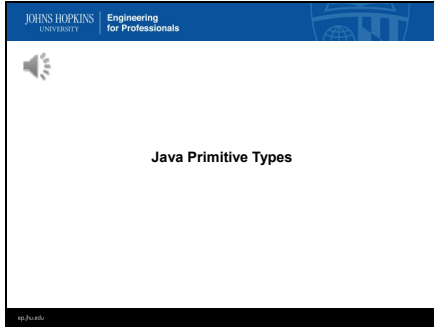


1



In this lecture you will learn about Java's primitive data types.

2

The slide is titled "Java Primitive Types". It features a blue header with the "JOHNS HOPKINS University" logo and the text "Engineering for Professionals". A speaker icon is located in the top left corner of the slide content area. Below the title is a table with two columns: "Type" and "Keyword".

Type	Keyword
Byte	byte
Short Integer	short
Integer	int
Long Integer	long
Floating Point	float
Double Precision	double
Character	char
Boolean	boolean

Java defines eight (8) fundamental data types. These fundamental types are referred to as primitive types, and you can define variables that store data that fall into one of three categories:

Integer or floating point values. There are 6 primitive numeric types.


Values of a single Unicode character

The logical values true or false

Java has 8 keywords that are used to declare variables for each of the primitive types.

Java also defines two non-primitive types...arrays and objects...that we will discuss in future lectures.

3



### Integer Types

Type	Storage Requirement	Range of Values
byte	1 byte	-128 to 127
short	2 bytes	-32,768 to 32,767
int	4 bytes	-2,147,483,648 to 2,147,483,647
long	8 bytes	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807

Integers types are used to represent signed whole numbers...numbers without fractional parts.

Java implements four integer types. The only difference between these types is the range of values that they can represent.


The smallest integer type, byte, can only be used to store values in the range -128 to 127.

The largest integer type, long, can be used to store values in a much larger range.

As you can see, the larger the number range, the more memory is required to store a particular type. To ensure portability, the storage requirements for each type are the same for each type of computer Java is implemented on...so bytes always require 1 byte, shorts always require 2 bytes, and so forth.

For purposes of this course, you don't have to worry about which type of integer is the best one to use for your programs. You'll be fine if you use the regular integer type...int.

4



### Declaring Integer Types

```


public class IntegerDeclarationDemol
{
    public static void main( String [] args )
    {
        byte b;      // Declare a byte type
        short s;     // Declare a short
        int x;       // Declare an int
        long aLong;  // Declare a long
        ...
    }
}

```

This program simply illustrates how to declare variables of each primitive type.

It declares a byte variable, a short integer variable, a regular integer variable, and a long integer variable.

5



### Integer Literals

```

public class IntegerLiteralDemo
{
    public static void main( String [] args )
    {
        int w;
        int x;
        int y;
        long z;

        w = 25;           // Assign w the decimal literal 25
        x = 0x010;        // Assign x a hexadecimal literal
        y = 0xCAB;        // Assign y a hexadecimal literal
        z = 40000L;       // Assign z a long decimal literal
    }
}

```

$0xCAB: 12 \cdot 16^2 + 10 \cdot 16^1 + 11 \cdot 16^0 = \text{decimal } 3243$   
 $0X010: 0 \cdot 16^2 + 1 \cdot 16^1 + 0 \cdot 16^0 = \text{decimal } 16$

Explicit data values that are used in programs are called LITERALS.

Integer types are usually assigned a decimal value, for example, 25, but they can also be assigned a hexadecimal value or even an octal value.

A hexadecimal number is a number expressed to a base of 16. In Java, hexadecimal literals begin with a zero and an 'x'. The 'x' can be upper or lower case. For example, the hexadecimal number 0X010 is equivalent to the decimal number 16. The letters A through F are used to represent digits with values in the range 10 to 15, so the hexadecimal number 0xCAB is equivalent to the decimal number 3243.


Octal numbers are 3 digit numbers expressed to a base of 8, and in Java they have a leading zero. Octal numbers were used in the old days of computing, but are not used much now. Just be careful not to put a leading zero at the start of an integer literal...or Java will treat it as an octal number.

Note that long literals are written with an "L" suffix. The L can be upper or lower case, but you should always use an upper case L because it's easy to confuse a lower case L with the number one.

One more important thing to note about literals is that you don't use commas in the thousands places. For example, in this program z is assigned the literal value forty thousand, but no commas were used. If you use a comma it is an error and your program will not compile.

Integer literals can be assigned to integer variables as part of a declaration statement or by using assignment statements as I've done here.

6



### Floating Point Types

Type	Storage Requirement	Range of Values
float	4 bytes	$\pm 3.40282347 \times 10^{38}$
double	8 bytes	$\pm 1.79769313486231570 \times 10^{308}$

```

float aFloat = 147.65F;    // Float literals have 'F' suffix
float aFloat2 = 1.4765E+2F; // Can be written in exponent format
double w = 1.2324565E-3;  // .001232456
double z = 1233333.2D;    // Can use a 'D' suffix

```

Floating point types are used to represent signed numbers that have fractional parts.

Java implements two floating point types...regular floating point and double precision.

Floats may have an 'F' suffix, and an upper or lower case F can be used.

Similarly, doubles may have an upper or lower case D suffix.


If a suffix is not specified for a floating point literal, Java assumes it is of type double.

Exponent format may also be used to specify literals. Exponent format uses a numeric value, followed by an upper or lower case E, followed by a power of 10. So, 1.5E+2 is equivalent to 1.5 times 10 to the power of 2, or 150.

Float and double literals may be assigned to variables as part of a declaration statement...as I have done here... or by using separate assignment statements like you have seen earlier in this lecture.

If you assign literals that are too big or too small for the declared type the Java compiler will issue an error message.

7



### Character Types

```
char aCharacter = 'B'; // A char variable
char s = 'hello';      // Error---single character only
char c = "x";          // Error---no regular quotes
```

Escape Sequence	Meaning	Unicode Value
\b	backspace	\u0008
\t	tab	\u0009
\n	newline or linefeed	\u000a
\r	carriage return	\u000d


The Java type 'char' is used to represent single character values.

Literals of type char are enclosed in single quotes when assign to variables. Char literals may be assigned to char variables as part of a declaration statement or by using a separate assignment statement.

char literals must be a single character only, And regular quotes cannot be used when assignments are made.

Char variables are stored in 2 bytes of memory as a Unicode character. A Unicode character is a special character set designed for handling international languages besides English. There are more than 65,000 characters in the Unicode character set. Unicode characters are specified using an escape sequence that begins with backslash-u. Regular escape sequences can also be used, as indicated in this table. Our use of Unicode characters in this course will be limited to those in the table. If you would like to find out more information about Unicode characters you can check out [www.unicode.org](http://www.unicode.org).

8



### Output Demonstration

```
public class OutputBasics
{
    public static void main( String [] args )
    {
        int x = 10;
        float y = 13.333F;

        System.out.println();
        System.out.println( x + "\t" + y );
    }
}
```

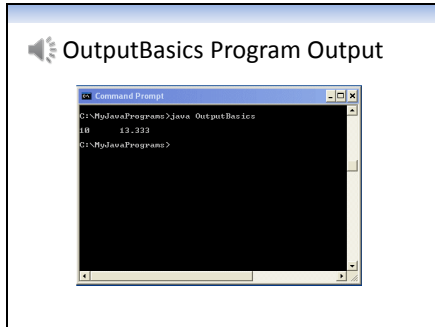
↑  
tab will occur between printing of x and y

This program demonstrates some basic output mechanics.

The program will display the value of variable x and variable y.

A tab character escape sequence was used in the System.out.println() argument list to insert space between the displayed values of these variables.

9



This is the output of the OutputBasics program.

Notice how the two numbers are aligned...this is due to using the tab character.

10

### Boolean Types

Used as part of conditional tests in program decision logic. Can only take on values true or false

```
boolean result = false; // A boolean variable
```

Boolean types are the eighth primitive type in Java.

Boolean types are often used in conditional tests to control the flow of execution in programs. Boolean variables can only take on the values true or false.

You will learn more about boolean variables in a later lecture...after decision-making statements are introduced.

11

### Declaring & Using Constants

A **CONSTANT** is a variable whose value cannot be changed once it is initialized..

```
public class ConstantDemo1
{
    public static void main( String [] args )
    {
        final int x = 10; // x is a constant
        int y = 3;
        final int z;      // z is a constant
        z = y;            // z is initialized
    }
}
```

Constants are variables whose values cannot be changed once initialized.


Constants are often useful when you write programs in which the value of some variables shouldn't be changed during the course of program execution. In those types of programs it's a good idea to define any variables whose values shouldn't be changed as constants, to protect against inadvertent errors.

Constants are declared by using the keyword 'final' at the beginning of the variable's declaration statement.

In this example, the variables x and z are declared as constants.

Constants may be initialized only once...either in their declaration or using an assignment statement. In this program, x is initialized in its declaration statement, and z is initialized in an assignment statement. Neither of these variables can be assigned another value in this program. If a programmer attempts to change their value the program will not compile.

12



### Declaring & Using Constants


```
public class ConstantDemo2
{
    public static void main( String [] args )
    {
        final int x = 10; // x is a constant
        int y = 3;
        final int z;      // z is a constant

        z = x;           // z is initialized
        ..
        x = 99;           // ERROR
    }
}
```

Since x and z are constants, neither of these variables can be assigned another value in this program. If a programmer attempts to change their value the program will not compile.

This program will not compile because there is an attempt to change the value of x.

13



### Declaring & Using Constants

```
public class ConstantDemo3
{
    public static void main( String [] args )
    {
        final double PI = 3.14;
        final double INCHES_PER_FOOT = 12;
        final double INCHES_PER_YARD = 36;
        double diameter;
        ..
        double circumference = PI * diameter;
        ..
    }
}
```

It is a common, and recommended programming practice, to name constants in all upper-case letters. Java doesn't care if this is done, but it helps them stand out visually and lets readers of a program know that they are constants.

This program defines a number of constants that will be used in calculations. The universal constant PI will be defined as a constant, as well as some constants for converting feet to inches.

Later in the course we will learn how to define a set of constants that can be shared across many different programs.

