**Lesson 1**

**Creating a basic Java program**

Every java program must have a class declaration and a main method that looks like the following:

public class *ClassName*

{

public static void main(String[] args)

{

}

}

where ClassName is named by the writer of the program.

The name of the file you save the code in must be the same as ClassName.

For example, in the following example, the code above must be put in a file named Qwerty.java

public class Qwerty

{

public static void main(String[] args)

{

}

}

Java is case sensitive. So you cannot put “Public” instead of “public”.

Also, the name of a class cannot start with a number and it cannot contain spaces.

Every single time you run a program, you must first compile it and then run it.

**Printing text to standard output**

The following line of code prints “Hello, World!”

System.out.print(“Hello, World!”);

In general, System.out.print(“*what you want to print*”);

prints what you want to print.

In java, every statement requires a semicolon at the end. So if you put

System.out.print(“Hello, World!”)

Your code will have an error when you compile.

Don’t put semicolons after the class and main method declaration.

Only put semicolons after statements.

For example, don’t do any of the following:

* public class Qwerty;
* public static void main(String[] args);
* {;

Currently, all the print statements you have learned can only print in one line.

The statement

System.out.println(“*what you want to print*”);

prints what you want to print followed by a new line.

This means that the next time you print, you will print on a new line.

Example:

public class PrintingDemo1

{

public static void main(String[] args)

{

System.out.print(“aaa”);

System.out.println(“bbb”);

System.out.print(“ccc”);

System.out.print(“ddd”);

}

}

Output:

aaabbb

cccddd

You can put any character inside quotation marks, including numbers and spaces.

**Comments**

Comments are statements that are ignored by the compiler.

So comments don’t affect how a program runs.

There are two ways to make comments:

//Comment here

and

/\* Comments start here

Comments end here\*/

**Escape characters**

The following are escape characters:

\n

\”

\\

Printing \n is equivalent to printing a new line.

For example,

System.out.print(“aaa\nbbb”);

prints

aaa

bbb

Printing \” is equivalent to printing a quotation mark.

For example,

System.out.print(“aaa\n\”bbb\””);

prints

aaa

“bbb”

Printing \\ is equivalent to printing a backslash.

For example,

System.out.print(“\\aaa\n\”bbb\””);

prints

\aaa

“bbb”

**Doing basic arithmetic**

You can also do math with java.

The statement

System.out.print(6 + 4);

prints 10.

Java follows the order of operations as well.

So the statement

System.out.print(3 \* 5 + 2 \* 2);

prints 19.

The statement

System.out.print(3 \* (5 + 2) \* 2);

prints 42.

**The % operator**

Now we will introduce the % operator.

a % b is the remainder when a is divided by b.

For example, 7 % 3 = 1, since the remainder when 7 is divided by 3 is 1.

The % operator has the same priority as multiplication and division in the order of operations.

**Integer Division**

The / operator is division.

For example, 8 / 2 = 4.

However, when dividing two integers, the / operator only takes the integer part.

For example, 7 / 3 = 2, and 15 / 4 = 3.

**Using numbers other than integers**

You can also use numbers other than integers to do arithmetic.

For example, the statement

System.out.print(7.3 \* 4 + 6.1 \* 2);

prints 41.4

To do actual division, you must make sure that at least one of the numbers is a decimal.

For example,

System.out.print(7 / 2);

outputs 3, and

System.out.print(7 / 2.0);

outputs 3.5

**String concatenation**

To merge two strings together, use the + operator.

For example,

System.out.print(“aaa” + “bbb”);

prints

aaabbb

**String concatenation with numbers**

To merge strings with numbers, use the + operator.

For example,

System.out.print(“1 + 1 = ” + 2);

will output

1 + 1 = 2

Note that “1 + 1 = ” is a string, and it is not computing anything.

However, be careful. Since the order of operations goes from left to right,

System.out.print(“1 + 1 = ” + 1 + 1);

will output

1 + 1 = 11

To print the right result, use parenthesis.

So the statement

System.out.print(“1 + 1 = ” + (1 + 1));

will print

1 + 1 = 2

**Primitive data types**

The following are primitive data types:

int – an integer from -2^31 to 2^31 – 1

double – a real number

boolean – true or false

char – a single character

long – an integer from -2^63 to 2^63 – 1

short – an integer from -32768 to 32767

byte – an integer from -128 to 127

float – a real number with less precision than double and takes up less space

Generally, we will use int, double, boolean, char, and long.

**Declaring and initializing variables**

To declare a variable, use the following statement:

*variableType* *variableName*;

For example, the statement

int i;

declares an int named i.

To set the value of a variable, use the following statement:

*variableName* = *value*;

For example, the statement

i = 5;

sets the value of i to 5.

You can also use the following statement to declare and initialize a variable:

*variableType variableName* = *value*;

For example, the statement

double d = 3.5;

declares a double named d and sets its value to 3.5.

**Variable basics**

The name of a variable can never start with a number and can never contain spaces.

You can’t name two variables the same name.

The statement

System.out.print(*variableName*);

prints the value of the variable variableName.

For example,

int i = 5;

System.out.print(i);

prints 5

You can also do arithmetic with variables.

For example,

int i = 5;

double d = 2;

System.out.print(i / d);

prints 2.5

You can also combine variables with strings.

For example,

int i = 5;

int j = 7;

System.out.print(i + “ + ” + j + “ = ” + (i + j));

outputs

5 + 7 = 12

You can also do arithmetic when assigning variables.

For example,

int i = 5;

int j = 7;

int k = i + j;

System.out.print(i + “ + ” + j + “ = ” + k);

outputs

5 + 7 = 12

**Incrementing and Decrementing Variables**

To increase a variable x by 5, you can use

x = x + 5;

You can also write

x += 5;

Both statements increase x by 5.

Similarly, you can use -=, \*=, /=, and %=.

Also, to increase x by 1, use

x++;

Similarly, to decrease x by 1, use

x--;

**Strings**

A string is a sequence of characters.

Examples are “qwerty” and “a\nb\”\\ c”

You can declare a String using

String *variableName*;

You can set the value of Strings using

*variableName* = *value*;

You can also declare and initialize a string using

String *variableName* = *value;*

For example,

String s1 = “1 + 1”;

String s2 = “ = ” + (1 + 1);

String s3 = s1 + s2;

System.out.println(s3);

prints

1 + 1 = 2

**Characters**

A char is a single character.

Examples are: ‘a’, ‘2’, and ‘\n’

‘ab’ is not a char since it contains more than one character.

Notice that strings are enclosed with double quotes, and that chars are enclosed with single quotes.