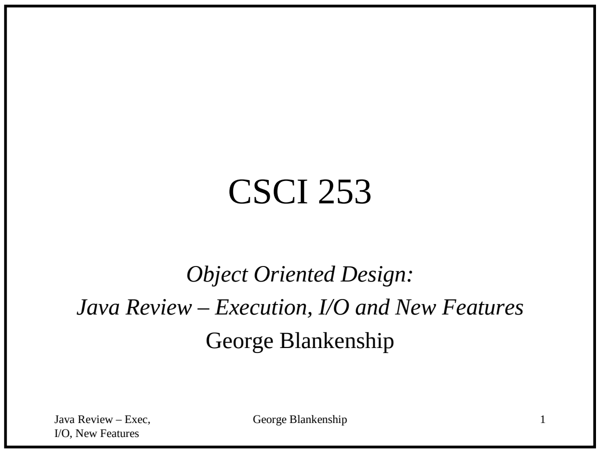
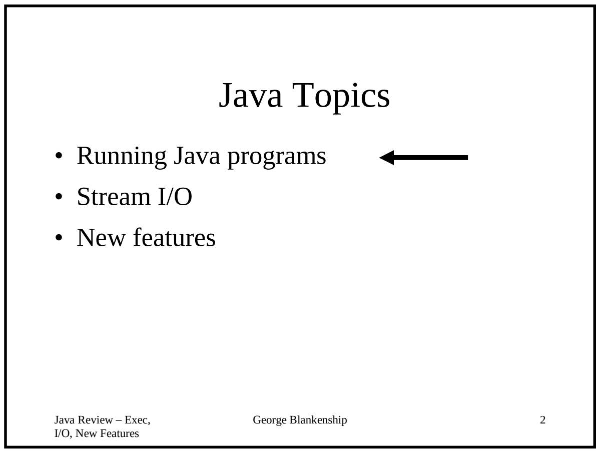


CSCI 253

Object Oriented Design:
Java Review – Execution, I/O and New Features
George Blankenship

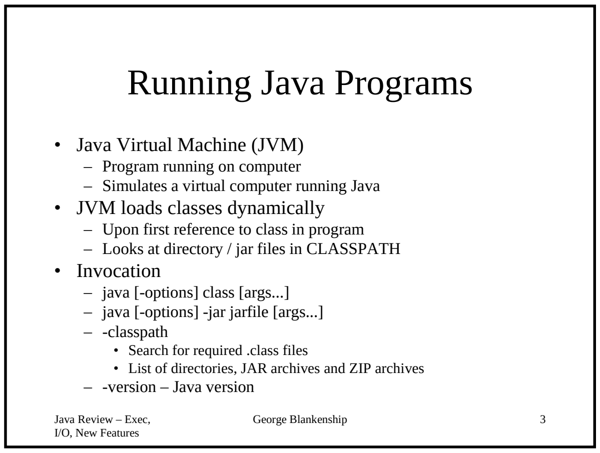
Java Review – Exec, I/O, New Features George Blankenship 1



Java Topics

- Running Java programs ←
- Stream I/O
- New features

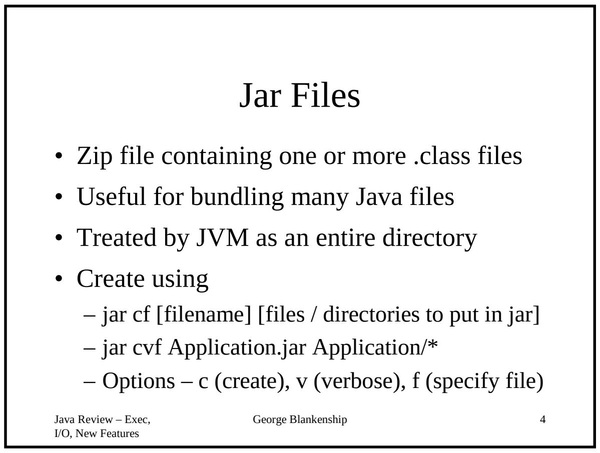
Java Review – Exec, I/O, New Features George Blankenship 2



Running Java Programs

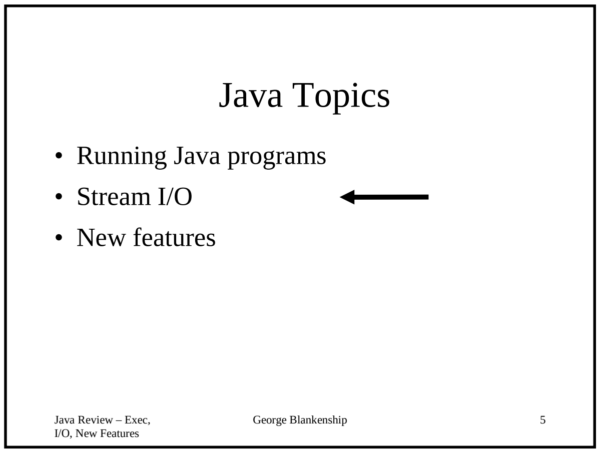
- Java Virtual Machine (JVM)
  - Program running on computer
  - Simulates a virtual computer running Java
- JVM loads classes dynamically
  - Upon first reference to class in program
  - Looks at directory / jar files in CLASSPATH
- Invocation
  - java [-options] class [args...]
  - java [-options] -jar jarfile [args...]
  - -classpath
    - Search for required .class files
    - List of directories, JAR archives and ZIP archives
  - -version – Java version

Java Review – Exec, I/O, New Features George Blankenship 3



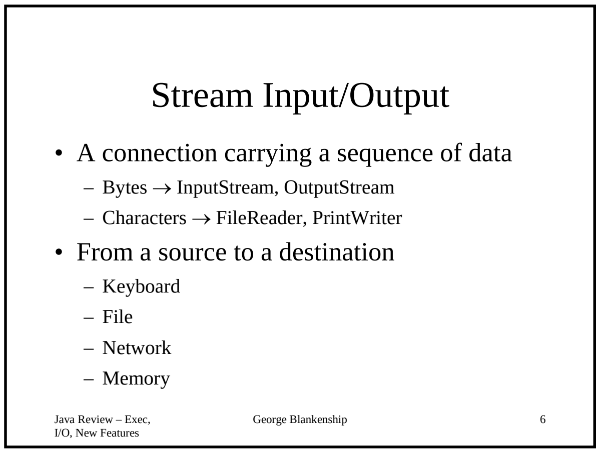
Jar Files

- Zip file containing one or more .class files
- Useful for bundling many Java files
- Treated by JVM as an entire directory
- Create using
  - jar cf [filename] [files / directories to put in jar]
  - jar cvf Application.jar Application/*
  - Options – c (create), v (verbose), f (specify file)



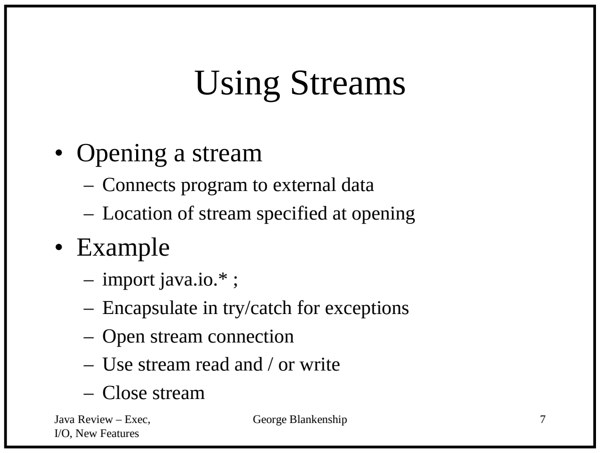
Java Topics

- Running Java programs
- Stream I/O $\longleftarrow$
- New features



Stream Input/Output

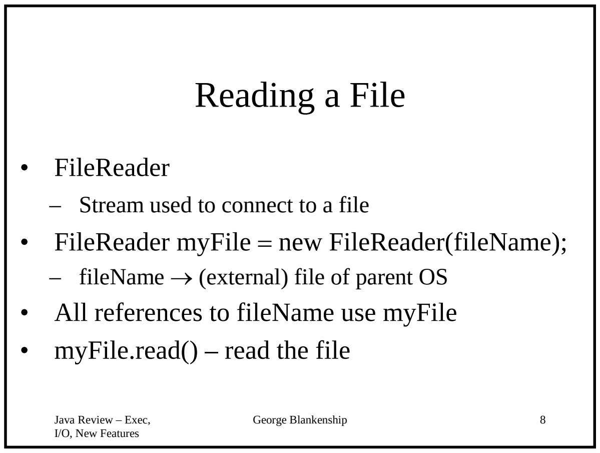
- A connection carrying a sequence of data
  - Bytes $\rightarrow$ InputStream, OutputStream
  - Characters $\rightarrow$ FileReader, PrintWriter
- From a source to a destination
  - Keyboard
  - File
  - Network
  - Memory



Using Streams

- Opening a stream
  - Connects program to external data
  - Location of stream specified at opening
- Example
  - import java.io.* ;
  - Encapsulate in try/catch for exceptions
  - Open stream connection
  - Use stream read and / or write
  - Close stream

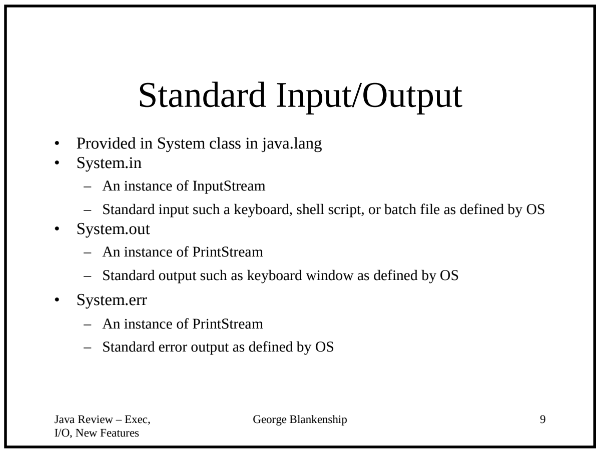
Java Review – Exec, I/O, New Features George Blankenship 7



Reading a File

- FileReader
  - Stream used to connect to a file
- FileReader myFile = new FileReader(fileName);
  - fileName → (external) file of parent OS
- All references to fileName use myFile
- myFile.read() – read the file

Java Review – Exec, I/O, New Features George Blankenship 8



Standard Input/Output

- Provided in System class in java.lang
- System.in
  - An instance of InputStream
  - Standard input such a keyboard, shell script, or batch file as defined by OS
- System.out
  - An instance of PrintStream
  - Standard output such as keyboard window as defined by OS
- System.err
  - An instance of PrintStream
  - Standard error output as defined by OS

Java Review – Exec, I/O, New Features George Blankenship 9

Simple Keyboard Input

```
import java.io.* ;

class BufferedReaderTest {
    public static void main(String [] args) throws IOException {
// Create a BufferedReader wrapping standard input
       InputStreamReader kb = new InputStreamReader(System.in));
       BufferedReader in = new BufferedReader(kb) ;

       String s ;
       s = in.readLine() ;      // Reads any string terminated by \n

       System.out.println("s = " + s) ;   // Print what was read
    }
}
```

Java Topics

- Running Java programs
- Stream I/O
- New features

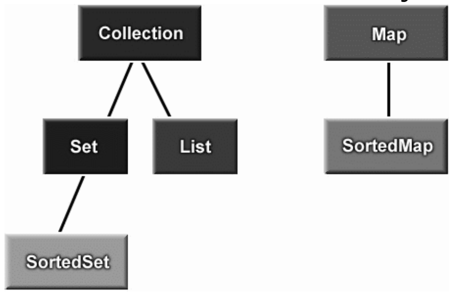
Java Collections Framework

- Collection
  - Object that groups multiple elements into one unit
  - Also called container
- Collection framework consists of
  - Interfaces - abstract data type
  - Implementations - reusable data structures
  - Algorithms - reusable functionality

Core Collection Interfaces

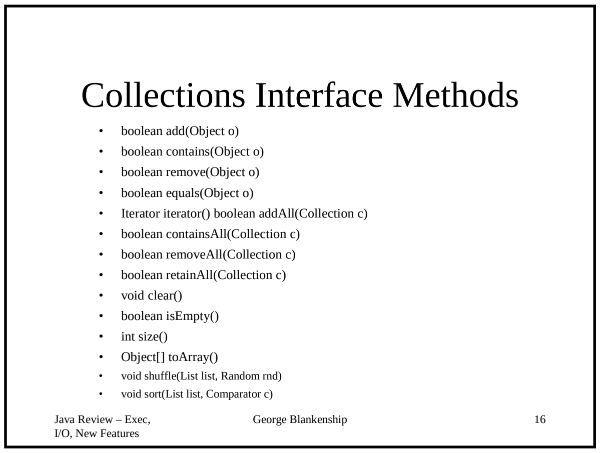
- *Collection* - group of elements
- *Set* - no duplicate elements
- *List* - ordered collection
- *Map* - maps keys to elements
- *SortedSet*, *SortedMap*
  - – Extends *Set* and *Map* (inheritance)
  - – Sorted ordering of elements

Core Collection Hierarchy



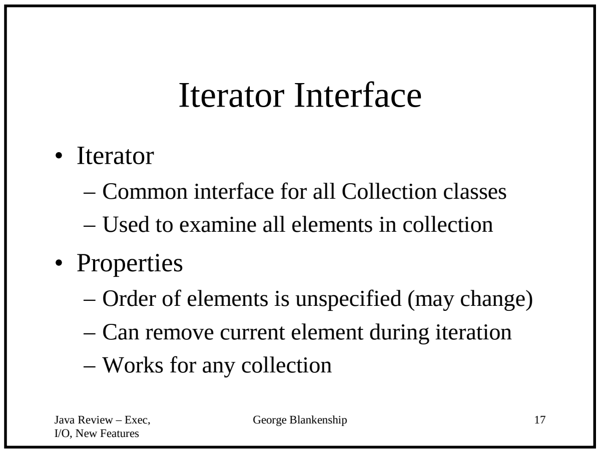
Collections Interface Implementations

- General implementations
  - – Primary public implementation
  - – Example
    - *List – ArrayList, LinkedList*
    - *Set – TreeSet, HashSet*
    - *Map – TreeMap, HashMap*
- Wrapper implementations
  - – Combined with other interfaces
  - – Example
    - Synchronized ArrayList, unmodifiable HashMap



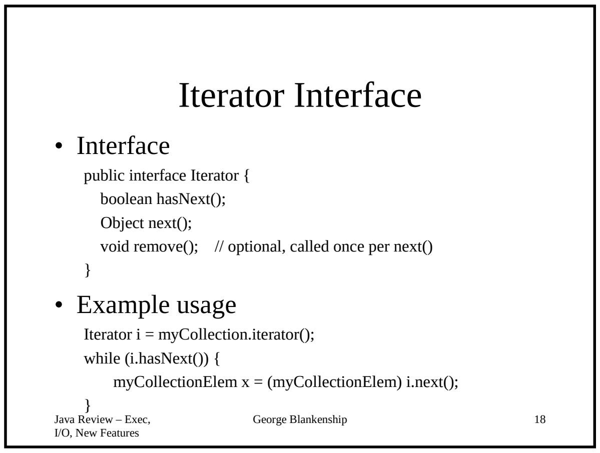
Collections Interface Methods

- boolean add(Object o)
- boolean contains(Object o)
- boolean remove(Object o)
- boolean equals(Object o)
- Iterator iterator() boolean addAll(Collection c)
- boolean containsAll(Collection c)
- boolean removeAll(Collection c)
- boolean retainAll(Collection c)
- void clear()
- boolean isEmpty()
- int size()
- Object[] toArray()
- void shuffle(List list, Random rnd)
- void sort(List list, Comparator c)



Iterator Interface

- Iterator
  - – Common interface for all Collection classes
  - – Used to examine all elements in collection
- Properties
  - – Order of elements is unspecified (may change)
  - – Can remove current element during iteration
  - – Works for any collection



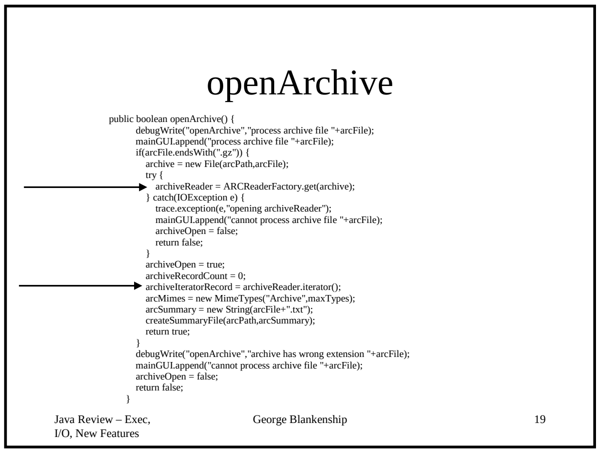
Iterator Interface

- Interface

```
public interface Iterator {
   boolean hasNext();
   Object next();
   void remove();    // optional, called once per next()
}
```

- Example usage

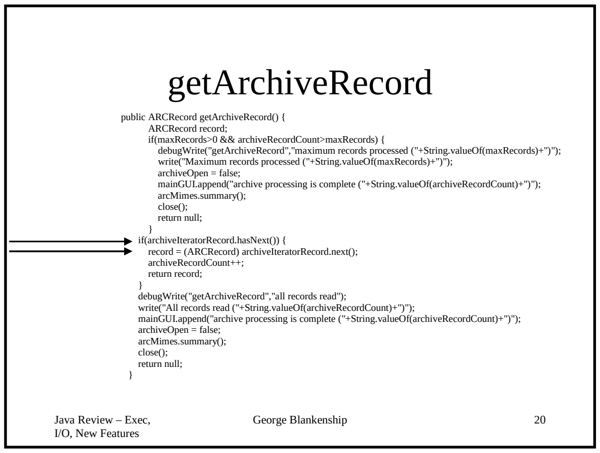
```
Iterator i = myCollection.iterator();
while (i.hasNext()) {
      myCollectionElem x = (myCollectionElem) i.next();
}
```



openArchive

```
public boolean openArchive() {
        debugWrite("openArchive","process archive file "+arcFile);
        mainGUI.append("process archive file "+arcFile);
        if(arcFile.endsWith(".gz")) {
           archive = new File(arcPath,arcFile);
           try {
              archiveReader = ARCReaderFactory.get(archive);
           } catch(IOException e) {
              trace.exception(e,"opening archiveReader");
              mainGUI.append("cannot process archive file "+arcFile);
              archiveOpen = false;
              return false;
           }
           archiveOpen = true;
           archiveRecordCount = 0;
           archiveIteratorRecord = archiveReader.iterator();
           arcMimes = new MimeTypes("Archive",maxTypes);
           arcSummary = new String(arcFile+".txt");
           createSummaryFile(arcPath,arcSummary);
           return true;
        }
        debugWrite("openArchive","archive has wrong extension "+arcFile);
        mainGUI.append("cannot process archive file "+arcFile);
        archiveOpen = false;
        return false;
    }
```

Java Review – Exec, I/O, New Features   George Blankenship   19



getArchiveRecord

```
public ARCRecord getArchiveRecord() {
          ARCRecord record;
          if(maxRecords>0 && archiveRecordCount>maxRecords) {
             debugWrite("getArchiveRecord","maximum records processed ("+String.valueOf(maxRecords)+")");
             write("Maximum records processed ("+String.valueOf(maxRecords)+")");
             archiveOpen = false;
             mainGUI.append("archive processing is complete ("+String.valueOf(archiveRecordCount)+")");
             arcMimes.summary();
             close();
             return null;
          }
       if(archiveIteratorRecord.hasNext()) {
          record = (ARCRecord) archiveIteratorRecord.next();
          archiveRecordCount++;
          return record;
       }
       debugWrite("getArchiveRecord","all records read");
       write("All records read ("+String.valueOf(archiveRecordCount)+")");
       mainGUI.append("archive processing is complete ("+String.valueOf(archiveRecordCount)+")");
       archiveOpen = false;
       arcMimes.summary();
       close();
       return null;
    }
```

Java Review – Exec, I/O, New Features   George Blankenship   20

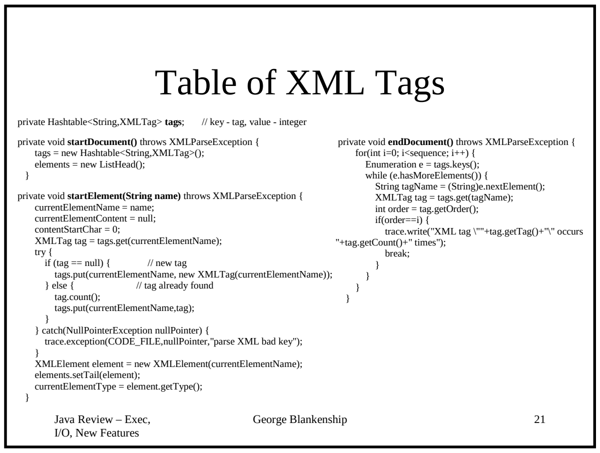


Table of XML Tags

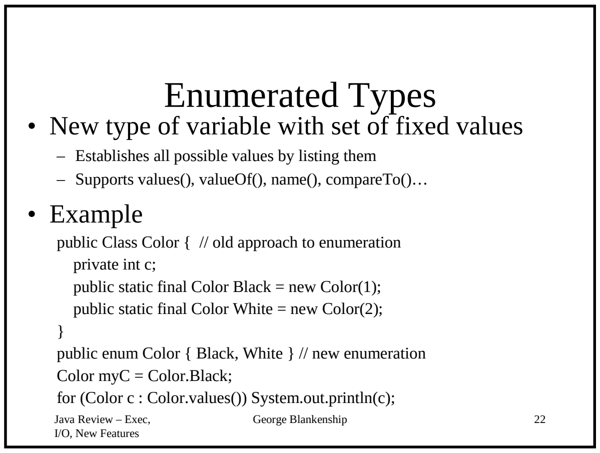
```
private Hashtable<String,XMLTag> tags;      // key - tag, value - integer

private void startDocument() throws XMLParseException {
      tags = new Hashtable<String,XMLTag>();
      elements = new ListHead();
  }

private void startElement(String name) throws XMLParseException {
      currentElementName = name;
      currentElementContent = null;
      contentStartChar = 0;
      XMLTag tag = tags.get(currentElementName);
      try {
         if (tag == null) {             // new tag
            tags.put(currentElementName, new XMLTag(currentElementName));
         } else {                       // tag already found
            tag.count();
            tags.put(currentElementName,tag);
         }
      } catch(NullPointerException nullPointer) {
         trace.exception(CODE_FILE,nullPointer,"parse XML bad key");
      }
      XMLElement element = new XMLElement(currentElementName);
      elements.setTail(element);
      currentElementType = element.getType();
  }
```

```
private void endDocument() throws XMLParseException {
      for(int i=0; i<sequence; i++) {
         Enumeration e = tags.keys();
         while (e.hasMoreElements()) {
            String tagName = (String)e.nextElement();
            XMLTag tag = tags.get(tagName);
            int order = tag.getOrder();
            if(order==i) {
               trace.write("XML tag \""+tag.getTag()+"\" occurs
"+tag.getCount()+" times");
               break;
            }
         }
      }
   }
```

Java Review – Exec, I/O, New Features   George Blankenship   21

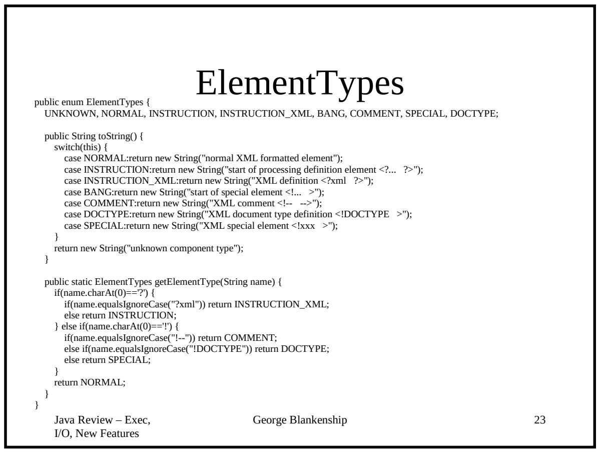


Enumerated Types

- New type of variable with set of fixed values
  - Establishes all possible values by listing them
  - Supports values(), valueOf(), name(), compareTo()…
- Example

```
public Class Color {  // old approach to enumeration
    private int c;
    public static final Color Black = new Color(1);
    public static final Color White = new Color(2);
}
public enum Color { Black, White } // new enumeration
Color myC = Color.Black;
for (Color c : Color.values()) System.out.println(c);
```

Java Review – Exec, I/O, New Features   George Blankenship   22



ElementTypes

```
public enum ElementTypes {
   UNKNOWN, NORMAL, INSTRUCTION, INSTRUCTION_XML, BANG, COMMENT, SPECIAL, DOCTYPE;

   public String toString() {
      switch(this) {
         case NORMAL:return new String("normal XML formatted element");
         case INSTRUCTION:return new String("start of processing definition element <?...   ?>");
         case INSTRUCTION_XML:return new String("XML definition <?xml   ?>");
         case BANG:return new String("start of special element <!...   >");
         case COMMENT:return new String("XML comment <!--   -->");
         case DOCTYPE:return new String("XML document type definition <!DOCTYPE   >");
         case SPECIAL:return new String("XML special element <!xxx   >");
      }
      return new String("unknown component type");
   }

   public static ElementTypes getElementType(String name) {
      if(name.charAt(0)=='?') {
         if(name.equalsIgnoreCase("?xml")) return INSTRUCTION_XML;
         else return INSTRUCTION;
      } else if(name.charAt(0)=='!') {
         if(name.equalsIgnoreCase("!--")) return COMMENT;
         else if(name.equalsIgnoreCase("!DOCTYPE")) return DOCTYPE;
         else return SPECIAL;
      }
      return NORMAL;
   }
}
```

Java Review – Exec, I/O, New Features   George Blankenship   23