Summary of remaining Java features

slides created by Marty Stepp
http://www.cs.washington.edu/331/
## Java keywords

<table>
<thead>
<tr>
<th>abstract</th>
<th>assert</th>
<th>boolean</th>
<th>break</th>
<th>byte</th>
</tr>
</thead>
<tbody>
<tr>
<td>case</td>
<td>catch</td>
<td>char</td>
<td>class</td>
<td>const</td>
</tr>
<tr>
<td>continue</td>
<td>default</td>
<td>do</td>
<td>double</td>
<td>else</td>
</tr>
<tr>
<td>enum</td>
<td>extends</td>
<td>final</td>
<td>finally</td>
<td>float</td>
</tr>
<tr>
<td>for</td>
<td>goto</td>
<td>if</td>
<td>implements</td>
<td>import</td>
</tr>
<tr>
<td>instanceof</td>
<td>int</td>
<td>interface</td>
<td>long</td>
<td>native</td>
</tr>
<tr>
<td>new</td>
<td>package</td>
<td>private</td>
<td>protected</td>
<td>public</td>
</tr>
<tr>
<td>return</td>
<td>short</td>
<td>static</td>
<td>strictfp</td>
<td>super</td>
</tr>
<tr>
<td>switch</td>
<td>synchronized</td>
<td>this</td>
<td>throw</td>
<td>throws</td>
</tr>
<tr>
<td>transient</td>
<td>try</td>
<td>void</td>
<td>volatile</td>
<td>while</td>
</tr>
</tbody>
</table>

- **Reserved words for literal values:**

<table>
<thead>
<tr>
<th>false</th>
<th>null</th>
<th>true</th>
</tr>
</thead>
</table>

---

*Note: The table and list provide a comprehensive overview of Java keywords, including both reserved words and those used for literal values.*
## Java keywords - types

<table>
<thead>
<tr>
<th>abstract</th>
<th>assert</th>
<th>boolean</th>
<th>break</th>
<th>byte</th>
</tr>
</thead>
<tbody>
<tr>
<td>case</td>
<td>catch</td>
<td>char</td>
<td>class</td>
<td>const</td>
</tr>
<tr>
<td>continue</td>
<td>default</td>
<td>do</td>
<td>double</td>
<td>else</td>
</tr>
<tr>
<td>enum</td>
<td>extends</td>
<td>final</td>
<td>finally</td>
<td>float</td>
</tr>
<tr>
<td>for</td>
<td>goto</td>
<td>if</td>
<td>implements</td>
<td>import</td>
</tr>
<tr>
<td>instanceof</td>
<td>int</td>
<td>interface</td>
<td>long</td>
<td>native</td>
</tr>
<tr>
<td>new</td>
<td>package</td>
<td>private</td>
<td>protected</td>
<td>public</td>
</tr>
<tr>
<td>return</td>
<td>short</td>
<td>static</td>
<td>strictfp</td>
<td>super</td>
</tr>
<tr>
<td>switch</td>
<td>synchronized</td>
<td>this</td>
<td>throw</td>
<td>throws</td>
</tr>
<tr>
<td>transient</td>
<td>try</td>
<td>void</td>
<td>volatile</td>
<td>while</td>
</tr>
</tbody>
</table>
The eight primitive types

- You're probably familiar with `int, double, char, boolean`.
  - `int` 32-bit signed integer from $-2^{31}$ to $2^{31}-1$
  - `double` 64-bit signed real number in range of +/- $1.7 \cdot 10^{308}$
  - `char` 16-bit unsigned integer from 0 to 65535 ($2^{16}-1$)

But Java has four other primitive types:

- `long` 64-bit signed integer from $-2^{63}$ to $2^{63}-1$
- `short` 16-bit signed integer from $-32768$ ($-2^{15}$) to 32767 ($2^{15}-1$)
- `byte` 8-bit unsigned integer from 0-255
- `float` 32-bit signed real number in range of +/- $3.4 \cdot 10^{38}$

- These types allow broader number ranges (`long`), saving memory (`float, short`), or compact representation of binary data (`byte`).
The switch statement

```java
switch (integer expression) {
    case value:
        statements;
        break;
    case value:
        statements;
        break;
    ...
    default: // if no other case is chosen
        statements;
        break;
}
```

- An alternative to the if/else statement when performing various actions based on the value of a given variable or expression.
The finally block

```
try {
    statement(s);
} catch (type name) {
    code to handle the exception
} finally {
    code to run after the try or catch finishes
}
```

- Often used for common "clean-up" code.

```
try {
    // ... write to output file; might throw
} catch (IOException ioe) {
    System.out.println("Caught IOException: "+ ioe.getMessage());
} finally {
    out.close();
}
```

- The `catch` block is optional; `try/finally` is also legal.
A few unusual keywords

• **const**  Reserved as a keyword for possible use with constants, but never used.

• **goto**  Reserved as a keyword for possible use with jumping in/out of loops, but never used.

• **strictfp**  Specifies how a certain piece of code should handle floating-point arithmetic for compatibility. (Rarely used.)

• **native**  Specifies a method or piece of code that is not implemented in Java but rather in a "native" system language such as C/C++. "JNI" (Rarely used except by high-performance APIs.)
Java's top-level packages

- java.applet
- java.beans
- java.io
- java.lang
- java.math
- java.net
- java.nio
- java.rmi
- java.security
- java.sql
- java.text
- java.util
- javax.accessibility
- javax.activation
- javax.activity
- javax.annotation
- javax.crypto
- javax.imageio
- javax.jws
- javax.lang.model
- javax.management
- javax.naming
- javax.net
- javax.print
- javax.rmi
- javax.script
- javax.security.auth
- javax.security.cert
- javax.security.sasl
- javax.sound.midi
- javax.sound.sampled
- javax.sql
- javax.swing
- javax.tools
- javax.transaction
- javax.xml
Package details

- **java.applet**
  - An *applet* is a Java GUI app embedded in a web page. (Horstmann Ch. 10)

- **java.beans**
  - A *bean* is a malleable object (e.g. GUI component) for use in a visual IDE.

- **java.math**
  - `BigInteger`, `BigDecimal` classes for large numeric computing.

- **java.net**
  - Network features such as sockets.

- **java.rmi**
  - Remote Method Invocation (RMI) lets you call a method on another computer transparently as though it were on your machine.

- **java.security**
  - Features for setting permissions and security of Java programs.
Package details 2

- **java.sql**
  - Features for connecting to databases.

- **java.text**
  - Features for advanced text formatting and processing.

- **javax.crypto**
  - Implementations of common encryption algorithms for securing data.

- **javax.script**
  - Interface between Java and JavaScript and other scripting languages.

- **javax.sound.midi, javax.sound.sampled**
  - Classes for playing various audio formats.

- **javax.xml**
  - Tools for reading/writing XML data in Java programs.