

Introductory Java 3

J3

Naming

Literals

Primitives

Java Packages

stargate.Sam “Sam” – but which one?

lordOfTheRings.Sam



stardewValley.Sam



dannyPhantom.Sam



Java Packages

stargate.Sam



Namespaces in which
“Sam” is well defined.

lordOfTheRings.Sam



stardewValley.Sam



dannyPhantom.Sam



Java Modules

- Group packages and related resources
- Strong encapsulation
- Explicit dependencies

```
module java.sql {  
    requires transitive java.logging;  
    requires transitive java.transaction.xa;  
    requires transitive java.xml;  
  
    exports java.sql;  
    exports javax.sql;  
  
    uses java.sql.Driver;  
}
```



Java Variables

Instance Variable – “field”

Each object (instance) has its own version (instance) of the field

```
public class MyClass {  
    int myField;  
    static String myStaticField;  
    public static void main(String[] args) {  
        String x = "hello!";  
    }  
}
```

Class Variable – “static field”

Exactly one version of the field exists, shared globally

Parameter

Temporary state, limited to execution scope of code, value passed from one method to another

Local Variable

Temporary state, limited to execution scope of code

Java Naming

Legal identifiers

- Start with: Unicode letter, `_`, or `$`
- Plus arbitrarily many more Unicode letters, `_`, `$`, and digits
- Except keywords and reserved words
- Capitalization matters!

Conventions

- Avoid `_` and `$`, especially at the start of an identifier
- CamelCase – combine words by starting each word with a capital letter
- Class names start with capital letters (e.g. `BikeShed`)
- Variable names start with a lower-case letter (e.g. `currentGear`)
- Constant names use all-caps and underscores (e.g. `MAX_GEAR_RATIO`)

Java's Primitive Data Types

Type	Description	Range	Default
<code>byte</code>	8-bit signed 2's complement integer	-128 to 127	0
<code>short</code>	16-bit signed 2's complement integer	-32768 to 32767	0
<code>int</code>	32-bit signed 2's complement integer	-2^{31} to $2^{31} - 1$	0
<code>long</code>	64-bit signed 2's complement integer	-2^{63} to $2^{63} - 1$	0L
<code>float</code>	single precision 64-bit IEEE 754 floating point number		0.0f
<code>double</code>	double precision 64-bit IEEE 754 floating point number		0.0d
<code>boolean</code>	Logically just a single bit: true or false	<code>true</code> , <code>false</code>	<code>false</code>
<code>char</code>	16-bit Unicode character	0 to 65535	0

NEVER use float/double where precision is needed, especially for money!



Java Literals

Numbers

- 48 (byte, short, int) or 48L (long)
- 0x30 or 0x30L (48 expressed in hex)
- 0b110000 or 0b110000L (48 expressed in binary)
- 48.0/48.0d (double) or 48f (float)
- Underscores can separate digits: `long creditCard = 1234_5678_9012_3456L;`

Others

- Booleans: `true`, `false`
- Strings: "Hello World!"
- Chars: 'a', '!
- Null: `null`  **"Billion Dollar Mistake"**