File I/O and Exceptions

April 5, 2012

CMPSCI 121, Spring 2012

Introduction to Problem Solving with Computers

Prof. Learned-Miller

File I/O

Exceptions

- Sometimes we try things in our programs and they don’t work:
  - double x= 3.0/0.0;
    - can’t divide by 0.
  - openFile(“foo”);
    - file foo may not exist
  - storeFile(myData,”foo”);
    - hard disk may be full!

In the old days

- Two choices:
  - Anticipate errors:
    - if (i!=0)
      - x= 3.0/i;
  - Rely on special codes returned from methods:
    - if (openFile(“foo”)==NULL) {
      - Print(“File foo does not exist.”);

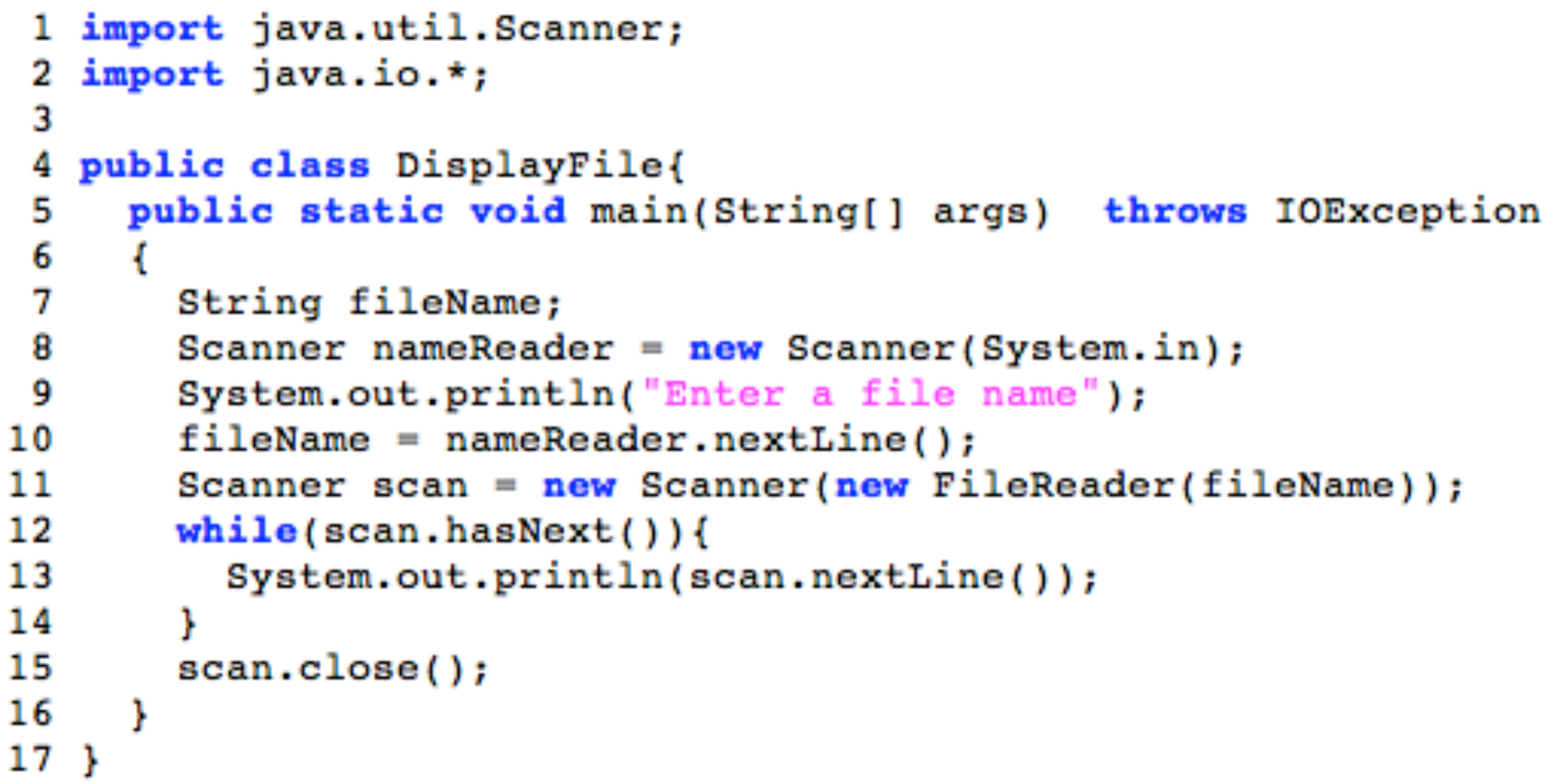
Exceptions

- The exception (error) handling mechanism in Java gives us some extra strategies for dealing with these situations.
  - checked exceptions
    - *must* be handled by application, otherwise won't compile
      - *example: reading a file (non-existent)*
  - unchecked exceptions
    - May or may not be handled
      - example: divide by 0.

File names

- C:\cs121\assignments\assign1.txt
- MyDisk/cs121/assignments/assign1.txt

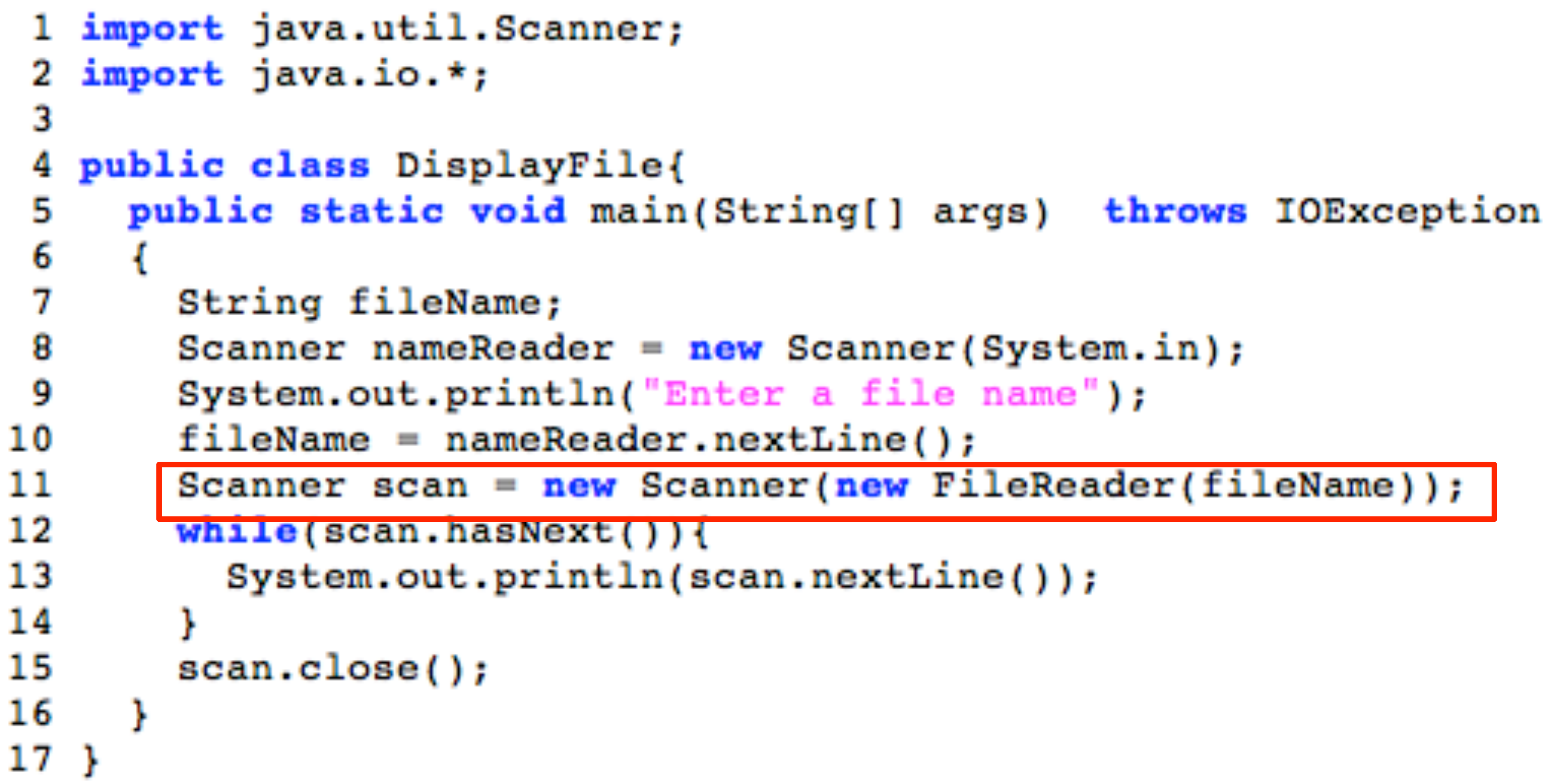
Reading from a file



```java
import java.util.Scanner;
import java.io.*;

public class DisplayFile{
  public static void main(String[] args)  throws IOException
  {
    String fileName;
    Scanner nameReader = new Scanner(System.in);
    System.out.println("Enter a file name");
    fileName = nameReader.nextLine();
    Scanner scan = new Scanner(new FileReader(fileName));
    while(scan.hasNext()){
      System.out.println(scan.nextLine());
    }
    scan.close();
  }
}
```

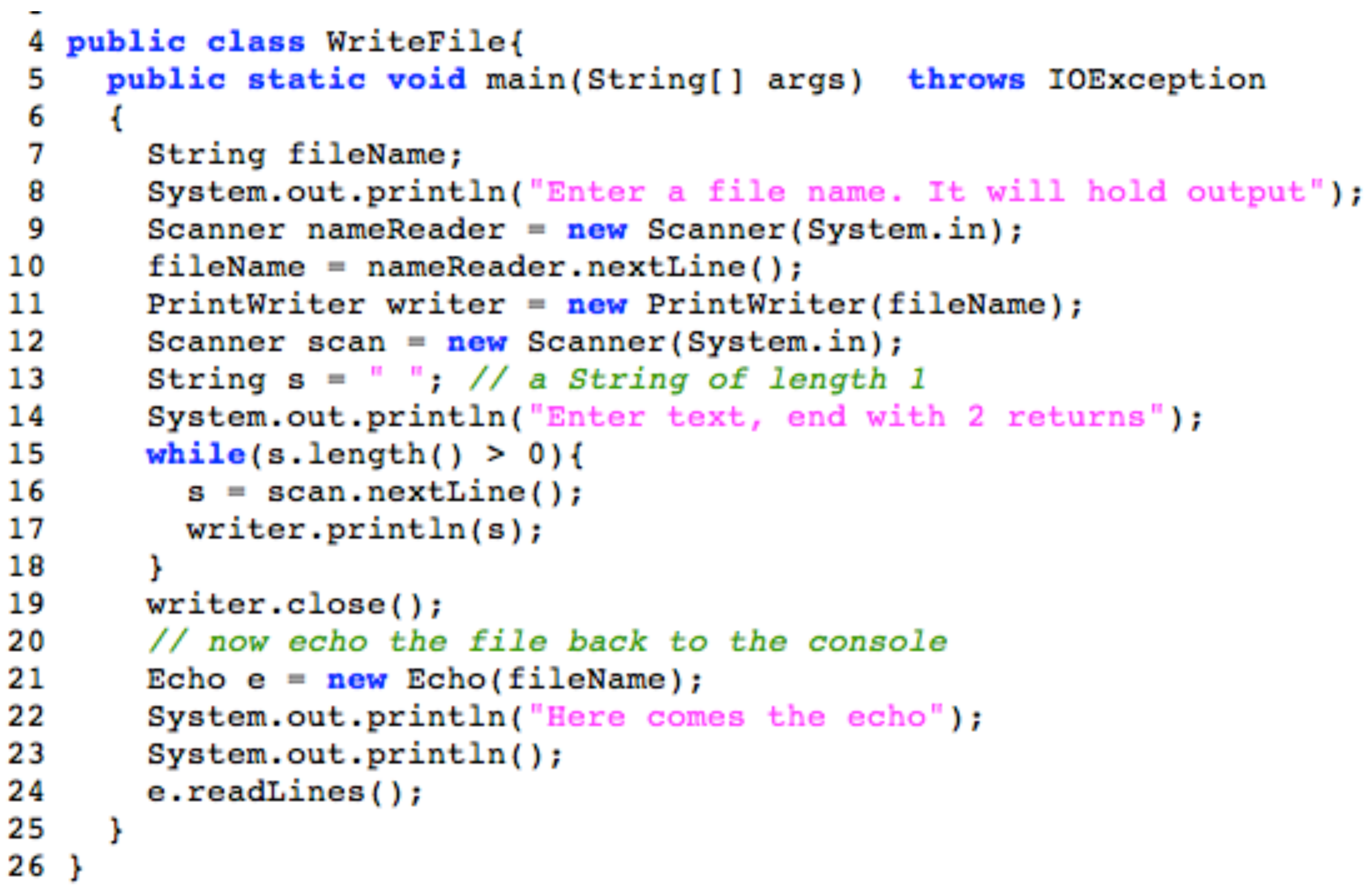
Reading from a file



```java
import java.util.Scanner;
import java.io.*;

public class DisplayFile{
  public static void main(String[] args)  throws IOException
  {
    String fileName;
    Scanner nameReader = new Scanner(System.in);
    System.out.println("Enter a file name");
    fileName = nameReader.nextLine();
    Scanner scan = new Scanner(new FileReader(fileName));
    while(scan.hasNext()){
      System.out.println(scan.nextLine());
    }
    scan.close();
  }
}
```

Writing a new file



```java
public class WriteFile{
  public static void main(String[] args)  throws IOException
  {
    String fileName;
    System.out.println("Enter a file name. It will hold output");
    Scanner nameReader = new Scanner(System.in);
    fileName = nameReader.nextLine();
    PrintWriter writer = new PrintWriter(fileName);
    Scanner scan = new Scanner(System.in);
    String s = " "; // a String of length 1
    System.out.println("Enter text, end with 2 returns");
    while(s.length() > 0){
      s = scan.nextLine();
      writer.println(s);
    }
    writer.close();
    // now echo the file back to the console
    Echo e = new Echo(fileName);
    System.out.println("Here comes the echo");
    System.out.println();
    e.readLines();
  }
}
```

Back to exceptions

```java
import java.io.*;

public class Except0{

  public static void main(String[] args){
    int k; int a = 3; ; int b = 0;
    k = a/b;
  }
}
```

Try and catch

```
import java.io.*;

public class Except1{

  public static void main(String[] args){
    int k; int a = 3;  int b = 0;
    try{
      k = a/b;
      System.out.println("this statement will not execute");
    }
    catch(ArithmeticException e){
      System.out.println("reached here directly from k=a/b: " + e);
    }
  }
}
```

The stack

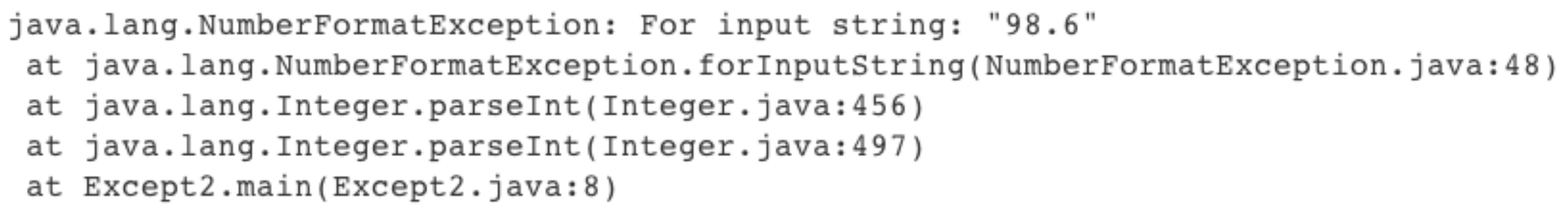
```java
public class Except2{

  public static void main(String[] args){
    String s = "98.6";
    int n;
    try{
      n = Integer.parseInt(s);
      System.out.println(n*n);        }
    catch(Exception e)
    {
      e.printStackTrace();
    }
  }
}
```

The Call Stack

- main()
  - myFamily.printInfo();
    - myInfant.printInfo();
      - System.out.println(...);

Actual “stack trace dump”



```
java.lang.NumberFormatException: For input string: "98.6"
 at java.lang.NumberFormatException.forInputString(NumberFormatException.java:48)
 at java.lang.Integer.parseInt(Integer.java:456)
 at java.lang.Integer.parseInt(Integer.java:497)
 at Except2.main(Except2.java:8)
```

Throwing an exception

```java
import java.io.*;
import java.util.*;

public class IntegerInput{

  public static void main(String[] args){
    int n = -1;
    Scanner scan = new Scanner(System.in);
    while (n < 0) {
      System.out.println("enter your age");
      try {
        if (scan.hasNextInt())
          n = scan.nextInt();
        else {  // non integer submitted
          String userInput = scan.next();
          throw new Exception("Bad input. "+ userInput +
                              " is not an integer.  You must input an integer");
        }
      }
      catch (Exception e)
      {        System.out.println(e.getMessage());      }
    }
    System.out.println("next year you will be " + (n + 1));
  }
}
```

DrJava interlude

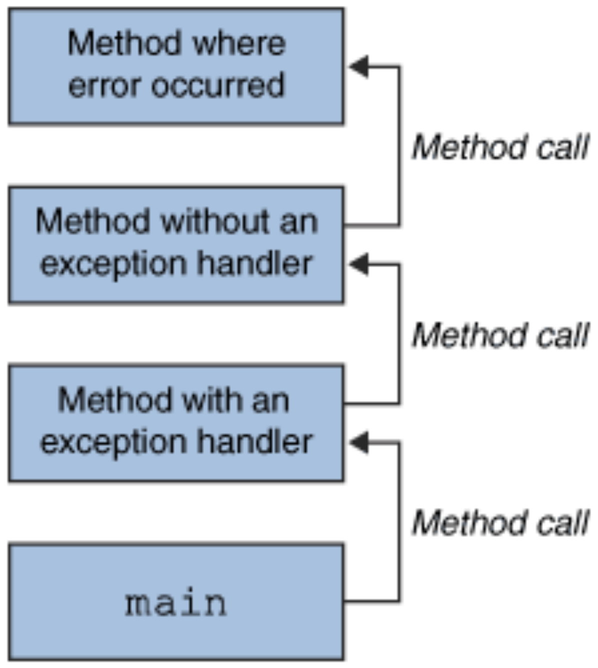
Who throws an exception

- code that someone else wrote:
  - divide by zero
  - open a file
- code that you wrote
  - getting input from the user (someone put in a negative age)

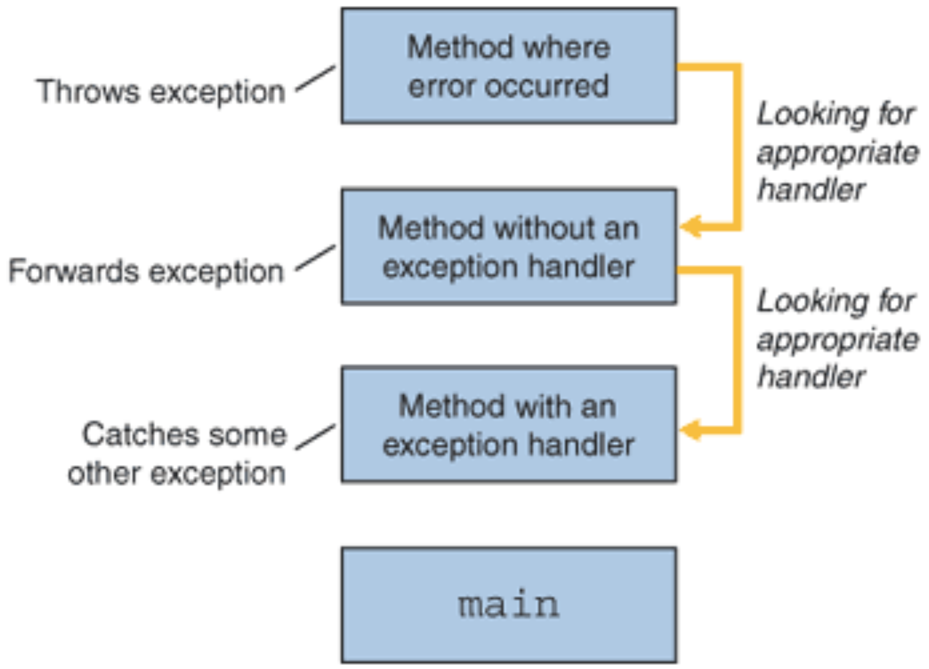
What happens when an exception is thrown?

- 2 basic possibilities:
  - The program stops (crashes).
  - The program doesn’t stop.
    - For the program to keep going, the exception must be “caught”.
    - It can be caught by:
      - The same method in which it was thrown, or
      - one of the calling methods, all the way back to main

The Call Stack


Method where error occurred
Method call
Method without an exception handler
Method call
Method with an exception handler
Method call
main

"Unwinding the stack"

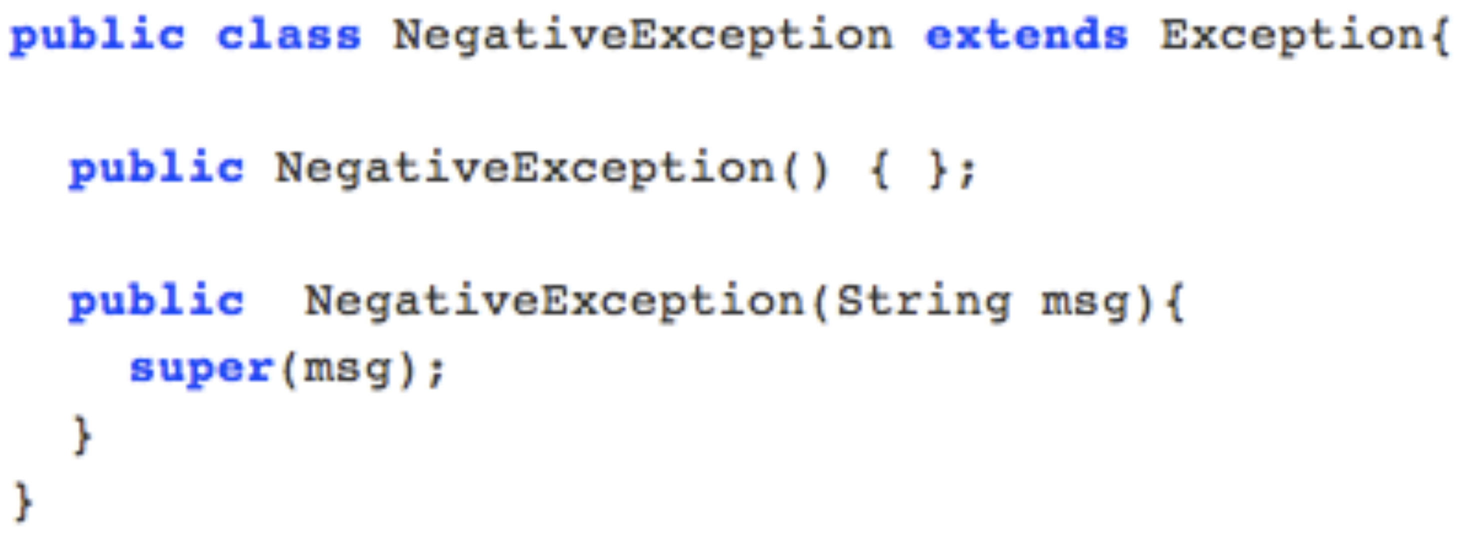

Throws exception
Method where error occurred
Looking for appropriate handler
Forwards exception
Method without an exception handler
Looking for appropriate handler
Catches some other exception
Method with an exception handler
main

Defining new types of exceptions

```java
import java.io.*;
import java.util.*;

public class PositiveInput{
  public static void main(String[] args)
  {    int n = -1;
    Scanner scan = new Scanner(System.in);
    while (n < 0) {
      System.out.println("enter your age");
      try {
        if (scan.hasNextInt())
          n = scan.nextInt();
        else {  // non integer submitted
          String userInput = scan.next();
          throw new Exception("Bad input. "+ userInput +
                                        "is not an integer.  You must input an integer");
        }
        if (n < 0) throw new NegativeException();
      }
      catch (NegativeException e)
      {         System.out.println("age must be >= 0");       }
      catch (Exception e)
      {         System.out.println(e.getMessage());       }
    }  // end while
    System.out.println("next year you will be " + (n + 1));
  }
}
```

Creating a new type of exception



```java
public class NegativeException extends Exception{

  public NegativeException() { };

  public  NegativeException(String msg){
    super(msg);
  }
}
```

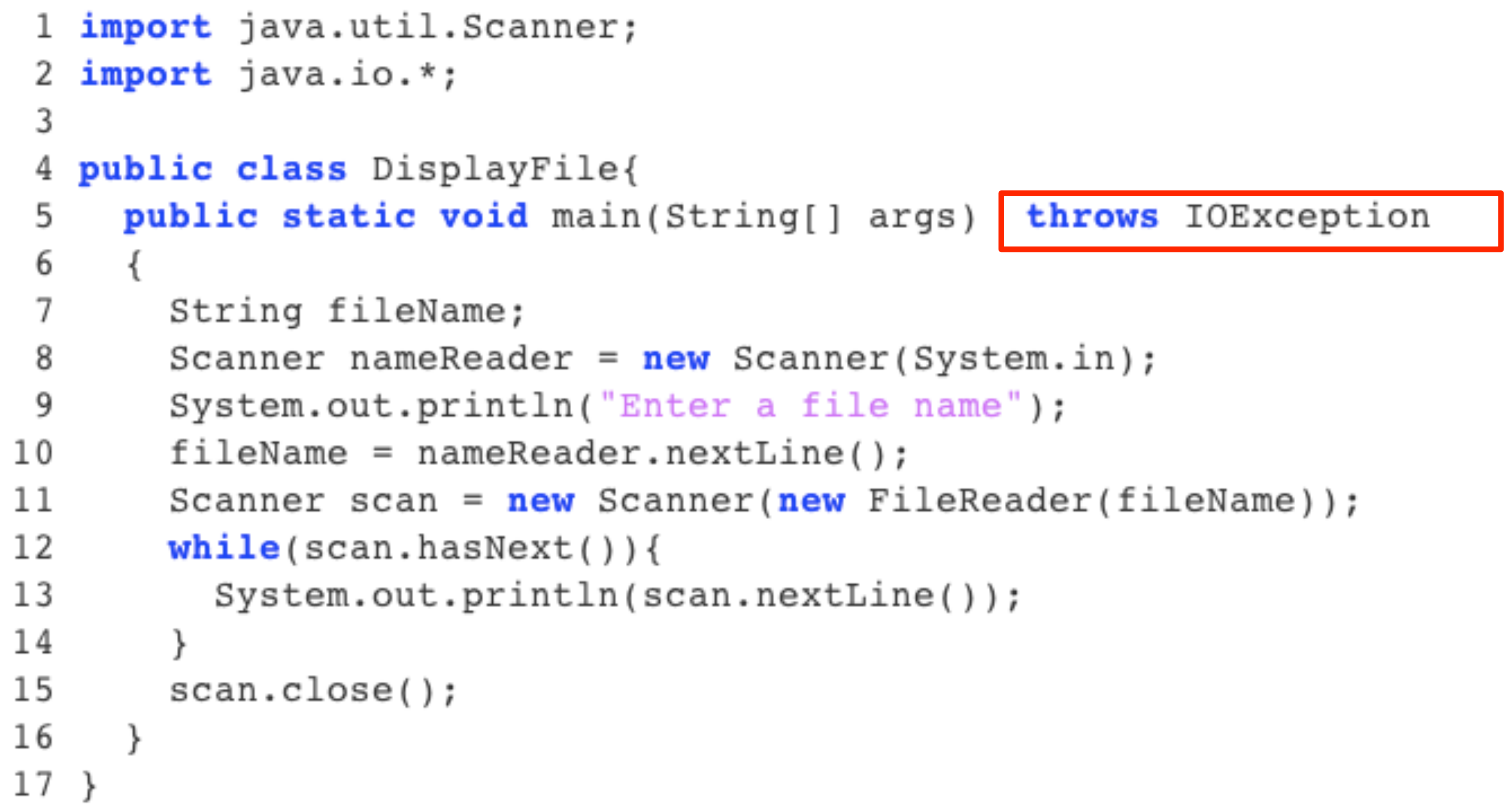
What benefit does new type of exception have?

Lazy way to deal with checked exceptions

```java
import java.util.Scanner;
import java.io.*;

public class DisplayFile{
  public static void main(String[] args)  throws IOException
  {
    String fileName;
    Scanner nameReader = new Scanner(System.in);
    System.out.println("Enter a file name");
    fileName = nameReader.nextLine();
    Scanner scan = new Scanner(new FileReader(fileName));
    while(scan.hasNext()){
      System.out.println(scan.nextLine());
    }
    scan.close();
  }
}
```

Lazy way to deal with checked exceptions



```java
import java.util.Scanner;
import java.io.*;

public class DisplayFile{
  public static void main(String[] args) throws IOException
  {
    String fileName;
    Scanner nameReader = new Scanner(System.in);
    System.out.println("Enter a file name");
    fileName = nameReader.nextLine();
    Scanner scan = new Scanner(new FileReader(fileName));
    while(scan.hasNext()){
      System.out.println(scan.nextLine());
    }
    scan.close();
  }
}
```

```java
//Note: This class won't compile by design!
import java.io.*;
import java.util.Vector;

public class ListOfNumbers {

    private Vector vector;
    private static final int SIZE = 10;

    public ListOfNumbers () {
        vector = new Vector(SIZE);
        for (int i = 0; i < SIZE; i++) {
            vector.addElement(new Integer(i));
        }
    }

    public void writeList() {
        PrintWriter out = new PrintWriter(
                                    new FileWriter("OutFile.txt"));

        for (int i = 0; i < SIZE; i++) {
            out.println("Value at: " + i + " = " +
                                 vector.elementAt(i));
        }

        out.close();
    }
}
```

```
private Vector vector;
private static final int SIZE = 10;

PrintWriter out = null;

try {
    System.out.println("Entered try statement");
    out = new PrintWriter(new FileWriter("OutFile.txt"));
    for (int i = 0; i < SIZE; i++) {
        out.println("Value at: " + i + " = "
                    + vector.elementAt(i));
    }
}
catch and finally statements . . .
```

```
try {

} catch (ExceptionType name) {

} catch (ExceptionType name) {

}
```

```
try {

} catch (FileNotFoundException e) {
    System.err.println("FileNotFoundException: "
                        + e.getMessage());
    throw new SampleException(e);

} catch (IOException e) {
    System.err.println("Caught IOException: "
                        + e.getMessage());
}
```