Computer Programming I Instructor: Greg Shaw

COP 2210

# Java's "Primitive" Types

**I. "Everything is an Object" (well, almost everything)**

The exception to the "everything in Java is an object" rule is the so-called "primitive" types.

For performance reasons, Java implemented the primitive types as "automatic" storage-class variables, i.e., they are created at compile-time as in C++. Objects, on the other hand, are “dynamic” storage-class variables, i.e., they are created at run-time (via the **new** operator). Creating dynamic objects entails more overhead.

However, Java does provide a "Wrapper" class associated with each of the primitive types. This allows us to create objects that store primitive-type values (i.e., are *references* to primitive types). The Wrapper classes will be covered in the future.

1. **Java's Primitive Data Types**

1. Unlike some other languages (like C++), the size (i.e., number of bits) of each of the primitive types is guaranteed to be consistent in all implementations of Java. This is one of the things that make Java programs portable.
2. Although Java provides 4 different integer types, there is generally no reason to use types **byte** and **short**.Use type **int** except in the rare case where you need to store integer values smaller than -2,147,483,648 or larger than +2,147,483,647 (the limits for type **int**). In that case, use type **long**
3. Although Java has 2 different floating-point types, there is no reason to use type **float,** which has considerably less precision than type **double.**

* Summary: *Which numeric type should I use?*
* If a number MAY have a fractional part (i.e. a decimal point), then use **double**
* Otherwise, if the number will *always* be in the range

-2,147,483,648 .. +2,147,483,647, use **int**

* Otherwise, use **long**

**Java’s Primitive Types**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Data Type** | **Size** | **Possible Values** | **Default Value\*** | **Wrapper**  **Type** |
| **byte** | 8  bits | integers  -128 .. +127 | 0 | **Byte** |
| **short** | 16  bits | integers  -32,768 .. +32,767 | 0 | **Short** |
| **int** | 32  bits | integers  -2,147,483,648 .. +2,147,483,647 | 0 | **Integer** |
| **long** | 64  bits | integers  -263 .. 263-1 | 0L | **Long** |
| **float** | 32  bits | floating-point numbers  -3.4E+38 .. +3.4E+38  (approx.) | 0.0f | **Float** |
| **double** | 64  bits | floating-point numbers  -1.7E+308 .. 1.7E+308 (approx.) | 0.0d | **Double** |
| **char** | 16  bits | Single characters  Unicode 0 .. Unicode 216-1 | '\u0000'  (null) | **Character** |
| **boolean** | --- | **true** and **false** | **false** | **Boolean** |

**\***Concerning default values (i.e., assigned by Java)

1. Default values are assigned to primitive type variables *only* when they are used as *instance variables* of a class.
2. If a primitive type variable is used as a "local" variable (i.e., declared in a method), then that variable is *not* initialized, and contains "garbage".
3. Fortunately, any attempt to use an uninitialized local variable in an expression is a syntax error.