Python → Java

1

Interpreted vs Compiled

- Python – Interpreted
  - Easy to run and test
  - Quicker prototyping
  - Program runs slower
- Java – Compiled
  - Execution time faster
  - Virtual Machine – compiled code portable

Java Compile

> javac TestClass.java

Creates a new executable file: TestClass.class

To run it:

> java TestClass

If TestClass uses other classes that you create, make sure they are in the same directory and they will be compiled at the same time with .class files of their own.

Classes, Objects, and Methods

Java	**Python**
class	class
object	object
method	method

In Python, a method is a function within an object, which is defined by a class.

In Java, all functions are in objects or classes so they are all called methods.

Hello World

```
def main():
   print “Hello World”
```

```
public class HelloPrinter
{
   public static void main(String[] args)
   {
        System.out.println(“Hello World");
   }
}
```

```
public class HelloPrinter
{
   public static void main(String[] args)
   {
        System.out.println(“Hello World");
   }
}
```

```
public class HelloPrinter
{
   public static void main(String[] args)
   {
      System.out.println("Hello World");
   }
}
```

A class definition for the class named HelloPrinter.

```
public class HelloPrinter
{
   public static void main(String[] args)
   {
      System.out.println("Hello World");
   }
}
```

Curly brackets mark the start and end of the class definition.

```
public class HelloPrinter
{
   public static void main(String[] args)
   {
      System.out.println("Hello World");
   }
}
```

A class definition

```
public class HelloPrinter
{
   public static void main(String[] args)
   {
      System.out.println("Hello World");
   }
}
```

The name of the file for a public class must match the class name: HelloPrinter.java

```
public class HelloPrinter
{
   public static void main(String[] args)
   {
      System.out.println("Hello World");
   }
}
```

public means it can be accessed from outside of this file.

private means only accessed within this file

```
public class HelloPrinter
{
   public static void main(String[] args)
   {
      System.out.println("Hello World");
   }
}
```

A method definition for the method named main.

```
public class HelloPrinter
{
   public static void main(String[] args)
   {
      System.out.println("Hello World");
   }
}
```

Curly brackets mark the start and end of the method definition.

```
public class HelloPrinter
{
   public static void main(String[] args)
   {
      System.out.println("Hello World");
   }
}
```

As in Python the method main has special meaning.

The execution of the file starts here.

```
public class HelloPrinter
{
   public static void main(String[] args)
   {
      System.out.println("Hello World");
   }
}
```

This portion allows you to create a main method that will take parameters (command line arguments).

```
public class HelloPrinter
{
   public static void main(String[] args)
   {
      System.out.println("Hello World");
   }
}
```

public means it can be accessed from outside of this class.

private means only accessed within this class.

```
public class HelloPrinter
{
   public static void main(String[] args)
   {
      System.out.println("Hello World");
   }
}
```

static means that this method (main) does not operate on an object. Without the word static it does operate on an object.

```
public class HelloPrinter
{
   public static void main(String[] args)
   {
      System.out.println("Hello World");
   }
}
```

void indicates the type of object it returns. In this case it does not return anything so to indicate that we use the word void.

```
public class HelloPrinter
{
   public static void main(String[] args)
   {
      System.out.println("Hello World");
   }
}
```

This line prints the string “Hello World” to the screen.

```
public class HelloPrinter
{
   public static void main(String[] args)
   {
      System.out.println("Hello World");
   }
}
```

println is a method that takes in a string as a parameter and prints it and then goes to a new line. print is another method that prints the parameter string, but does not add the newline.

```
public class HelloPrinter
{
   public static void main(String[] args)
   {
      System.out.println("Hello World");
   }
}
```

println operates on the object out (an object of the class PrintStream), which refers to the system output or the console window.

```
public class HelloPrinter
{
   public static void main(String[] args)
   {
      System.out.println("Hello World");
   }
}
```

out is declared in the class System, which is a class that is part of the Java library.

```
public class HelloPrinter
{
   public static void main(String[] args)
   {
      System.out.println("Hello World");
   }
}
```

The semicolon ; marks the end of the line of execution.

Java Has a Free-Form Layout

```
public class HelloPrinter{ public static void main(String[]
   args) { System.out.println("Hello World");}}
```

Means the same as:

```
public class HelloPrinter
{
   public static void main(String[] args)
   {
      System.out.println("Hello World");
   }
}
```

Java Has a Free-Form Layout

Punctuation is very important!

Python relied on indentation to differentiate one command from another.

Java uses punctuation such as curly brackets { } and semicolons **;**

Nevertheless, indentation is very important for the readability of Java programs.

Your Python training will serve you well.

Java Capitalization Conventions

- The first letter of a class name should be capitalized, but not all letters in the name.
- The first letter of all variables and methods should be lower case.
- All the letters of a constant should be capitalized.

Typing

- Java has strong typing
- All values have a type.
- Variables must be defined as a type and only used to store values of that type.

Strong Typing

Once a variable is defined to be of a certain type, you can only assign it values of that type.

```
int num;
num = 7;        // OK at this point
num = 8.5;      // will result in an error
```

Typing

- There are class types and primitive types.
- Example class type: String
- Example primitive types: int, float, char

Class Type

String greeting = “Hello”;

In this command, the object greeting is defined to be of type String and is assigned the value “Hello”.

This can be done in two steps if desired:

```
String greeting;
greeting = “Hello”;
```

Class Type Method

Since String is a class, there are methods defined to operate on objects of that class. For example the method length returns the number of characters in the string.

greeting.length() will return 5

Primitive Type

- Java defined types that are not classes.
- The primitive building block
- Integers: byte, short, int, long – 1,2,4,8 bytes
- Real Numbers: float, double – 4, 8 bytes
- Characters: char – 2 bytes
- Boolean: boolean (true or false) – 1 bit

Primitive Type

int num = 256;

In this command, variable num is declared to be of type int and assigned the value 256.

Primitive Type Method

Since int is not a class, there are no methods defined to operate on an int.

For example there is no method length that operates on an int:

num.length() will return an error

Static Methods

- Could be called Class Methods
- Static methods do not operate on objects.
- The reason for static methods is to operate on the parameters of the method as opposed to objects.

Static Methods

There are static methods defined in Java that can be used to manipulate primitive types.

For example sqrt is a static method defined in the class Math.

Math.sqrt(num) will return 16.

Static Methods

```
public static int max3 (int k, int m, int n) {
    if (k > m)
        if (k > n)
            return k;
        else
            return n;
    else
        if (m > n)
            return m;
        else
            return n;
}
```

```
public class Util {
    public static int max3 (int k, int m, int n) {
        if (k > m)
            if (k > n)
                return k;
            else
                return n;
        else
            if (m > n)
                return m;
            else
                return n;
    }
}
```

Static Methods

In the program, you would call the static method as follows:

```
int a = 5;
int b = 8;
int c = 4;
int max_int;

max_int = Util.max3(a, b, c);
or possibly:
max_int = Util.max3(9, b, c);
```

Mathematical Operators

Python	Java
+	+
-	-
*	*
/	/
**	Math.pow(x, y)
%	%
abs(x)	Math.abs(x)
sin(x)	Math.sin(x)
x+=1	x++

Increment / Decrement

```
int i = 5;
int j;

j = i++;
```
j has the value 5 and i has the value 6

```
i = 5;
j = ++i;
```
Both j and i have the value 6

Works the same with --

Division

- The same in Python and Java

10 / 4	returns 2
10.0 / 4.0	returns 2.5
10.0 / 4	returns 2.5

ints are automatically converted to floats

Casting

x = 3.3

Python
int(x) returns 3

Java
(int) x returns 3

Strings

- Both Python and Java have +

"abc" + "def" returns "abcdef"

- Substring

abcs = "abcdefghijk"

Python
abcs[0:3] returns "abc"

Java
abcs.substring(0,3) returns "abc"

String → number

- Python

eval("19") returns 19
eval("19.8") returns 19.8

- Java

Integer.parseInt("19") returns 19
Double.parseDouble("19.8") returns 19.8

Reading Input

- Python

```
grade = input("Enter grade: ")
```

- Java

```
import java.util.Scanner;   // before class def

Scanner in = new Scanner(System.in);
System.out.print("Enter grade: ");
int grade = in.nextInt();
```

Booleans

- Comparisons the same: <, <=, ==, >=, >, !=
- For Java Strings string1.equals(string2)
  ! string1.equals(string2)

- Python: True, False
  - Operators: and, or

- Java: true, false
  - Operators: &&, ||

Conditionals

- Python

```
if x < 0:
   x = x * x
```

- Java

```
if (x < 0) {
   x = x * x;
}
```

Conditionals

```
if (x < 0) {
   x = x * x;
}
```

same as:

```
if (x < 0)
   x = x * x;
```

Conditionals

```
if (x < 0) {
   x = x * x;
   x = x + 5;
}
```

different from:

```
if (x < 0)
   x = x * x;
   x = x + 5;
```

Conditionals

```
if (x < 0) {
   x = x * x;
   x = x + 5;
}
```

different from:

```
if (x < 0)
   x = x * x;
x = x + 5;
```

Conditionals

- Python

```
if x < 0:
   x = x * x
   print x
```

- Java

```
if (x < 0) {
   x = x * x;
   System.out.println(x);
}
```

Conditionals

- Python

```
if x < 0:
   x = x * x
else:
   x = 2 * x
```

- Java

```
if (x < 0)
   x = x * x;
else
   x = 2 * x;
```

Conditionals

- Python

```
if x < 0:
   x = x * x
elif x > 0:
   x = 2 * x
else:
   x = -1
```

- Java

```
if (x < 0)
   x = x * x;
else if (x > 0)
   x = 2 * x;
else
   x = -1;
```

Conditionals

```
if (x < 0) {
   x = x * x;
   System.out.println(x);
}
else if (x > 0)
   x = 2 * x;
else {
   x = -1;
   System.out.println(x);
}
```

Conditionals

```
if (x < 0)
   x = x * x;
else if (x > 0)
   x = 2 * x;
else
   x = -1;
```

Same as:

```
if (x < 0) x = x * x; else if (x > 0) x = 2 * x;
   else x = -1;
```

Conditionals

Same as:

```
if (x < 0)
   x = x * x;
else
   if (x > 0)
        x = 2 * x;
   else
        x = -1;
```

Conditionals

```
if (x < 0)
   x = x * x;
else if (x > 0)
   x = 2 * x;
else
   x = -1;
```

Very readable. The first true condition executes; the rest are not executed.

Nested Conditionals

```
if (x < 0)
   if (y < 0)
        x = x * y;
   else
        x = x * x;
else if (x > 0)
   if (y < 0)
        x = 2 * x;
   else
        x = y;
else
   x = -1;
```

For Loops

- Python

```
x = 5.2
for i in range(10):
   x = 3.9 * x * (1-x)
   print x
```

- Java

```
double x = 5.2;
int i;
for (i = 0; i < 10; i++) {
   x = 3.9 * x * (1-x);
   System.out.println(x);
}
```

For Loops

- Python

```
for i in range(2,11,2):
    print i
```

- Java

```
for (int i = 2; i < 11; i=i+2)
    System.out.println(i);
```

For Loops

```
for (i = 2; i < 11; i=i+2)
    System.out.println(i);

for (i = 2; i <= 10; i=i+2)
    System.out.println(i);
```

While Loops

- Python

```
i = 1
while i <= 10:
    print i
    i = i + 1
```

- Java

```
int i = 1;
while (i <= 10) {
    System.out.println(i);
    i = i + 1;
}
```