Java

Spring, 2019

History

- Java 1
  - James Gosling, Sun Microsystems.
  - Java 1.0.2 Oak 1996: 250 classes, applets.
  - Java 1.1: 500 classes, faster, better GUI code.
- Java 2 – J2ME, J2SE, J2EE
  - 2300 classes (including Swing), much faster.
  - Java 1.2 Playground
  - Java 1.3 Kestrel – HotSpot JVM included (JIT compilation)
  - Java 1.4 Merlin
- Java 5.0 Tiger (originally Java 1.5)
  - A number of new language features: metadata, autoboxing, enumerations, varargs, for each, etc.
- Java 6 Mustang
- Java 7 Dolphin
  - Binary literals, String in switch statement, try-with-resources, underscores, etc.
- Java 8 (March 2014)
  - Lambda expression
- Java 10 (March 2018)
- Java 11 (Sep. 2018)

Java Development Kit

- Java Platform
  - Java Runtime Environment(JRE)
    - JVM, libraries, etc.
    - Required to run Java applications and applets.
  - Java Development Kit(JDK)
    - JRE+javac, debugger, depolyment tools, etc.
    - Required to develop Java applications and applets.
- Profiles
  - Standard Edition(SE)
  - Enterprise Edition(EE)
  - Micro Edition(ME)
- Download Java Platform (JDK) 8 or higher

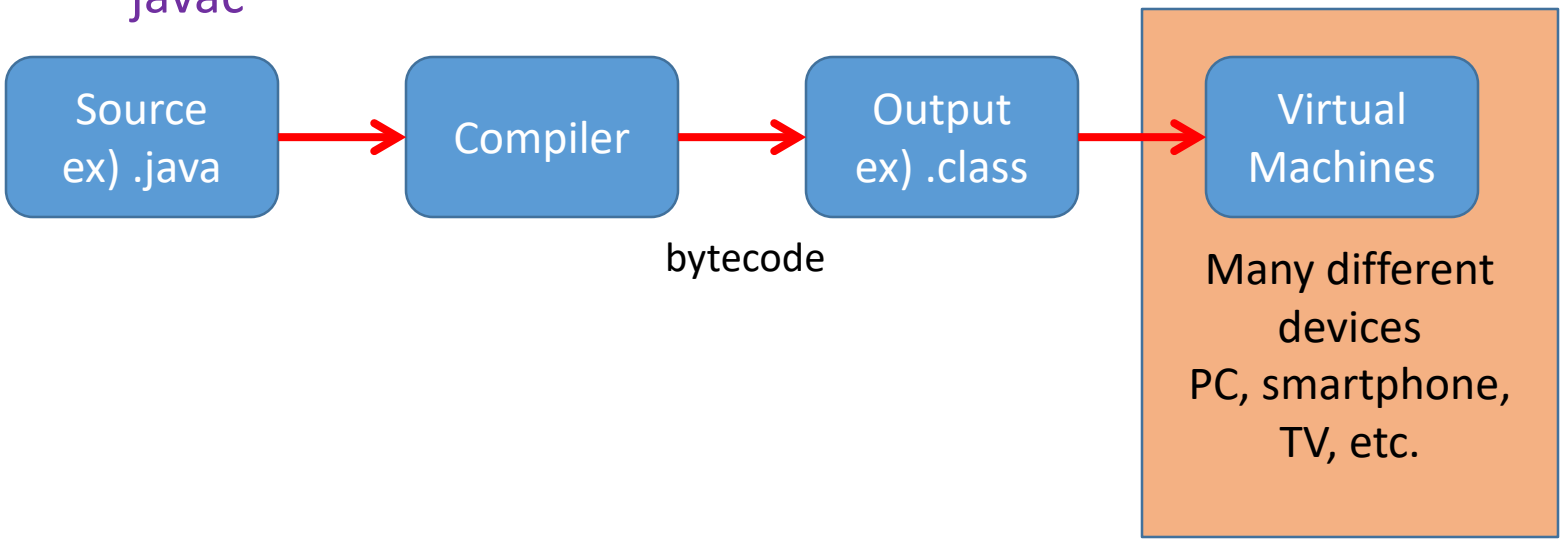
Java Development Kit

- Source code editor
  - MadEdit
  - AcroEdit
  - Notepad(메모장)
  - Vim
  - ...

- IDE Tools
  - eclipse
  - NetBeans
  - **IntelliJ IDEA**
  - Android Studio
  - etc.

Java – JVM & Compiler

- Java Virtual Machine (JVM)
  - java
- Compiler
  - javac



Sample Java code

```
int size = 27;
String name = "Fido";
Dog myDog = new Dog(name, size);
x = size - 5;
if (x < 15) myDog.bark(8);
```

```
while (x > 3) {
    myDog.play(); x += 1;
}
```

```
int[] numList = {2, 4, 6, 8};
System.out.print("Hello");
System.out.print("Dog: " + name);
String num = “8”;
int z = Integer.parseInt(num);
```

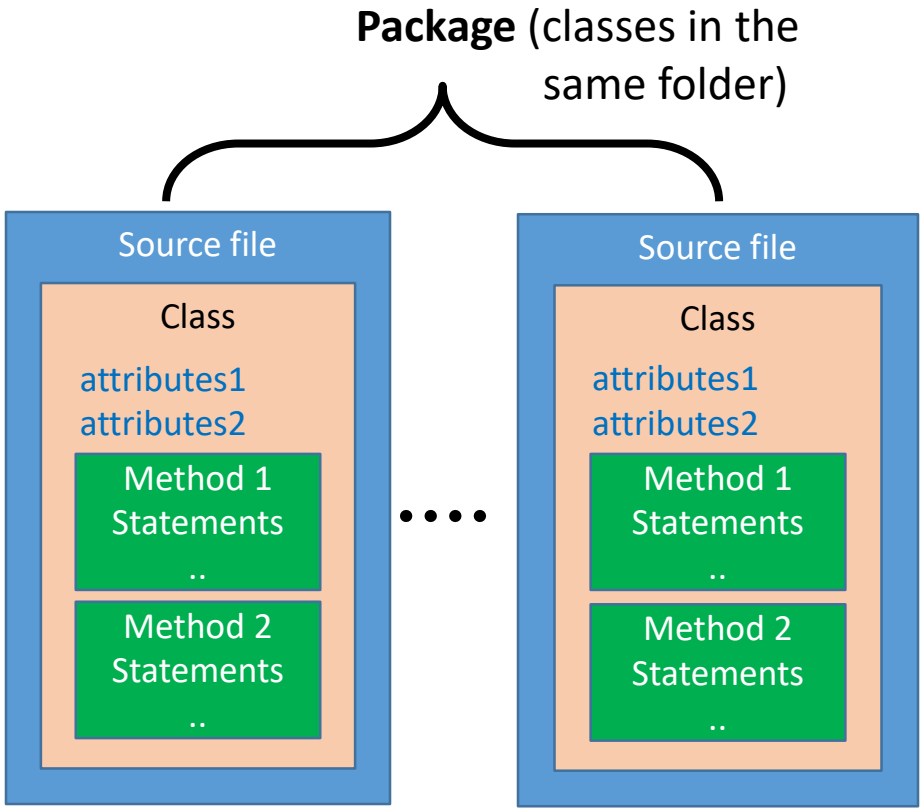
```
try {
    readTheFile("myFile.txt");
} catch (FileNotFoundException ex) {
    System.out.print("File not found. ");
}
```

Guess what each line of code is doing !!

Code Structure in Java

- Source file
- What goes in a **source** file?
  - Class
  - What goes in a **class**?
    - Variables (attributes)
    - Methods
    - What goes in a **method**?
      - Statements.

개과(CANIDAE)
- 바둑이
- 누렁이
- ..



Code structure

- Package
- Class

```
package org.foo.example;

public class Cube {
    ...

    public int getVolume() {
        int volume = width * height * depth;
        return volume;
    }
}
```

- Path

```
${CLASSPATH}/org/foo/example/Cube.class
```

I Rule !

- In Java (MyFirstApp.java)

```
public class MyFirstApp {

    public static void main (String[] args) {
        System.out.println("I Rule!");
        System.out.println(“The World");
    }
}
```

- In C (a.c)

```
#include <stdio.h>
void main (int argc, char * argv[]) {
    puts("I Rule!");
    puts("The World");
}
```

I Rule ! – Build & Run

- In Java (MyFirstApp.java)

```
$ javac MyFirstApp.java
$ Java MyFirstApp
I Rule!
The World
$ _
```

- In C (a.c)

```
$ gcc –c a.c
$ gcc –o a a.o
$ a
I Rule!
The World
$ _
```

Primitive Data Types

- Primitive data types
  - long val = 0b11010010_01101001_10010100_10010010L;
  - long hexBytes = 0xFF_EC_DE_5E;

type	desc.
`byte`	8-bit signed two's complement integer
`short`	16-bit signed two's complement integer
`int`	32-bit signed two's complement integer
`long`	64-bit signed two's complement integer; `256L`
`float`	single-precision 32-bit IEEE 754 floating point; `3.14f`
`double`	double-precision 64-bit IEEE 754 floating point; `3.14d`
`boolean`	`true` or `false`
`char`	16-bit Unicode charactor (UTF-16); `'\u0000' ~ '\uffff'`

- Strings: `String greeting = "Hello, world!";`

Operators

- Operator precedence

Operators	Precedence
postfix	`expr++ expr--`
unary	`++expr --expr +expr -expr ~ !`
multiplicative	`* / %`
additive	`+ -`
shift	`<< >> >>>`
rational	`< > <= >= instanceof`
equality	`== !=`
bitwise AND	`&`
bitwise exclusive OR	`^`
bitwise inclusive OR	`|`
logical AND	`&&`
logical OR	`||`
ternary	`? :`
assignment	`= += -= *= /= %= &= ^= |= <<= >>= >>>=`

Array

```
int[] anArrayA = new int[10];
// or
int[] anArrayB = {1, 2, 3, 8, 4, 13, 35, 4, 0 , -1};

anArrayA[3] = 4;
anArrayB[0] = anArrayA[8];
int lengthOfArray = anArrayA.length; // 10

anArrayA[10] = 9;
```

Statements

- Declarations, assignments, method calls, etc.

```
int x = 3;
String name = "Dirk";
x = x * 17;
System.out.print("x is " + x);
double d = Math.random();
```

Statements: Loops

- *for, while, do-while*

```
while (x > 12) {
    x = x - 1;
}
```

```
do { x = x – 1; } while (x > 12);
```

```
double[] data = {1.0, 2.8, 3.2};
for (i = 0; i < data.length; i++) {
    System.out.println("data: " + data[i]);
}
```

```
for (double datum : data) {
    System.out.println("data: " + datum);
}
```

Statements: if-then-else

- *if-else* test

```
if (x == 10) {
    System.out.println("x must be 10");
} else {
    System.out.println("x isn't 10");
}


if ((x < 3) && (name.equals("Dirk"))) {
    System.out.println("Gently");
}
System.out.println("this line runs no matter what");
```

Statements: Switch

- byte, short, char, int, String, enum, Character, Byte, Short, Integer

```
public String getTypeOfDay(int dayOfWeekArg) {
    String typeOfDay = null;
    switch (dayOfWeekArg) {
        case 0:
            typeOfDay = “Start of work week";
            break;
        case 1:
        case 2:
        case 3:
            typeOfDay = “Midweek";
            break;
        case 4:
            typeOfDay = “End of work week";
            break;
        case 5:
        case 6:
            typeOfDay = “Weekend";
            break;
        default:
            System.out.println(“Invalid value.”);
    }
    return typeOfDay;
}
```

Statements: Switch

- byte, short, char, int, String, enum, Character, Byte, Short, Integer

```
public String getTypeOfDay(String dayOfWeekArg) {
    String typeOfDay = null;
    switch (dayOfWeekArg) {
        case "Monday":
            typeOfDay = “Start of work week";
            break;
        case "Tuesday":
        case "Wednesday":
        case "Thursday":
            typeOfDay = “Midweek";
            break;
        case "Friday":
            typeOfDay = “End of work week";
            break;
        case "Saturday":
        case "Sunday":
            typeOfDay = “Weekend";
            break;
        default:
            System.out.println(“Invalid value.”);
    }
    return typeOfDay;
}
```

Statements: Branch (1)

- *break, continue, return*

```
for (i = 0; i < 8; i++) {
    if (data[i] == 3) {
        break;
    }
}

for (i = 0; i < 8; i++) {
    if (i == 2) {
        continue;
    }
    sum += i;
}
```

Statements: Branch (2)

- labeled *continue and break;*

```
...


   for (int i = 0; i < 8; i++) {
      data = storage[i].getData();
      for (int j = 0; j < 256; j++) {
          if (data[j] == -1) {
              continue;
          }
          sum += data[j];
      }
      success++;
   }
```

Statements: Branch (2)

- labeled *continue and break;*

```
...


   for (int i = 0; i < 8; i++) {
      data = storage[i].getData();
      for (int j = 0; j < 256; j++) {
          if (data[j] == -1) {
              break;
          }
          sum += data[j];
      }
      success++;
   }
```

Statements: Branch (2)

- labeled *continue and break;*

```
   int j;
...

   for (int i = 0; i < 8; i++) {
      data = storage[i].getData();
      for (j = 0; j < 256; j++) {
          if (data[j] == -1) {
              break;
          }
          sum += data[j];
      }
      if (j != 256) continue;
      success++;
   }
```

Statements: Branch (2)

- labeled *continue and break;*

```
...


   for (int i = 0; i < 8; i++) {
      data = storage[i].getData();
      for (int j = 0; j < 256; j++) {
          if (data[j] == -1) {
              continue;
          }
          sum += data[j];
      }
      success++;
   }
```

Statements: Branch (2)

- labeled *continue and break;*

```
...

test:
   for (int i = 0; i < 8; i++) {
      data = storage[i].getData();
      for (int j = 0; j < 256; j++) {
          if (data[j] == -1) {
              continue test; // labeled continue
          }
          sum += data[j];
      }
      success++;
   }
```

Q&A