Exceptions and File I/O

CS111 Computer Programming

Department of Computer Science
Wellesley College

Exceptions in Java

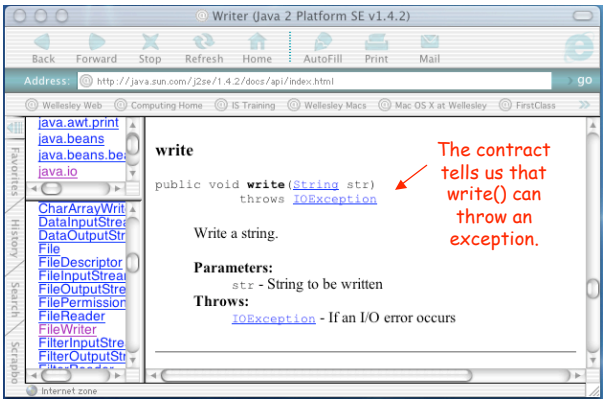
- For **reliability:** a means of describing and recovering from run time errors.
- When unusual condition arises, program **throws** an exception. Code that responds to the situation **catches** the exception.
- Exceptions are objects whose classes inherit from the `Throwable` class.
- **Unchecked** exceptions (subclasses of `Error` or `RuntimeException`)
- **Checked** exceptions must be acknowledged.

List Exceptions

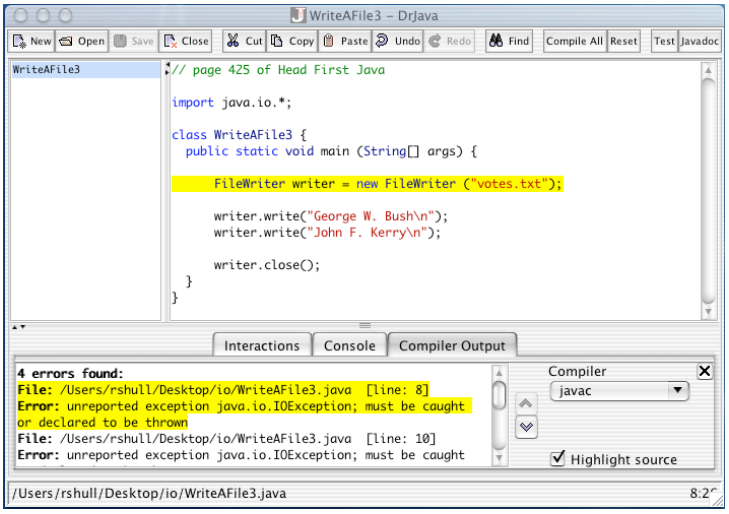
```
class ListException extends RuntimeException
{
      public ListException (String message)
      {
            super(message);
      }
}

class EmptyIntList extends IntList
{
      ...
      public int head ()
      {
            throw new ListException("Cannot take head of empty list");
      }

      public IntList tail ()
      {
            throw new ListException("Cannot take tail of empty list");
      }
      ...
}
```


When good programs go bad

Methods in java use exceptions to tell the calling code, "Something bad happened. I failed."



Risky business

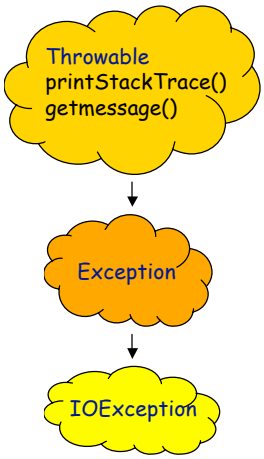


Humoring Java

- The compiler needs to know that YOU know you're calling a risky method.
- You can pass the problem along to your caller
- Or you can wrap the risky code in something called a try/catch to make the compiler relax.

An exception is an object*


```
try {

  // do risky thing

} catch (Exception ex) {

  // try to recover
  // block runs only
  // if Exception is
  // thrown

}
```



*Of type Exception.

Example: Catching an exception

```
public static isMember(int n, IntList lst)
{
     try {
          return (n == head(lst)) || isMember(tail(lst));
     } catch (ListException e) {
          return false;
     }
}
```

*This is generally considered poor programming practice.

File I/O: Election 2008

- Electronic voting is very much in the news.
- After the citizens vote, the results must be written to a file for safe keeping.
- How do we do this?

Writing a String to a text file

```
import java.io.*;

class WriteAFile
{
  public static void main (String[] args)
  {
    FileWriter writer = new FileWriter ("votes.txt");

    writer.write( "George W. Bush\n" );
    writer.write( "John F. Kerry\n" );

    writer.close();
  }
}
```

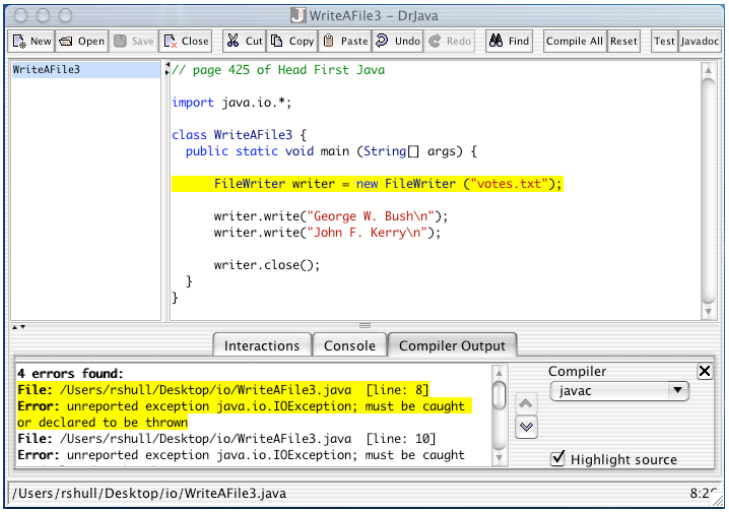
FileWriter lives in java.io package

If the file "votes.txt" does not exist, FileWriter will create it

The write() method takes a String

Tidy up when done

Risky business



Guarding against failure

```
import java.io.*;

class WriteAFile
{
  public static void main (String[] args)
  {
    try {
      FileWriter writer = new FileWriter (”votes.txt");


      writer.write(”George W. Bush\n”);
      writer.write(”John F. Kerry\n”);

      writer.close();

    } catch (IOException ex) {
      System.out.println(“Voter error. Inform Supreme Court.”);
      ex.printStackTrace();
    }
  }
}
```

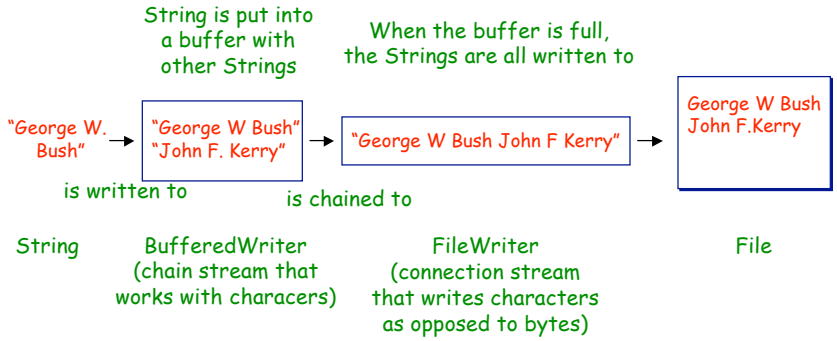
BufferedWriter

```
import java.io.*;

class WriteAFile
{
  public static void main (String[] args)
  {
    try {
      BufferedWriter writer =
             new BufferedWriter(new FileWriter ("votes.txt" ));

      writer.write("George W. Bush\n");
      writer.write("John F. Kerry\n");

      writer.close();

    } catch (IOException ex) {
      System.out.println("Voter error. Inform Supreme Court.");
      ex.printStackTrace();
    }
  }
}
```

We chain a BufferedWriter to FileWriter for more efficient I/O.

The beauty of buffers

File I/O without buffers is like shopping without a cart.



Reading from a text file

```
import java.io.*;
class ReadAFile
{
  public static void main (String[] args)
  {
    try {
      File myFile = new File( "votes.txt" );

      FileReader fileReader = new FileReader(myFile);

      BufferedReader reader = new BufferedReader(fileReader);

      String line = null; // holds each input line
      while ((line = reader.readLine()) != null) {
        System.out.println(line);
      }
      reader.close();
    } catch (Exception ex) {
      ex.printStackTrace();
    }
  }
}
```

FileReader is a connection stream for characters connecting to text file

Wrap in try/catch

Chain fileReader to Buffered Reader for more efficient reading

A magic eight ball

- We write an "eight ball" program and answers yes/no questions.
- The eight ball keeps responses like

  ```
  Without a doubt.
  It seems unlikely.
  My sources say no.
  It is certain.
  Concentrate and ask again.
  ```

  in a file named `answers.txt`

Magic8Ball

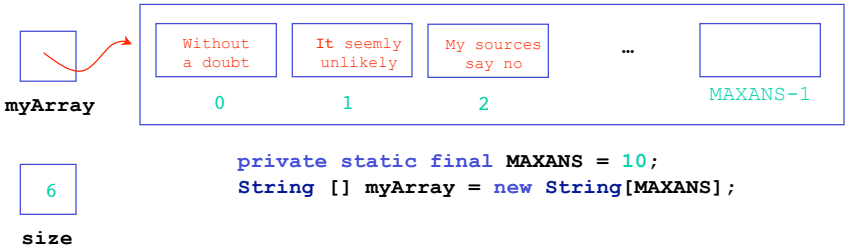
```
import java.awt.*;
import java.applet.*;
public class Magic8Ball
{
  public static void main (String[] args)
  {
      System.out.println(StringChooser.chooseLine("answers.txt"));

  }
}
```

StringChoose.chooseLine()
reads responses from
"answers.txt", then
randomly returns one

First we load our answers

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

public class StringChooser
{
  public static String chooseLine (String inFile)
  {
      try {
          File myFile = new File(inFile);
          BufferedReader reader =
              new BufferedReader(new FileReader (myFile));

          // But where do we put them all?


          reader.close();
      } catch (Exception ex) {
          ex.printStackTrace();
      }
      return "Ask me again tomorrow";     // Got to return something
  }
}
```

An array might be nice*




```
private static final MAXANS = 10;
String [] myArray = new String[MAXANS];
```

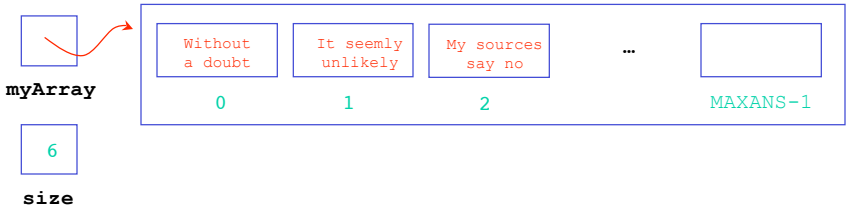
*Why?

Stuffing the answers into an array

```
public class StringChooser
{
  private static final MAXANS = 10;       // Max number of answers allowed
  public static String chooseLine (String inFile)
  {
    String [] myArray = new String[MAXANS]; // Stuff into a String array
    try {
      File myFile = new File(inFile);
      BufferedReader reader =
        new BufferedReader(new FileReader (myFile));

      String line = null;
      int size = 0;
      while ((line = reader.readLine()) != null) {
        myArray[size] = line;
        size++;
      }

      reader.close();
    } catch (Exception ex) {
      ex.printStackTrace();
    }
    return "Ask me again tomorrow"; }
}
```

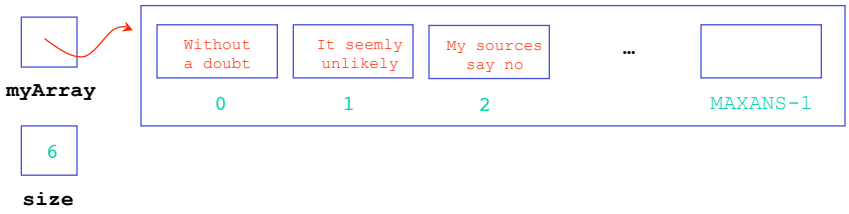
Focus please!

```
…
private static final MAXANS = 10;  // Max number answers allowed
…
 String [] myArray = new String[MAXANS];
…
  String line = null;
  int size = 0;
  while ((line = reader.readLine()) != null) {
    myArray[size] = line;
    size++;
  }
```



Avoiding array out of bounds

```
…
private static final MAXANS = 10;  // Max number answers allowed
…
  String [] myArray = new String[MAXANS];
…
   String line = null;
   int size = 0;
   while (((line = reader.readLine()) != null) && (size < MAXANS)){
     myArray[size] = line;
     size++;
   }
```



Rolling the dice

```
public class StringChooser
{
  // Stuff from previous slides
  private static Randomizer rzer =
    new Randomizer(System.currentTimeMillis());  // Our dice

  public static String chooseLine (String inFile)
  {
    try {
      // Stuff in the try block to fill up the array

      return (String) (myArray[rzer.intBetween(0,size-1)]);

    } catch (Exception ex) {
      ex.printStackTrace();
    }
    // Rest of program
  }
}
```

Returns time in milliseconds for a random seed

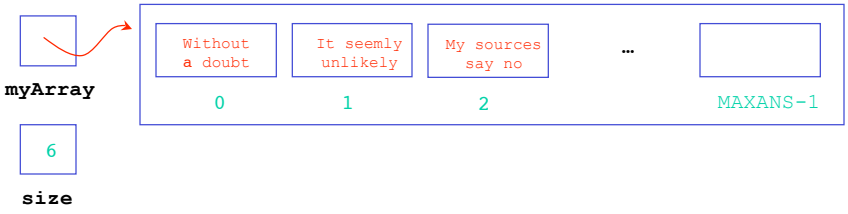
Returns a random integer between and size-1

Pick a number

```
import java.util.*;
public class Randomizer
{
  // Instance Variable:
  private Random rand;

  // Constructor Method:
  public Randomizer (long seed)
  {
    rand = new Random(seed);   // Set seed for repeatability.
  }
  public int intBetween (int lo, int hi)
  {
    int result = lo + (Math.abs(rand.nextInt()) % (hi + 1 - lo));
    return result;
  }
}
```

The dice are tossed

```
private static Randomizer rzer =
  new Randomizer(System.currentTimeMillis());

private static chooseLine (String filename)
{
  // lots of stuff in between
  return (String) (myArray[rzer.intBetween(0, size-1)]);
}

// meanwhile in the Randomizer class
public int intBetween (int lo, int hi)
{
    int result = lo + (Math.abs(rand.nextInt()) % (hi + 1 - lo));
    return result;
}
```



Putting it all together

```
public class StringChooser
{
  private static final MAXANS = 10;  // Max number of answers allowed
  private static Randomizer rzer =
    new Randomizer(System.currentTimeMillis());  // Our dice
  public static String chooseLine (String inFile)
  {
    String [] myArray = new String[MAXANS];
    try {
      File myFile = new File(inFile);
      BufferedReader reader =
        new BufferedReader(new FileReader (myFile));
      String line = null;
      int size = 0;
      while (((line = reader.readLine()) != null) && (size < MAXANS)) {
        myArray[size] = line;
        size++;
      }
      reader.close();
      return (String) (myArray[rzer.intBetween(0,size-1)]);
    } catch (Exception ex) {
      ex.printStackTrace();
    }
    return "Ask me again tomorrow";
  }
}
```