3. Exception Handling :: Throwing Exceptions

Checked exceptions must be **thrown** or **handled**. In this section we will discuss how to throw a checked exception.

One type of checked exception is *java.io.FileNotFoundException* which is the superclass for all exceptions that arise because of failed input/output operations. For example, suppose we try to open a file for reading that does not exist:

```
public void someFunction() {
  Scanner in = new Scanner(new File("does-not-exist.txt"));
}
```

If *does-not-exist.txt* does not exist then the *Scanner* constructor will throw a *java.io.FileNotFound Exception*. Since this is a checked exception we either have to modify *someFunction*() to **throw** the exception or **write an exception handler**.

3. Exception Handling :: Throwing Exceptions (continued)

```
import java.io.FileNotFoundException;

public class SomeClass {
  public static void main(String[] args) throws FileNotFoundException {
    new SomeClass();
  }

  public SomeClass() throws FileNotFoundException {
    someFunction();
    ...
  }

  public void someFunction() throws FileNotFoundException {
    ...
    someOtherFunction();
    ...
  }

  public void someOtherFunction() throws FileNotFoundException {
    Scanner in = new Scanner(new File("this-file-does-not-exist.txt"));
    ...
  }
}
```

3. Exception Handling :: Catching Exceptions

To write an exception handler for a checked (or unchecked exception) we write a **try/catch** block, which has this syntax:

```
try {
  // code which may throw an exception is here
} catch (SomeExceptionClass e) {
  // code which gets executed when an exception of the SomeExceptionClass is thrown
  // appears here.
} catch (SomeOtherExceptionClass e) {
  // code which gets executed when an exception of the SomeOtherExceptionClass is
  // thrown appears here.
} catch (YetAnotherExceptionClass e) {
  // code which gets executed when an exception of the YetAnotherExceptionClass is
  // thrown appears here.
}
```

3. Exception Handling :: Catching Exceptions

```
import java.io.FileNotFoundException;

public class SomeClass {
  public static void main(String[] args) {
    new SomeClass();
  }

  public SomeClass() {
    someFunction();
    ...
  }

  public void someFunction() {
    try {
      someOtherFunction(fname);
      ...
    } catch (FileNotFoundException e) {
      // Write code here that handles the exception.
    }
  }

  public void someOtherFunction(String fname) throws FileNotFoundException {
    Scanner in = new Scanner(new File(fname));
    ...
  }
}
```

3. Exception Handling :: Finally Clause

A **try block** may have an associated **finally clause:**

```
public void someMethod() {
  try {
    // Code to be executed that may throw an exception.
    // Some resource is acquired here and this resource must be released before
    // someMethod() returns.
  } catch (SomeException e) {
    // Code that handles the exception.
  } catch (SomeOtherException e) {
    // Code that handles the exception.
  } finally {
    // Code that is always executed whether or not an exception occurs appears
    // here. The idea is that the code in the try block may acquire some resource
    // which needs to be released before execution continues.
  }
}
```

3. Exception Handling :: Finally Clause (continued)

For example,

```
// out is declared here so it will be in scope in the finally clause.
PrintWriter out;

// Try to send some stuff to an output file named output.txt.
try {
  out = new PrintWriter(new File("output.txt"));
  out.println("blah blah blah");
  out.println("blah blah blah");
  ...
} catch {FileNotFoundException e) {
  // Code is here that does something to handle the exception.
} finally {
  // This code will be executed whether or not the code in the try block throws an
  // exception. This code guarantees that the output file will be closed.
  if (out != null) {  // out will be nonnull if the PrintWriter ctor succeeded.
    out.close();
  }
}
```