02_1 Java Basics

Object-Oriented Programming

Basic structure of a Java program (1/5)

```
/**
 * This is a JavaDoc comment to generate documentation
 */
public class HelloWorld1 {
    public static void main(String[] args) {
        // Single statement
        System.out.println("Hello, World!");

        // Block of statements
        {
            System.out.println("This is a block.");
            System.out.println("It contains multiple statements.");
        }
    }
}
```

- public: access modifier
  - The class can be used anywhere
- class: keyword for class definition
- HelloWorld1: the name of the class
- begins with { and ends with }

Basic structure of a Java program (2/5)

```
/**
 * This is a JavaDoc comment to generate documentation
 */
public class HelloWorld1 {
    public static void main(String[] args) {
        // Single statement
        System.out.println("Hello, World!");

        // Block of statements
        {
            System.out.println("This is a block.");
            System.out.println("It contains multiple statements.");
        }
    }
}
```

- Method “main”
- The first method executed when the program starts
- Should be included in any executable program

Basic structure of a Java program (3/5)

```
/**
 * This is a JavaDoc comment to generate documentation
 */
public class HelloWorld1 {
    public static void main(String[] args) {
        // Single statement
        System.out.println("Hello, World!");

        // Block of statements
        {
            System.out.println("This is a block.");
            System.out.println("It contains multiple statements.");
        }
    }
}
```

- public: access modifier
  - main method can be called from anywhere
- static: method belongs class not instance
- void: no return type
- main: method name
- Strings[] args: array of Strings, command line arguments

Basic structure of a Java program (4/5)

```
/**
 * This is a JavaDoc comment to generate documentation
 */
public class HelloWorld1 {
    public static void main(String[] args) {
        // Single statement
        System.out.println("Hello, World!");

        // Block of statements
        {
            System.out.println("This is a block.");
            System.out.println("It contains multiple statements.");
        }
    }
}
```

multi-line comments: /* .... */

/** ... */: can be captured by JavaDoc tool (automatic manual generation)

single-line comments: // .....

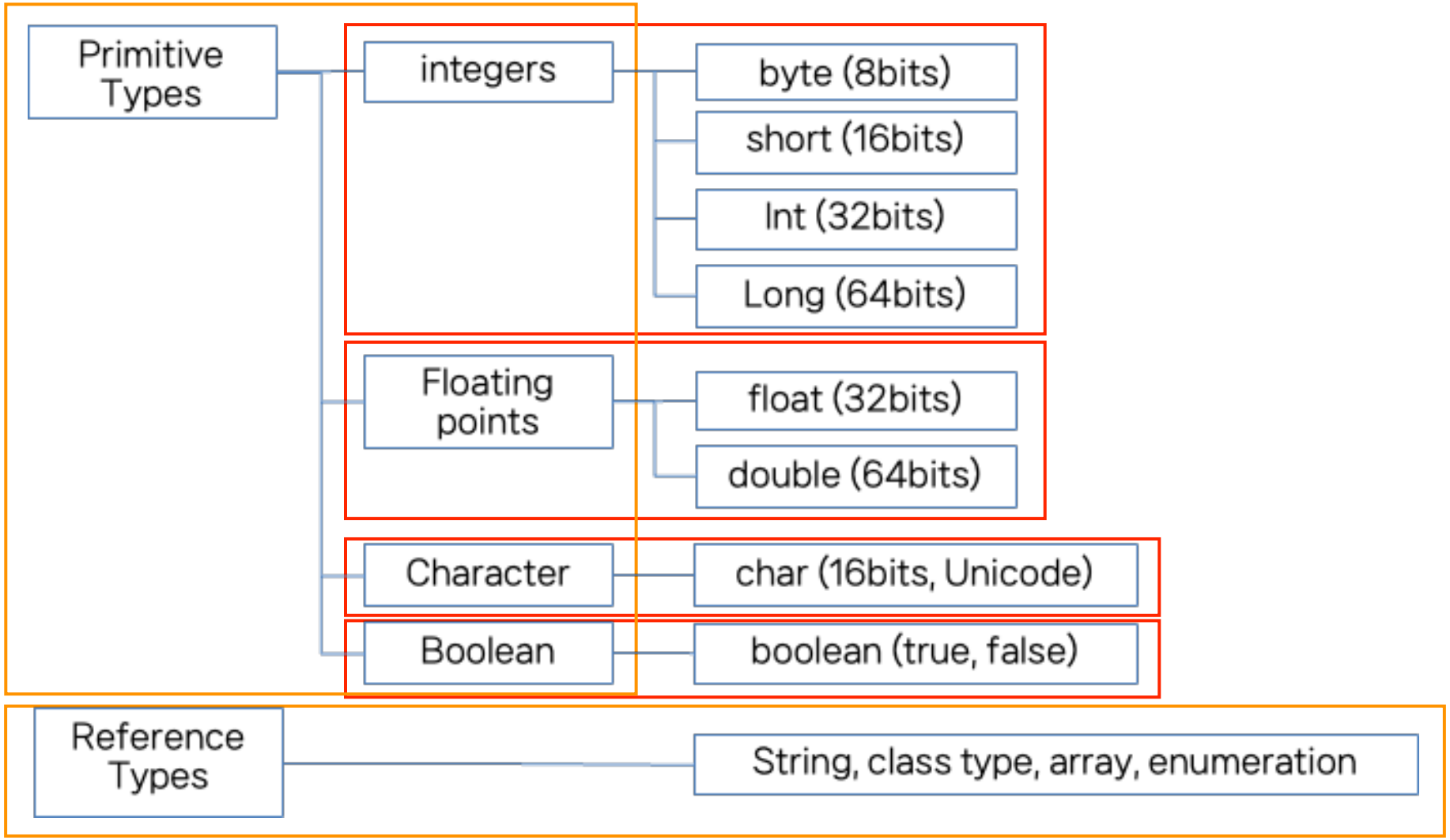
Basic structure of a Java program (5/5)

```java
/**
 * This is a JavaDoc comment to generate documentation
 */
public class HelloWorld1 {
    public static void main(String[] args) {
        // Single statement
        System.out.println("Hello, World!");

        // Block of statements
        {
            System.out.println("This is a block.");
            System.out.println("It contains multiple statements.");
        }
    }
}
```

Identifiers

- Identifiers
  - The name of variables, constants, methods, parameters, classes, ...
- Identifier Rules
  - **Case sensitivity**: MyVariable ≠ myVariable
  - **Reserved words (keywords) cannot be used**: class, public, void, …
  - **General Naming Rules:**
    - Cannot start with numeric characters
    - Cannot use special characters except $ and _
  - **Naming Conventions**:
    - camelCase for variables and methods: myVariable
    - PascalCase for classes: MyClass

Data Types


Primitive Types
integers
byte (8bits)
short (16bits)
Int (32bits)
Long (64bits)
Floating points
float (32bits)
double (64bits)
Character
char (16bits, Unicode)
Boolean
boolean (true, false)
Reference Types
String, class type, array, enumeration

Data Types: Example

```
public class DataTypesExample {
    public static void main(String[] args) {
        int myNumber; // variable declaration
        myNumber = 10; // initialization
        int yourNumber = 10; // declaration with initialization
        float f1 = 3.151492f; // float type literal should be ended with the suffix 'f'
        double d1 = 3.151492; // double type literal doesn't have to have any suffix.

        System.out.println("myNumber=" + myNumber + " yourNumber=" + yourNumber +
                " f1=" + f1 + " d1=" + d1);
    }
}
```

```
OUTPUT:
myNumber=10 yourNumber=10 f1=3.151492 d1=3.151492
```

Type Conversion

```java
public class TypeConversion {
    public static void main(String[] args) {

        int myInt = 10;
        double myDouble = myInt;    // Implicit conversion from int to double
                                    // range of (double) > range of (int)
        System.out.println("myInt(" + myInt + ") myDouble(" + myDouble + ")");

        myDouble = 9.78;
        myInt = (int) myDouble;     // Explicit conversion from double to int

        System.out.println("myInt(" + myInt + ") myDouble(" + myDouble + ")");
    }
}
```

```
OUTPUT:
myInt(10) myDouble(10.0)
myInt(9) myDouble(9.78)
```

Naming Constants

- Named constants having names:

```
public static final int INCHES_PER_FOOT = 12;
public static final double RATE = 0.14;
```

  - Cannot change the value in the program

  - Naming convention for constants: Use all uppercase letters, and designate word boundaries with an underscore character

Strings

- A class used to handle text

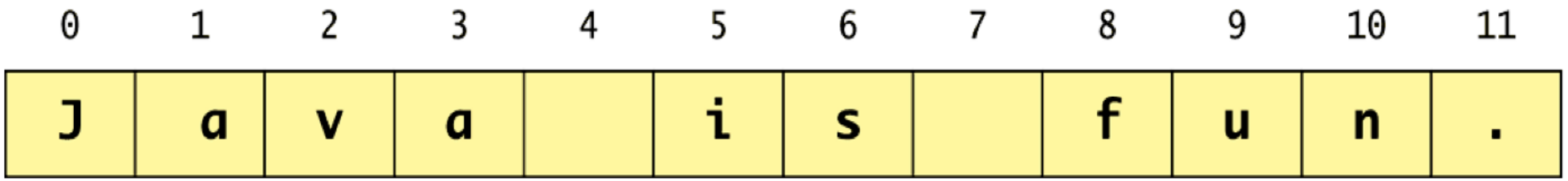
```
public class StringClass {
    public static void main(String[] args) {
        String greeting = "Hello, World!";
        String firstName = "John";
        String lastName = "Doe";

        // Concatenation
        String fullName = greeting + " " + firstName + " " + lastName;

        System.out.println(fullName); // Outputs "Hello, World! John Doe"
    }
}
```

Concatenation of Strings

```
String str4 = “The answer is " + 42     // “The answer is 42”
int k = 35;
String str5 = “Yes ” + k;               // “Yes 35”
```

String Indexes

"Java is fun."



String Methods (1/3)

```
public class StringMethods {
    public static void main(String[] args) {

        String str = "Hello, World!";

        int length = str.length(); // length of the string: 13
        char ch = str.charAt(0); // character at a specific index position: 'H'

        // substring from the given begin index to the end: "World!"
        String substr1 = str.substring(7);

         // substring from index1 to index2: "Hello"
        String substr2 = str.substring(0, 5);

        // str과 given String의 내용 비교 (reference, 즉, 주소 비교 아님) : true
        boolean isEqual = str.equals("Hello, World!”);

        // 대소문자 구분없이 내용 비교: true
        boolean isEqualIgnoreCase = str.equalsIgnoreCase("hello, world!");
```

String Methods (2/3)

```
// dictionary order로 str > "Hello" 이면 positive, str < "Hello" 이면 negative
// str == "Hello" 이면 0을 return
int comparison = str.compareTo("Hello"); // Positive value
int comparisonIgnoreCase = str.compareToIgnoreCase("hello"); // Positive value

int index = str.indexOf("World");       // 처음 출현하는 World의 W의 index: 7
int lastIndex = str.lastIndexOf("o");   // 마지막 출현하는 o의 index: 8
boolean contains = str.contains("Hello"); // 주어진 substring을 포함하는가? true

String replacedStr = str.replace('o', 'a'); // "Hella, Warld!"
String replacedStr2 = str.replace("World", "Java"); // "Hello, Java!"
String replacedAllStr = str.replaceAll("l", "L"); // "HeLLo, WorLd!"
String replacedFirstStr = str.replaceFirst("l", "L"); // "HeLlo, World!"

String upper = str.toUpperCase(); // 대문자로: "HELLO, WORLD!"
String lower = str.toLowerCase(); // 소문자로: "hello, world!"
```

String Methods (3/3)

```
        String trimmedStr = str.trim(); // 앞뒤 공백이 제거된 문자열
        String[] words = str.split(", "); // words[0] = “Hello”, words[1] = “World!”
        String joinedStr = String.join(", ", "Hello", "World"); // "Hello, World"

        String intStr = String.valueOf(123);  // integer 123을 String "123" 으로
        String boolStr = String.valueOf(true); // boolean true를 String "true"로

        boolean startsWith = str.startsWith("Hello"); // true
        boolean endsWith = str.endsWith("!"); // true
        boolean isEmpty = str.isEmpty(); // false
    }
}
```

Escape Sequences

1) 프로그램에서 특별한 의도로 사용되는 문자
2) 눈에 안보이는 특수 문자

```
\" Double quote.
\' Single quote.
\\ Backslash.
\n New line. Go to the beginning of the next line.
\r Carriage return. Go to the beginning of the current line.
\t Tab. White space up to the next tab stop.
```

```
Ex) System.out.print(“Hey Guys\\\n\”Oh\t\’Yes!!”);

Hey Guys\
“Oh    ‘Yes!!
```

String Processing

- A **`String`** object in Java
  - immutable, i.e., the characters it contains cannot be changed

- **`StringBuffer`**
  - Can be changed

- [NOTE] Possible to change the value of a **`String`** variable by using an assignment statement

```
String name = "Soprano";
name = "Anthony " + name;
```

Character Sets - ASCII

- ASCII: A character set used by many programming languages that contains all the characters normally used on an English-language keyboard, plus a few special characters
  - Each character is represented by a particular number
  - 1 byte (8 bits)

Ctrl	Dec	Hex	Char	Code
^@	0	00		NUL
^A	1	01		SOH
^B	2	02		STX
^C	3	03		ETX
^D	4	04		EOT
^E	5	05		ENQ
^F	6	06		ACK
^G	7	07		BEL
^H	8	08		BS
^I	9	09		HT
^J	10	0A		LF
^K	11	0B		VT
^L	12	0C		FF
^M	13	0D		CR
^N	14	0E		SO
^O	15	0F		SI
^P	16	10		DLE
^Q	17	11		DC1
^R	18	12		DC2
^S	19	13		DC3
^T	20	14		DC4
^U	21	15		NAK
^V	22	16		SYN
^W	23	17		ETB
^X	24	18		CAN
^Y	25	19		EM
^Z	26	1A		SUB
^[	27	1B		ESC
^\	28	1C		FS
^]	29	1D		GS
^^	30	1E	▲	RS
^-	31	1F	▼	US

Dec	Hex	Char
32	20	
33	21	!
34	22	"
35	23	#
36	24	$
37	25	%
38	26	&
39	27	'
40	28	(
41	29	)
42	2A	*
43	2B	+
44	2C	,
45	2D	-
46	2E	.
47	2F	/
48	30	0
49	31	1
50	32	2
51	33	3
52	34	4
53	35	5
54	36	6
55	37	7
56	38	8
57	39	9
58	3A	:
59	3B	;
60	3C	<
61	3D	=
62	3E	>
63	3F	?

Dec	Hex	Char
64	40	@
65	41	A
66	42	B
67	43	C
68	44	D
69	45	E
70	46	F
71	47	G
72	48	H
73	49	I
74	4A	J
75	4B	K
76	4C	L
77	4D	M
78	4E	N
79	4F	O
80	50	P
81	51	Q
82	52	R
83	53	S
84	54	T
85	55	U
86	56	V
87	57	W
88	58	X
89	59	Y
90	5A	Z
91	5B	[
92	5C	\
93	5D	]
94	5E	^
95	5F	_

Dec	Hex	Char
96	60	`
97	61	a
98	62	b
99	63	c
100	64	d
101	65	e
102	66	f
103	67	g
104	68	h
105	69	i
106	6A	j
107	6B	k
108	6C	l
109	6D	m
110	6E	n
111	6F	o
112	70	p
113	71	q
114	72	r
115	73	s
116	74	t
117	75	u
118	76	v
119	77	w
120	78	x
121	79	y
122	7A	z
123	7B	{
124	7C	|
125	7D	}
126	7E	~
127	7F	⌂*

* ASCII code 127 has the code DEL. Under MS-DOS, this code has the same effect as ASCII 8 (BS). The DEL code can be generated by the CTRL + BKSP key.

Character Sets - Unicode

- Unicode: A character set used by the Java language
  - includes all the ASCII characters plus many of the characters used in languages with a different alphabet from English (ex. Korean)
  - 2 bytes (16 bits)