

More Streams

- Handling Exceptions
- Files
- The File class

Introduction to Java



See Also

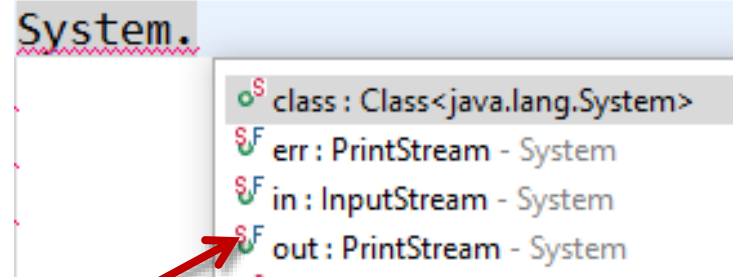
<https://docs.oracle.com/javase/tutorial/essential/io/>

<http://tutorials.jenkov.com/java-io/streams.html>



Standard IO

- System.in is an InputStream
- System.out is a PrintStream
- System.err is a PrintStream



- The OS treats err and out differently. You can pipe them to different places.
- These globals are read-only

Input Scanner

```
InputStream is = new FileInputStream("Test.txt");  
Scanner sc = new Scanner(is);
```

```
Reader r = new FileReader("Test.txt");  
Scanner sc = new Scanner(r);
```

```
sc.close();
```

```
Scanner sc = new Scanner(System.in);
```

- nextLine() : String - Scanner - 27%
- next() : String - Scanner - 24%
- next(Pattern pattern) : String - Scanner
- next(String pattern) : String - Scanner
- nextBigDecimal() : BigDecimal - Scanner
- nextBigInteger() : BigInteger - Scanner
- nextBigInteger(int radix) : BigInteger - Scanner
- nextBoolean() : boolean - Scanner
- nextByte() : byte - Scanner
- nextByte(int radix) : byte - Scanner
- nextDouble() : double - Scanner
- nextFloat() : float - Scanner
- nextInt() : int - Scanner
- nextInt(int radix) : int - Scanner
- nextLong() : long - Scanner
- nextLong(int radix) : long - Scanner
- nextShort() : short - Scanner
- nextShort(int radix) : short - Scanner
- hasNext() : boolean - Scanner - 59%
- hasNextLine() : boolean - Scanner - 14%
- hasNext(Pattern pattern) : boolean - Scanner
- hasNext(String pattern) : boolean - Scanner

IO Exceptions: Test Code

```
import java.io.IOException;

public class TestResource {
    String name;
    boolean onUse;
    boolean onClose;

    public TestResource(String name, boolean onOpen,
        boolean onUse, boolean onClose) throws IOException {

        System.out.println("OPEN: "+name);
        this.name = name;
        this.onUse = onUse;
        this.onClose = onClose;
        if(onOpen) {
            throw new IOException("Open: "+name);
        }
    }

    public void use() throws IOException {
        System.out.println("USE: "+ name);
        if(onUse) {
            throw new IOException("Use: "+ name);
        }
    }
}

    public void close() throws IOException {
        System.out.println("CLOSE: "+ name);
        if(onClose) {
            throw new IOException("Close: "+name);
        }
    }
}
```

IO Exceptions

```
import java.io.IOException;
```

```
public class Tinker {
```

```
    public static void main(String [] args) throws IOException {
```

```
        System.out.println("Start");
```

Create Use Close

```
        TestResource r = new TestResource("MyFile", false, false, false);
```

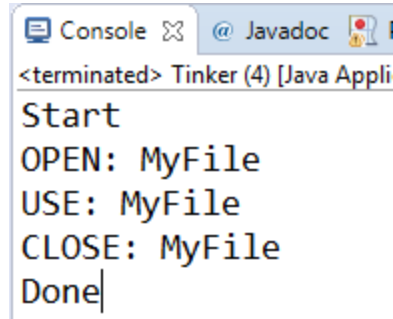
```
        r.use();
```

```
        r.close();
```

```
        System.out.println("Done");
```

```
    }
```

```
}
```




The screenshot shows a Java IDE console window with the following content:

```
Console [X] @ Javadoc [Icon] [Icon]
<terminated> Tinker (4) [Java Appli
Start
OPEN: MyFile
USE: MyFile
CLOSE: MyFile
Done|
```

IO Exceptions

TestResource r = **new** TestResource("MyFile", **false**, **true**, **false**);
r.use();
r.close();

Create Use Close



```
Console  @ Javadoc  Problems  Declaration
<terminated> Tinker (4) [Java Application] C:\Program Files\Java\jdk1.8.0_45\bin\javaw.exe (Aug 9, 2015, 6:08:10 PM)
Start
OPEN: MyFile
USE: MyFile
Exception in thread "main" java.io.IOException: Use: MyFile
    at TestResource.use(TestResource.java:22)
    at Tinker.main(Tinker.java:10)
```

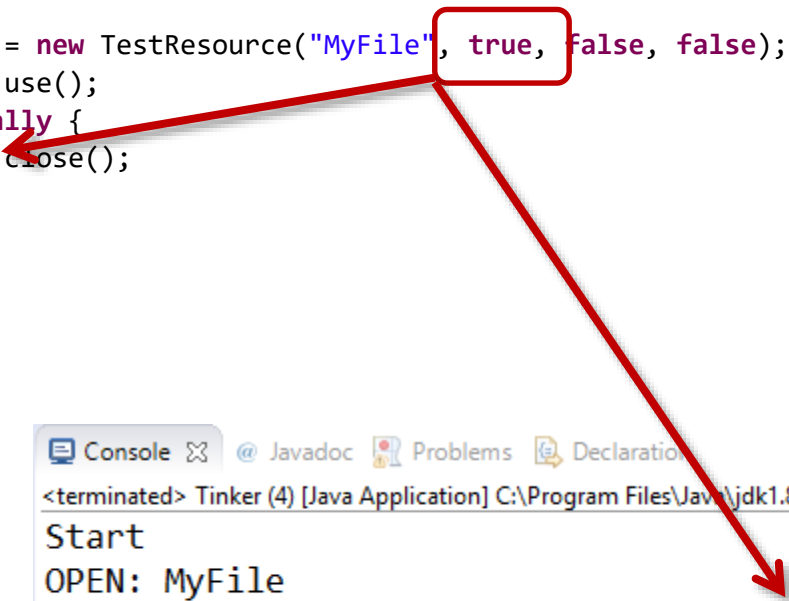

IO Exceptions

```
TestResource r = null;  
try {  
    r = new TestResource("MyFile", true, false, false);  
    r.use();  
} finally {  
    r.close();  
}
```

Create

Use

Close



Console @ Javadoc Problems Declaration

<terminated> Tinker (4) [Java Application] C:\Program Files\Java\jdk1.8.0_45\bin\javaw.exe (Aug 9, 2015, 6:14:02 PM)

Start
OPEN: MyFile
Exception in thread "main" [java.lang.NullPointerException](#)
at Tinker.main([Tinker.java:14](#))

IO Exceptions

```
try {  
    r = new TestResource("MyFile", true, false, false);  
    r.use();  
} finally {  
    if(r!=null) {  
        r.close();  
    }  
}
```

Console @ Javadoc Problems Declaration

<terminated> Tinker (4) [Java Application] C:\Program Files\Java\jdk1.8.0_45\bin\javaw.exe (Aug 9, 2015, 6:16:30 PM)

Start

OPEN: MyFile

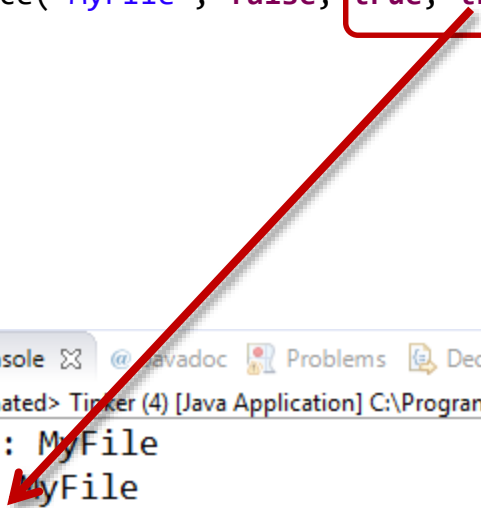
Exception in thread "main" [java.io.IOException](#): Open: MyFile
at [TestResource.<init>\(TestResource.java:15\)](#)
at [Tinker.main\(Tinker.java:11\)](#)

IO Exceptions

```
TestResource r = null;  
try {  
    r = new TestResource("MyFile", false, true, true);  
    r.use();  
} finally {  
    if(r!=null) {  
        r.close();  
    }  
}
```

Create Use Close

true, true



```
Console  @ javadoc  Problems  Declaration  
<terminated> Tinker (4) [Java Application] C:\Program Files\Java\jdk1.8.0_45\bin\javaw.exe (Aug 9, 2015, 6:18:52 PM)  
OPEN: MyFile  
USE: MyFile  
CLOSE: MyFile  
Exception in thread "main" java.io.IOException: Close: MyFile  
    at TestResource.close(TestResource.java:29)  
    at Tinker.main(Tinker.java:15)
```

IO Exceptions

```
TestResource r = null;
try {
    r = new TestResource("MyFile", false, true, true);
    r.use();
} catch (Exception e) {
    if(r!=null) {
        try {
            r.close();
        } catch (Exception e2) {
            e.addSuppressed(e2);
            throw e;
        } finally {
            r = null;
        }
    }
} finally {
    if(r!=null) {
        r.close();
    }
}
```

Create Use Close



```
Console  @ Javadoc  Problems  Declaration
<terminated> Tinker (4) [Java Application] C:\Program Files\Java\jdk1.8.0_45\bin\javaw.exe (Aug 9, 2015, 6:36:30 PM)
Start
OPEN: MyFile
USE: MyFile
CLOSE: MyFile
Exception in thread "main" java.io.IOException: Use: MyFile
    at TestResource.use(TestResource.java:22)
    at Tinker.main(Tinker.java:12)
    Suppressed: java.io.IOException: Close: MyFile
        at TestResource.close(TestResource.java:30)
        at Tinker.main(Tinker.java:16)
```

AutoCloseable

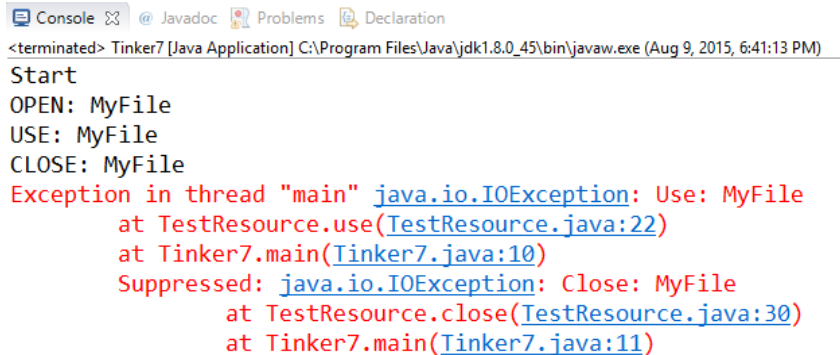
```
System.out.println("Start");
```

Create Use Close

```
try (TestResource r = new TestResource("MyFile", false, true, true)) {  
    r.use();  
}
```

```
System.out.println("Done");
```

```
public class TestResource implements AutoCloseable {  
  
    @Override  
    public void close() throws IOException {  
  
    }  
}
```



```
Console @ Javadoc Problems Declaration  
<terminated> Tinker7 [Java Application] C:\Program Files\Java\jdk1.8.0_45\bin\javaw.exe (Aug 9, 2015, 6:41:13 PM)  
Start  
OPEN: MyFile  
USE: MyFile  
CLOSE: MyFile  
Exception in thread "main" java.io.IOException: Use: MyFile  
    at TestResource.use(TestResource.java:22)  
    at Tinker7.main(Tinker7.java:10)  
Suppressed: java.io.IOException: Close: MyFile  
    at TestResource.close(TestResource.java:30)  
    at Tinker7.main(Tinker7.java:11)
```

Files Utility (Java7)

- Utilities for dealing with files
- Statics in Files and Paths
- Automatically closes files after copy/read/etc.

```
import java.nio.file.Files;  
import java.nio.file.Paths;
```

```
List<String> lines =  
    Files.readAllLines(Paths.get("Test.txt"));
```

```
System.out.println(lines);
```

```
class : Class<java.nio.file.Files>  
copy(Path source, OutputStream out) : long - java.nio.file.Files  
copy(InputStream in, Path target, CopyOption... options) : lon  
copy(Path source, Path target, CopyOption... options) : Path -  
createDirectories(Path dir, FileAttribute<?>... attrs) : Path - java  
createDirectory(Path dir, FileAttribute<?>... attrs) : Path - java.  
createFile(Path path, FileAttribute<?>... attrs) : Path - java.nio.  
createLink(Path link, Path existing) : Path - java.nio.file.Files  
createSymbolicLink(Path link, Path target, FileAttribute<?>... a  
createTempDirectory(String prefix, FileAttribute<?>... attrs) : P  
createTempDirectory(Path dir, String prefix, FileAttribute<?>...  
createTempFile(String prefix, String suffix, FileAttribute<?>... a  
createTempFile(Path dir, String prefix, String suffix, FileAttrib  
delete(Path path) : void - java.nio.file.Files  
deleteIfExists(Path path) : boolean - java.nio.file.Files  
exists(Path path, LinkOption... options) : boolean - java.nio.file  
find(Path start, int maxDepth, BiPredicate<Path, BasicFileAttrib  
getAttribute(Path path, String attribute, LinkOption... options)  
getFileAttributeView(Path path, Class<V> type, LinkOption... c  
getFileStore(Path path) : FileStore - java.nio.file.Files  
getLastModifiedTime(Path path, LinkOption... options) : FileTi  
getOwner(Path path, LinkOption... options) : UserPrincipal - ja  
getPosixFilePermissions(Path path, LinkOption... options) : Set  
isDirectory(Path path, LinkOption... options) : boolean - java.n  
isExecutable(Path path) : boolean - java.nio.file.Files  
isHidden(Path path) : boolean - java.nio.file.Files  
isReadable(Path path) : boolean - java.nio.file.Files  
isRegularFile(Path path, LinkOption... options) : boolean - java  
isSameFile(Path path, Path path2) : boolean - java.nio.file.Files  
isSymbolicLink(Path path) : boolean - java.nio.file.Files  
isWritable(Path path) : boolean - java.nio.file.Files
```

File

- The “File” class
- Making directories, etc.
- Deleting files

```
File f = new File("Test.txt");  
f.
```

- exists() : boolean - File - 47%
- isDirectory() : boolean - File - 13%
- isAbsolute() : boolean - File - 10%
- mkdirs() : boolean - File - 10%
- getAbsolutePath() : String - File - 8%
- delete() : boolean - File - 7%
- canExecute() : boolean - File
- canRead() : boolean - File
- canWrite() : boolean - File
- compareTo(File pathname) : int - File
- createNewFile() : boolean - File
- deleteOnExit() : void - File



Tinkering

- Modify your “capitalizer” to use the try-with-resource
- Use “Files” to copy a text file to a new name
- Print a list of files and directories in the root directory of your disk

