = new BufferedReader
          (new FileReader("Resources.java"))  ){
      return in.readLine(); // use the resource
    }catch(IOException e){
      System.err.println("Couldn't open Resources.java");
      return null;
    }
  }
}
$ javac Resources.java && java Resources
First line of this file: import java.io.*;
```