# JAVA PROGRAMMING

2023/10/15 12:00 PM

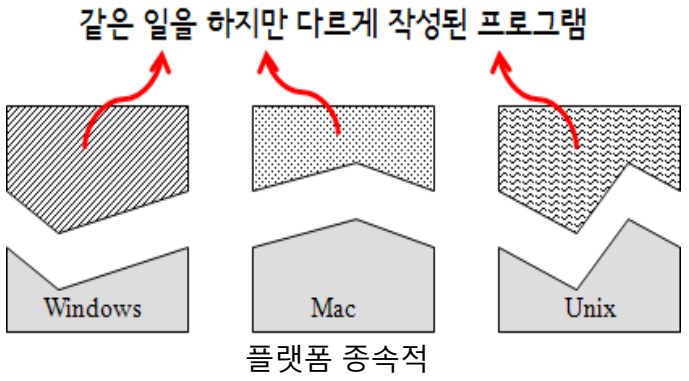
# CONTENTS

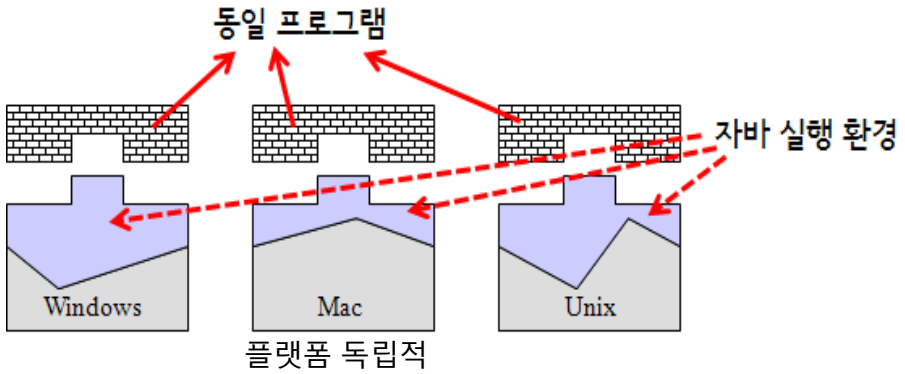
# INTRODUCTION

- Java
- Java Platform
- How Java Works
- Java Packages

# Java

- Object-oriented language (OOP; 객체 지향 프로그래밍 언어)
- HW/OS independent (virtual machine based)
- Type system: Static, Strong, Safe
- Supports garbage collection.
- Simple and lean code.
- Modularity
- Secure programs.



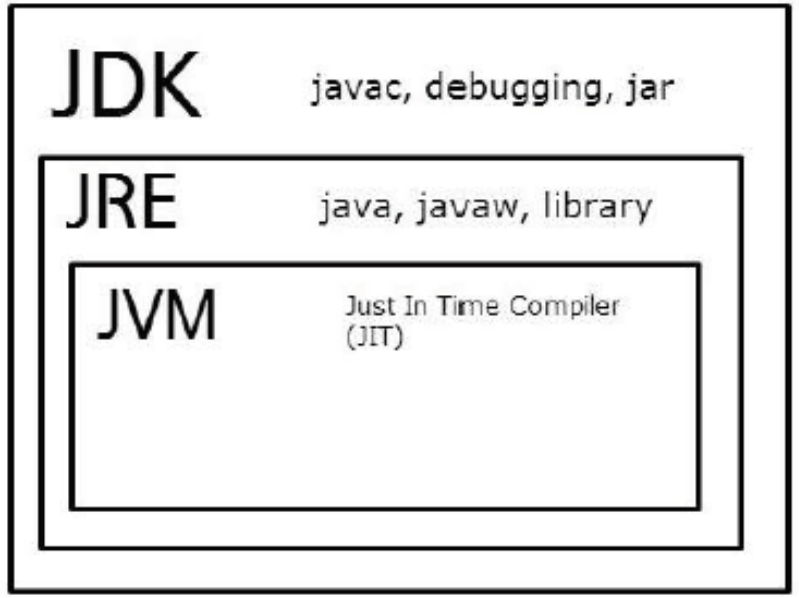



# Java Platform

- **JDK** (Java Development Kit)

  : JRE + development tools

  - Needed for Java programming
  - Has compilers, debuggers, JavaDoc




- **JRE** (Java Runtime Environment)

  : libraries + other files that JVM uses at runtime

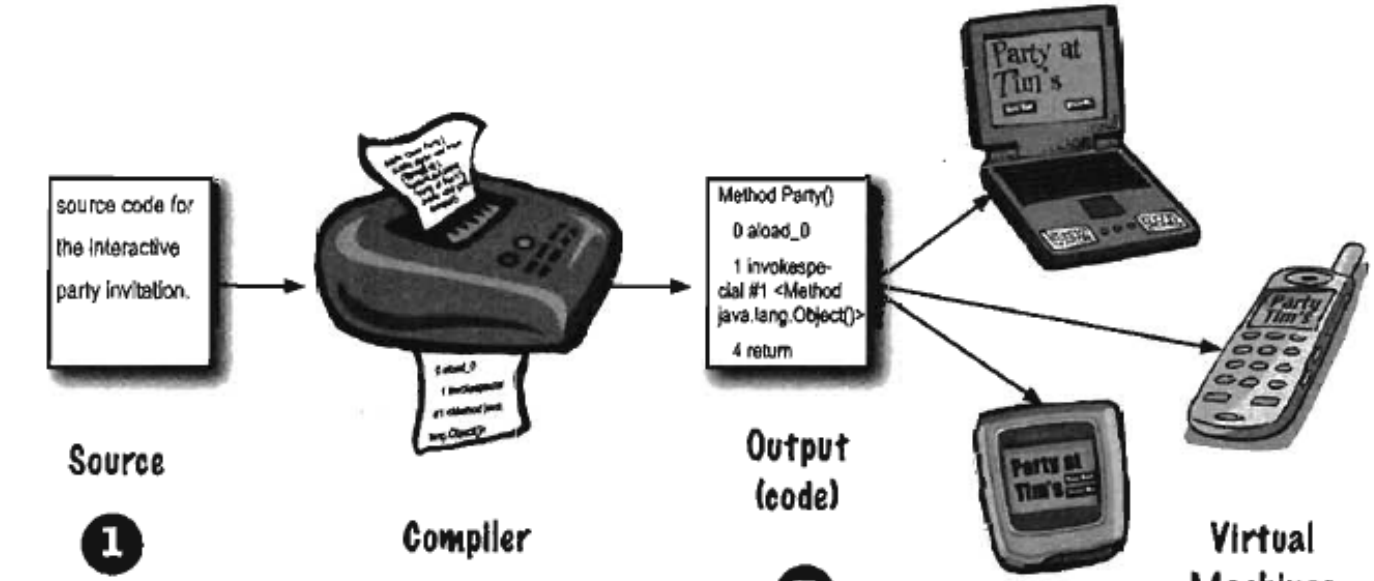
  - Only for running Java programs on browser or computer

- **JVM** (Java Virtual Machine)

  : interprets the class files' orders (to bytecode) for the OS to run

  - dependent on OS (Windows, Mac, Linux)
  - "Write once, run anywhere."

# How Java Works




**Source**

1

Create a source document. Use an established protocol (in this case, the Java language).

**Compiler**

2

Run your document through a source code compiler. The compiler checks for errors and won't let you compile until it's satisfied that everything will run correctly.

**Output (code)**

3

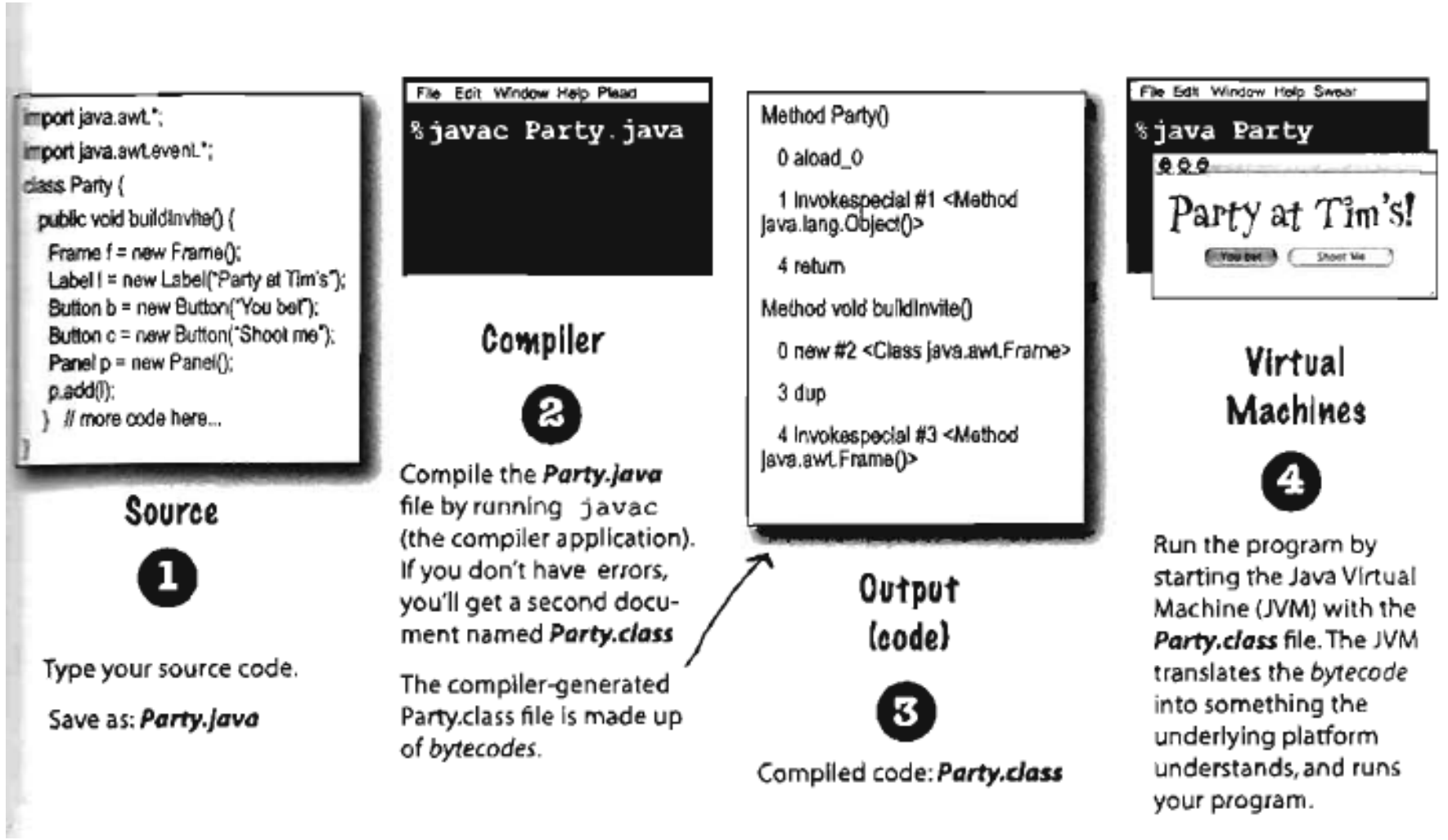
The compiler creates a new document, coded into Java ***bytecode***. Any device capable of running Java will be able to interpret/translate this file into something it can run. The compiled bytecode is platform-independent.

**Virtual Machines**

4

Your friends don't have a physical Java Machine, but they all have a ***virtual*** Java machine (implemented in software) running inside their electronic gadgets. The virtual machine reads and *runs* the bytecode.

# How Java Works (cont.)



# Java Packages

- Package = Directory; has similar functions
- Java classes can be grouped inside a package.
- Java API packages are provided. (https://docs.oracle.com/javase/8/docs/api/)
- Java API provides classes and packages:
  - * `java.lang` : support classes, math functions, thread, etc…
  - * `java.util` : utility classes (hash tables, random, etc…)
  - * `java.io` : input/output supporting classes
  - * `java.awt` : GUI supporting classes
- Uses '.' to access child packages.

  ex) `java.lang`, `java.util`

# Java Packages (cont.)

- How to use classes inside a package?
  - Ex) Want to use a class "`ArrayList`" inside `java.util` package:
- Option 1) Declare directly

```
java.util.ArrayList myList = new java.util.ArrayList;
```

- Option 2) **Use `import` statement**

```
import java.util.ArrayList;
ArrayList myList = new ArrayList;
```

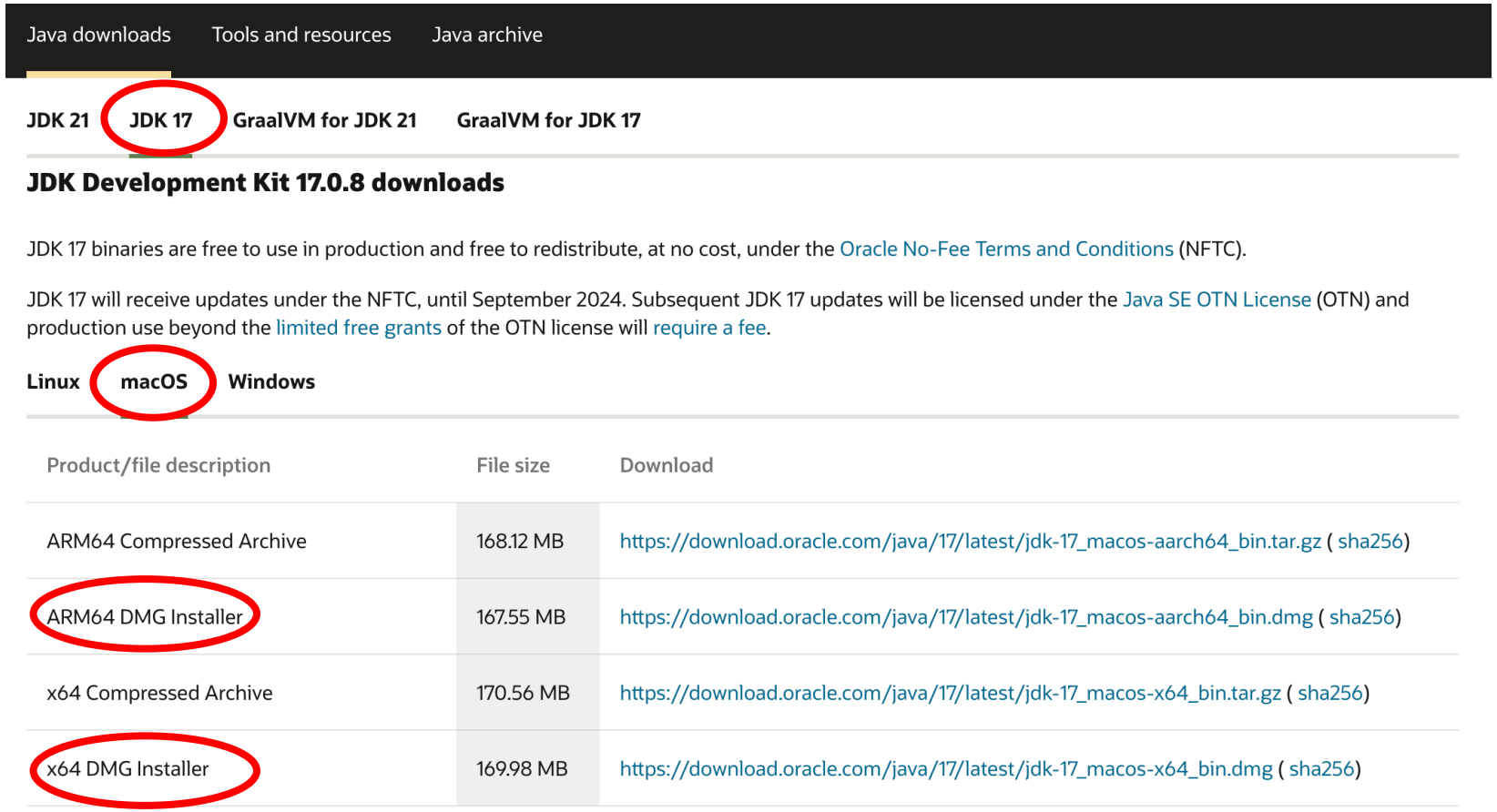
- **If you want to use all the classes inside a package, use ***

```
import java.util.*;
```

# ENVIRONMENT SETTINGS

# 1. Installing JDK

- https://www.oracle.com/java/technologies/downloads/



Java downloads | Tools and resources | Java archive

JDK 21 | JDK 17 | GraalVM for JDK 21 | GraalVM for JDK 17

**JDK Development Kit 17.0.8 downloads**

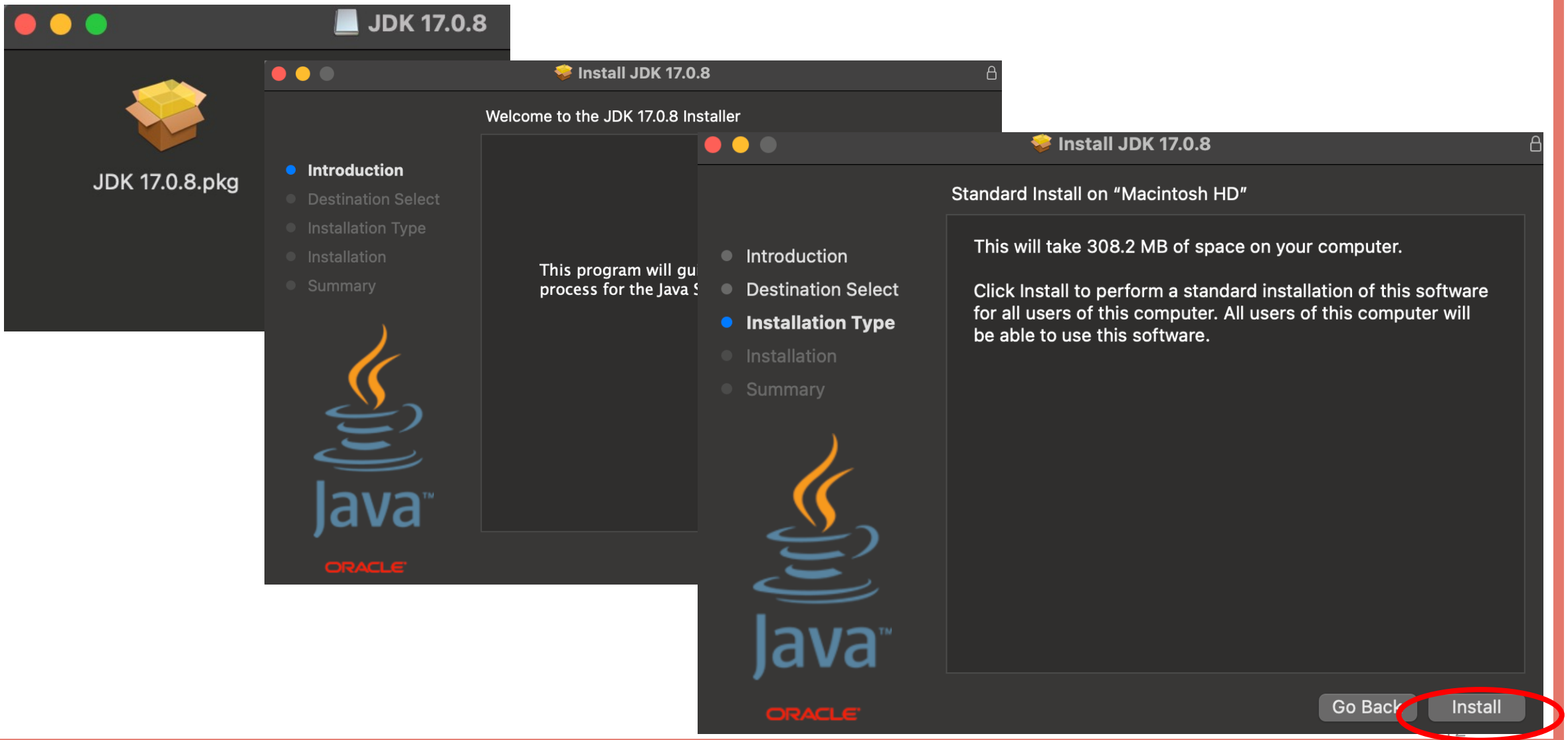
JDK 17 binaries are free to use in production and free to redistribute, at no cost, under the Oracle No-Fee Terms and Conditions (NFTC).

JDK 17 will receive updates under the NFTC, until September 2024. Subsequent JDK 17 updates will be licensed under the Java SE OTN License (OTN) and production use beyond the limited free grants of the OTN license will require a fee.

Linux | macOS | Windows

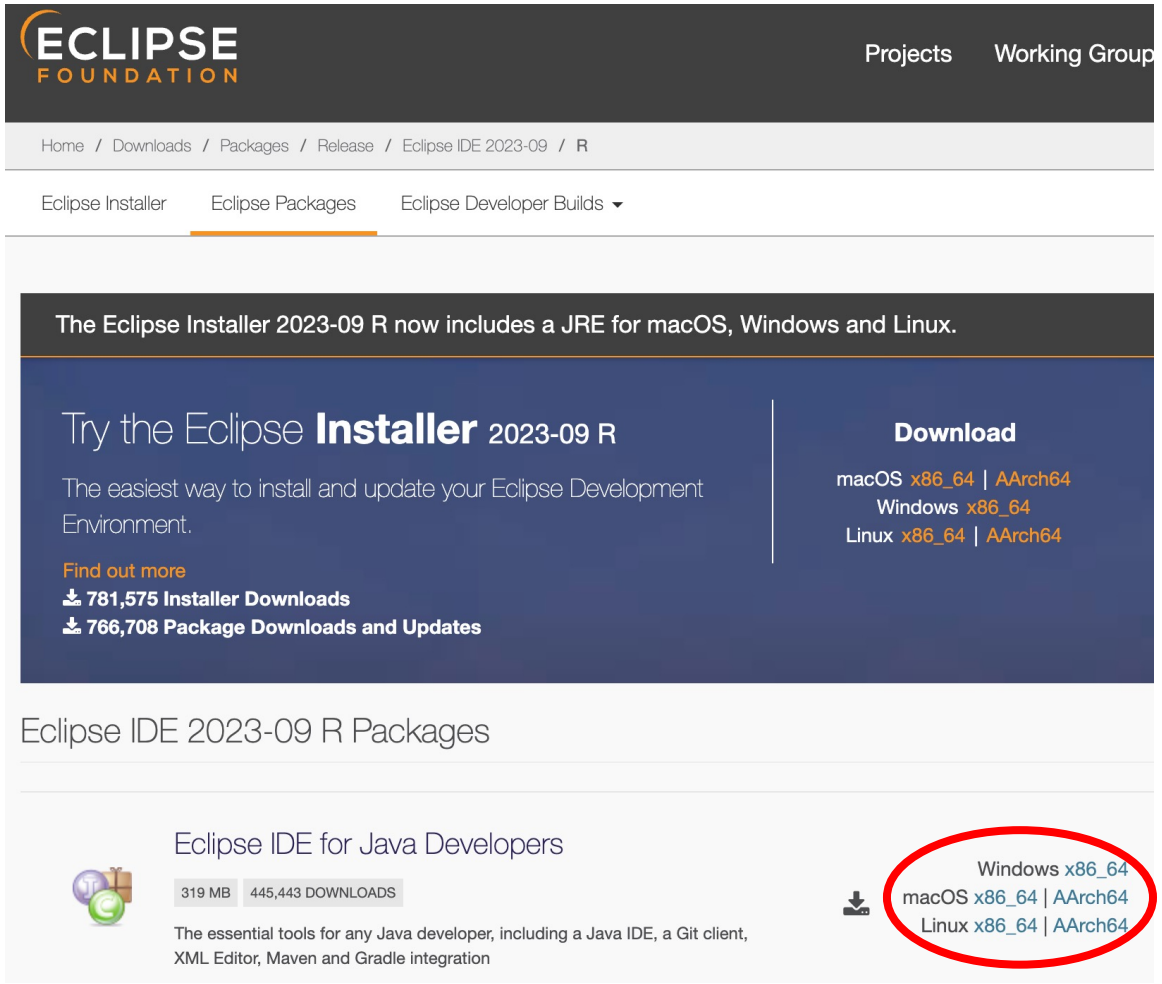
| Product/file description | File size | Download |
|---|---|---|
| ARM64 Compressed Archive | 168.12 MB | https://download.oracle.com/java/17/latest/jdk-17_macos-aarch64_bin.tar.gz ( sha256) |
| ARM64 DMG Installer | 167.55 MB | https://download.oracle.com/java/17/latest/jdk-17_macos-aarch64_bin.dmg ( sha256) |
| x64 Compressed Archive | 170.56 MB | https://download.oracle.com/java/17/latest/jdk-17_macos-x64_bin.tar.gz ( sha256) |
| x64 DMG Installer | 169.98 MB | https://download.oracle.com/java/17/latest/jdk-17_macos-x64_bin.dmg ( sha256) |

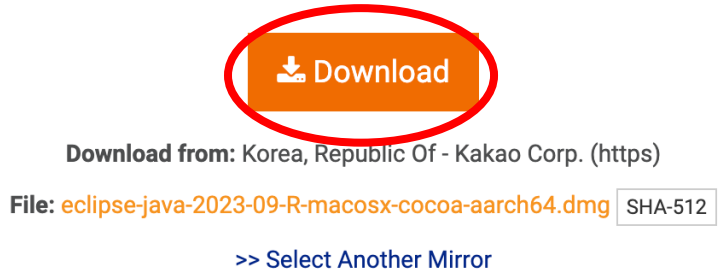
# 1. Installing JDK (cont.)



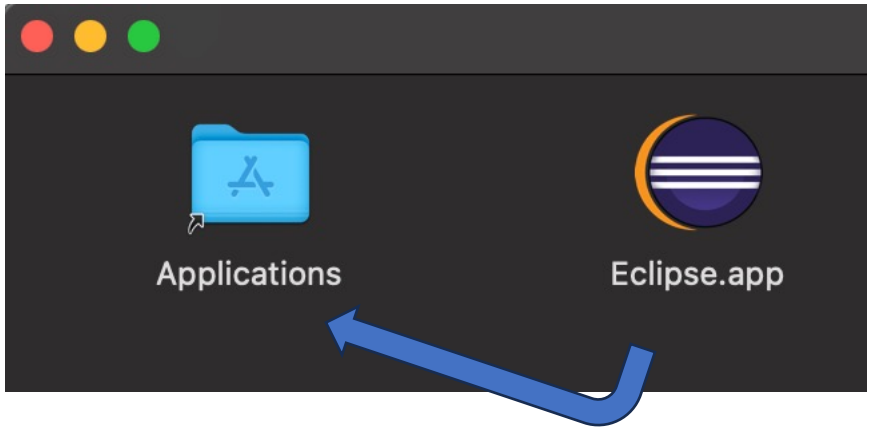
# 2. Installing Eclipse

- https://www.eclipse.org/downloads/packages/



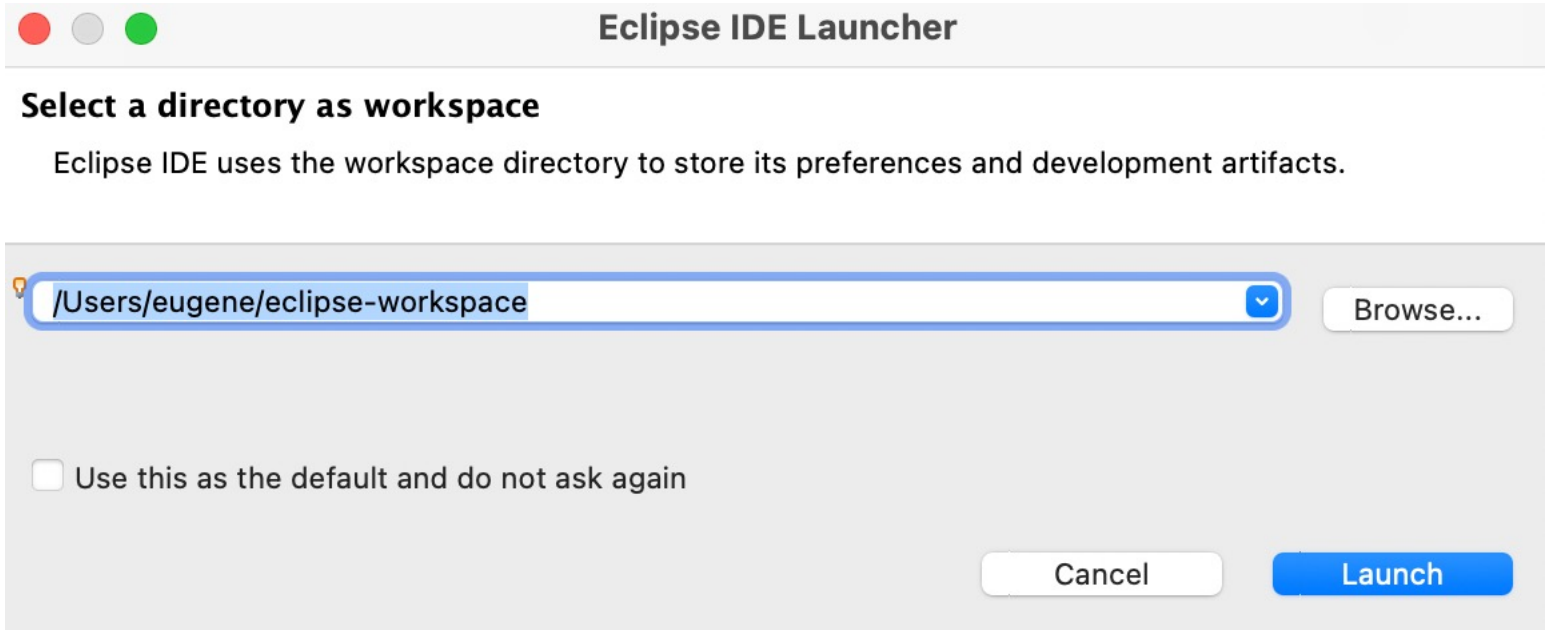
# 2. Installing Eclipse (cont.)



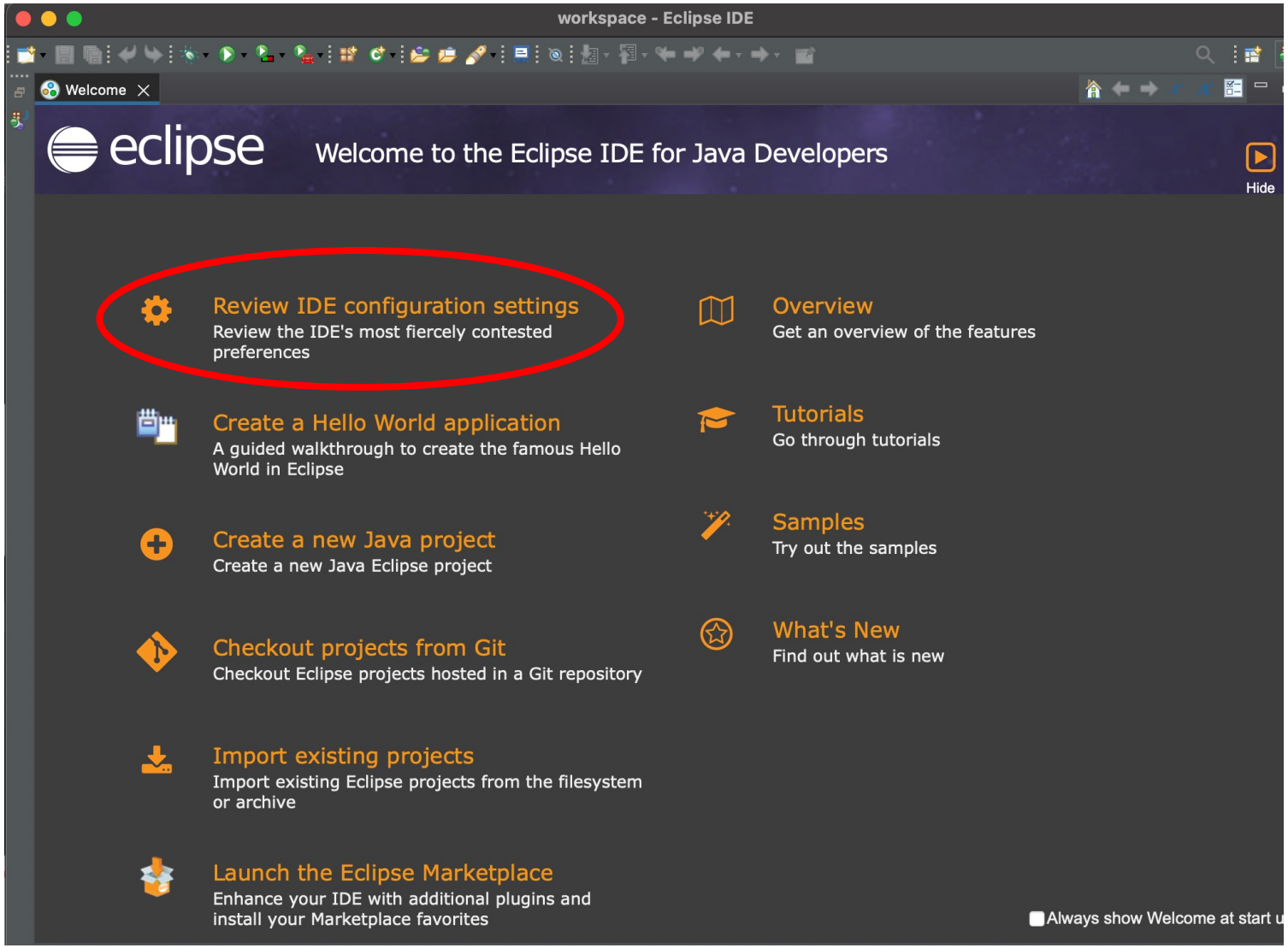

Drag and Drop

# 3. Set up Eclipse Workspace

- Make a workspace folder where you want to work on.
- Directory is recommended to be in English (not Korean) without blank spaces or special letters.



# 3. Set up Eclipse Workspace (cont.)



# 3. Set up Eclipse Workspace (cont.)


## Refresh Resources Automatically?

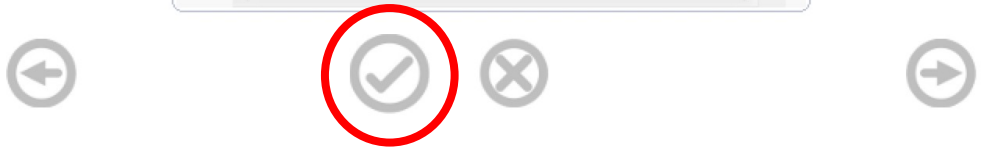
AbstractIterator.java

```
22 public abstract class AbstractIterator<T> implements Iterator<T>
23 {
24   /**
25    * The token to be used in {@link #computeNextElement()} to indicate the
26    */
27   protected static final Object END_OF_DATA = new Object();
28
29   private boolean nextComputed;
30
31   private T next;
32
33   public AbstractIterator()
```


## Show Line Numbers in Editors?

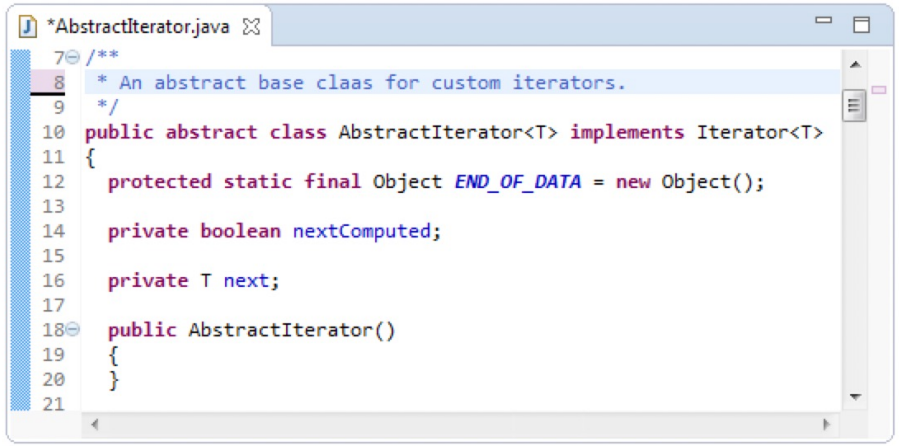
*HelloWorld.java

```
6  /*
7   * Example snippet: Hello World
8   * http://www.eclipse.org/swt/snippets
9   */
10 public class HelloWorld {
11   public static void main(String[] args) {
12     Display display = new Display();
13     Shell shell = new Shell(display);
14     shell.setText("Hello World");
15     shell.open();
16     while (!shell.isDisposed()) {
17       if (!display.readAndDispatch()) {
18         display.sleep();
19       }
20     }
21     display.dispose();
```



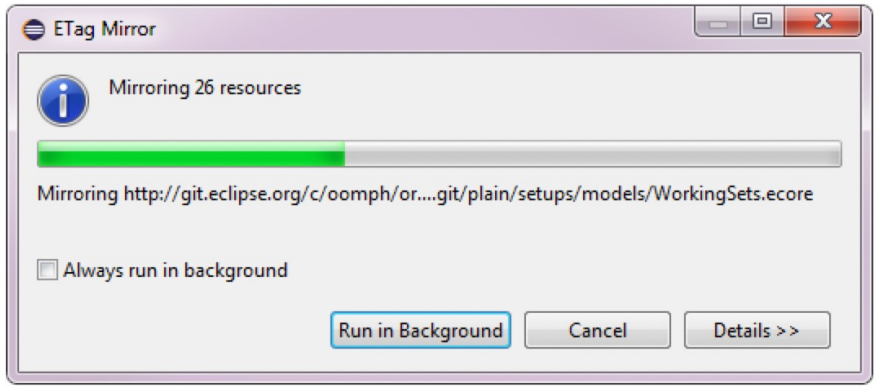
# 3. Set up Eclipse Workspace (cont.)


## Check Spelling in Text Editors?



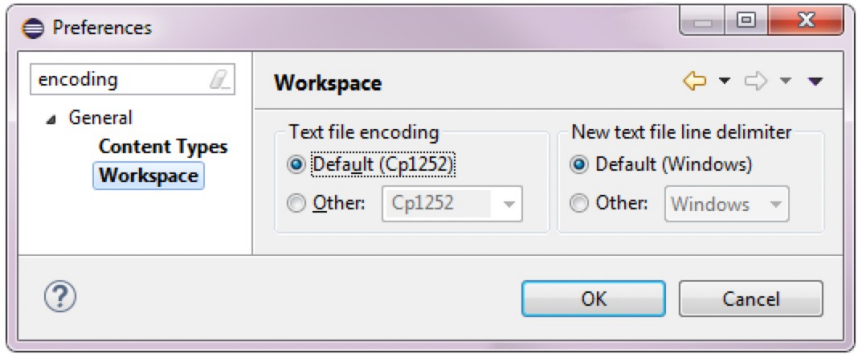


## Execute Jobs in Background?



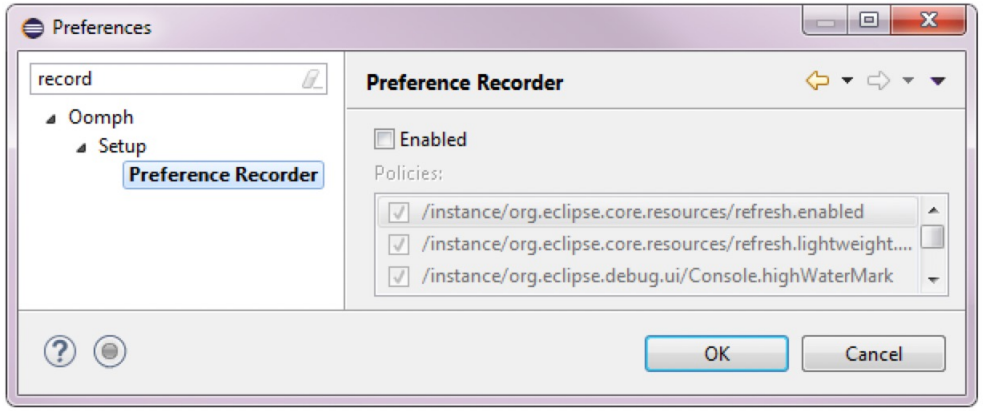

# 3. Set up Eclipse Workspace (cont.)


## Encode Text Files with UTF-8?





## Enable Preference Recorder?




# 3. Set up Eclipse Workspace (cont.)

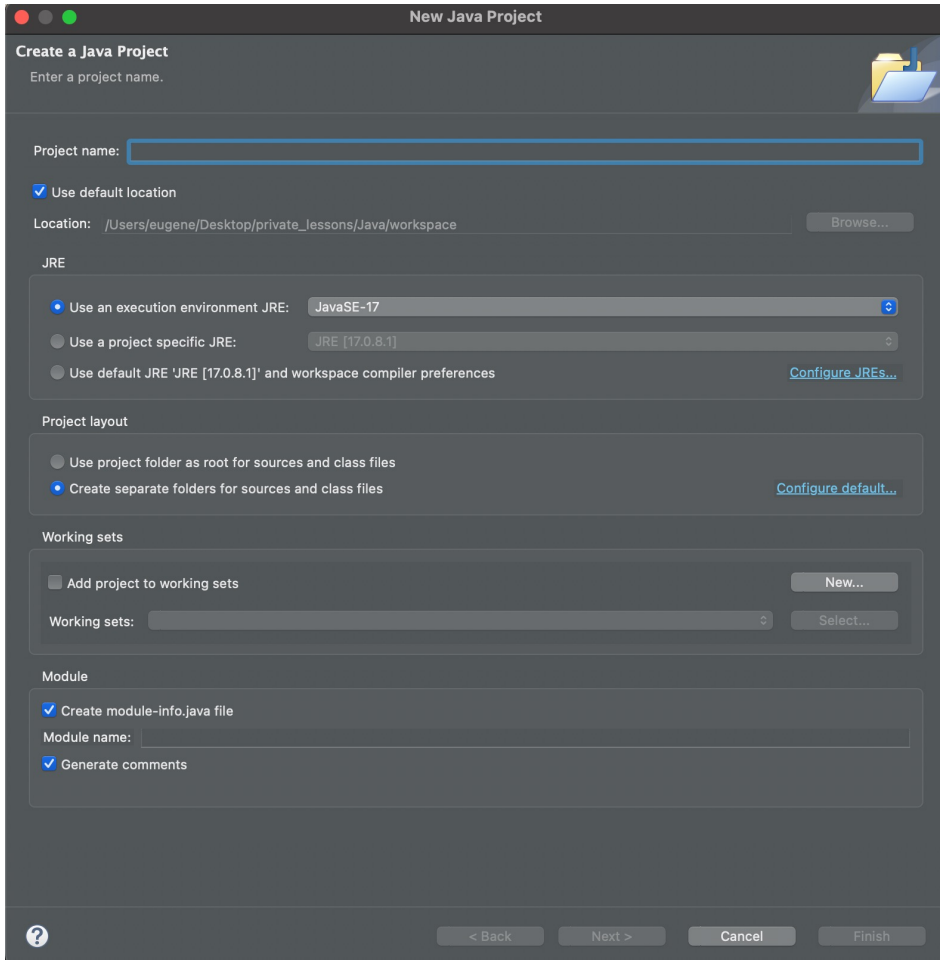

## Summary

- Refresh Resources Automatically
- Show Line Numbers in Editors
- Check Spelling in Text Editors
- Execute Jobs in Background
- Encode Text Files with UTF-8
- Enable Preference Recorder

Finish

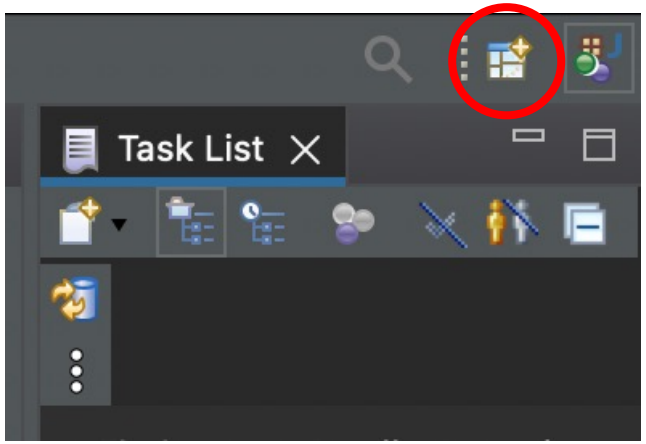
# 4. Create a Java Project

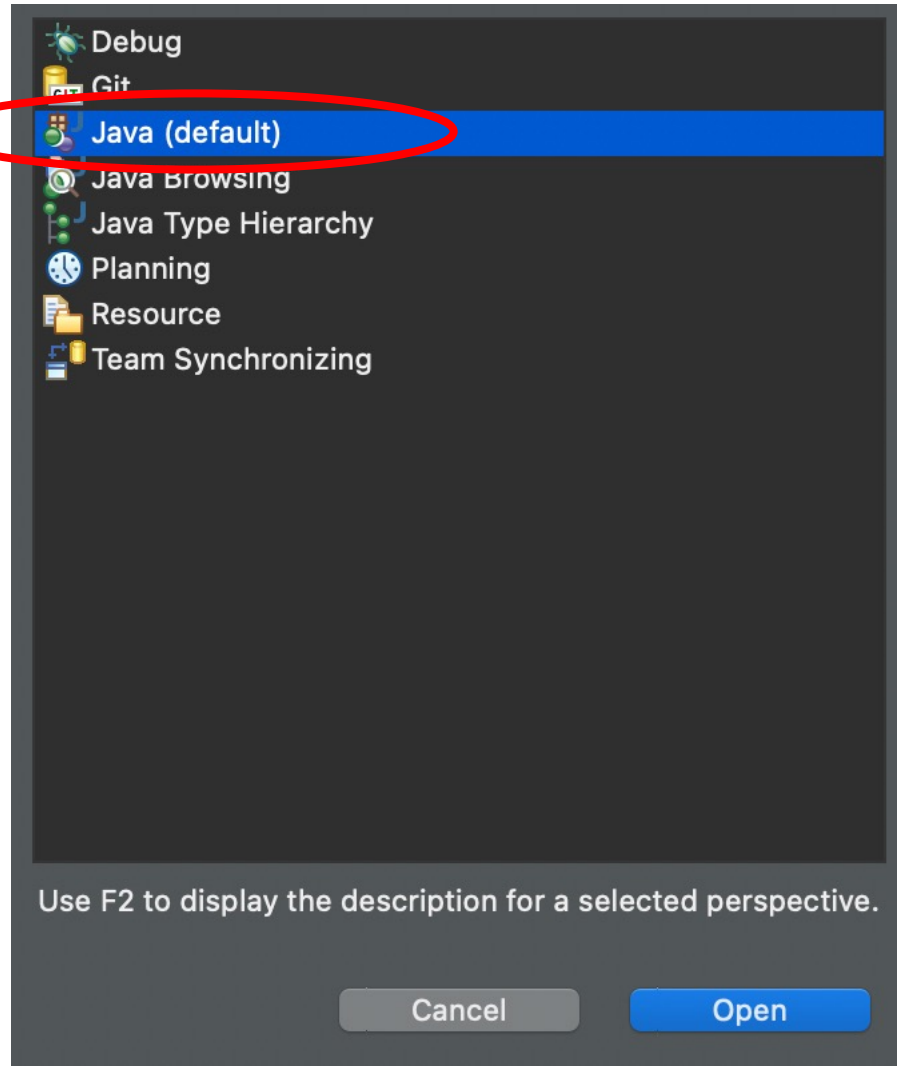
- Project name is recommended to be named in English without blanks. (same as directory namings)



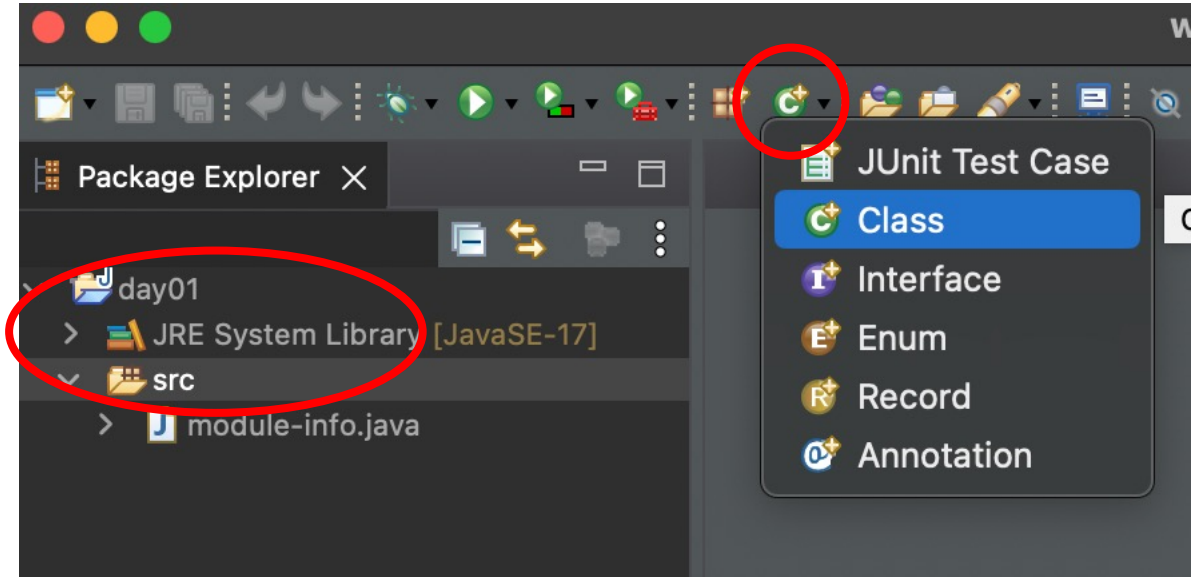
# 4. Create a Java Project (cont.)

- Check if Perspective is in Java (default).



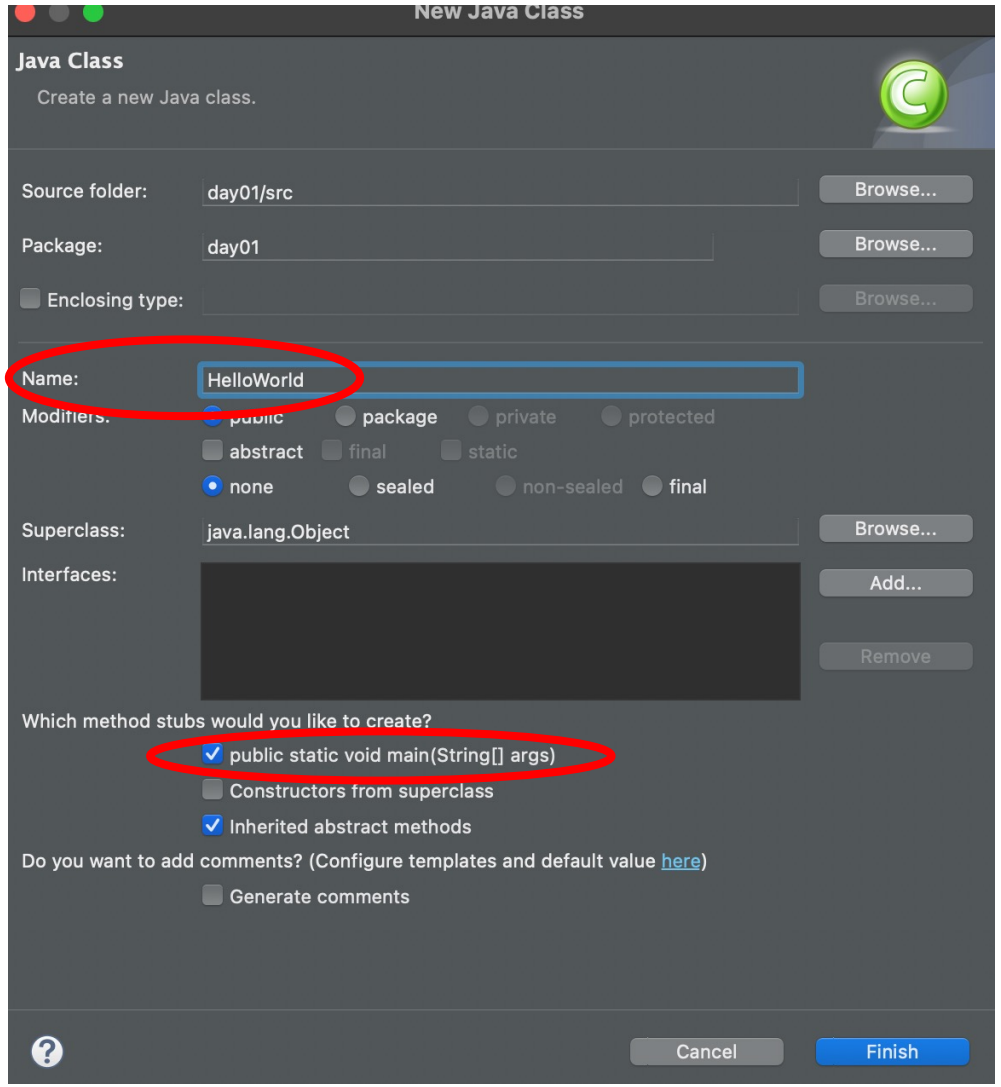
# 5. Create a Class File



# 5. Create a Class File (cont.)

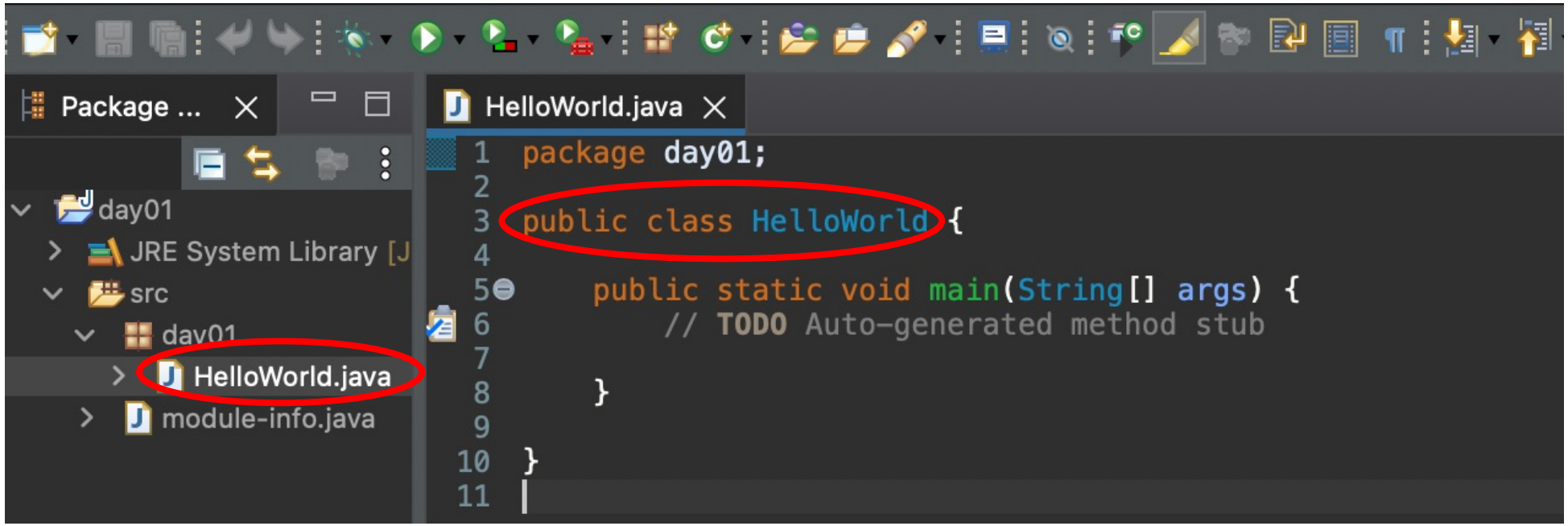
- Java type names usually start with an uppercase letter by convention.

- Check public static void main(String[] args) option if this is the main class.



# 5. Create a Class File (cont.)

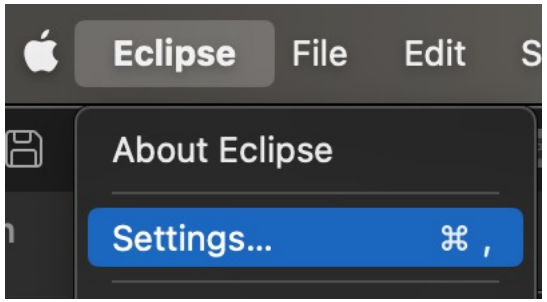
- In Java, file names (.java) should be the same as the class name.

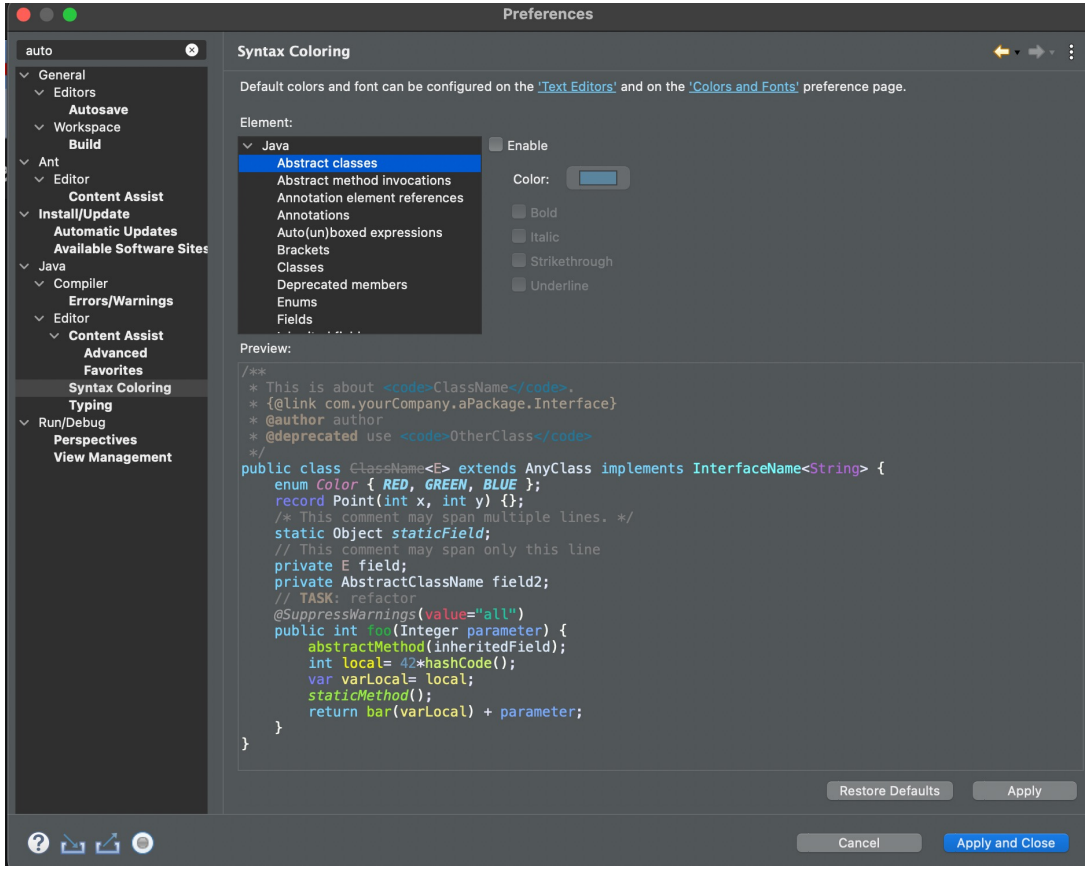



- cf) In Python, file names (.py) can be different from class names inside the file.

# 6. Eclipse Preferences

- Eclipse → Settings → General → Appearance

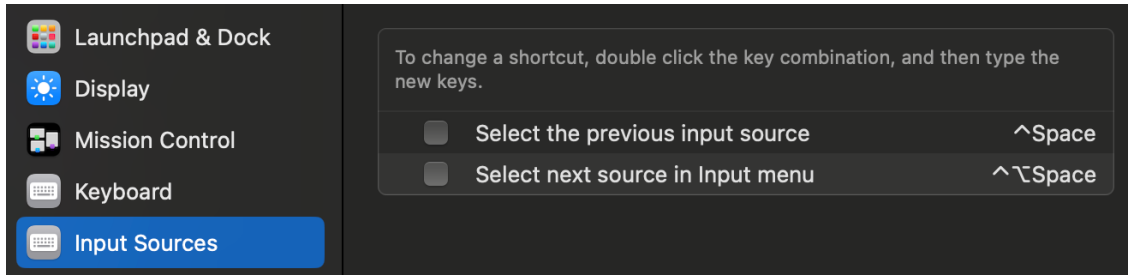


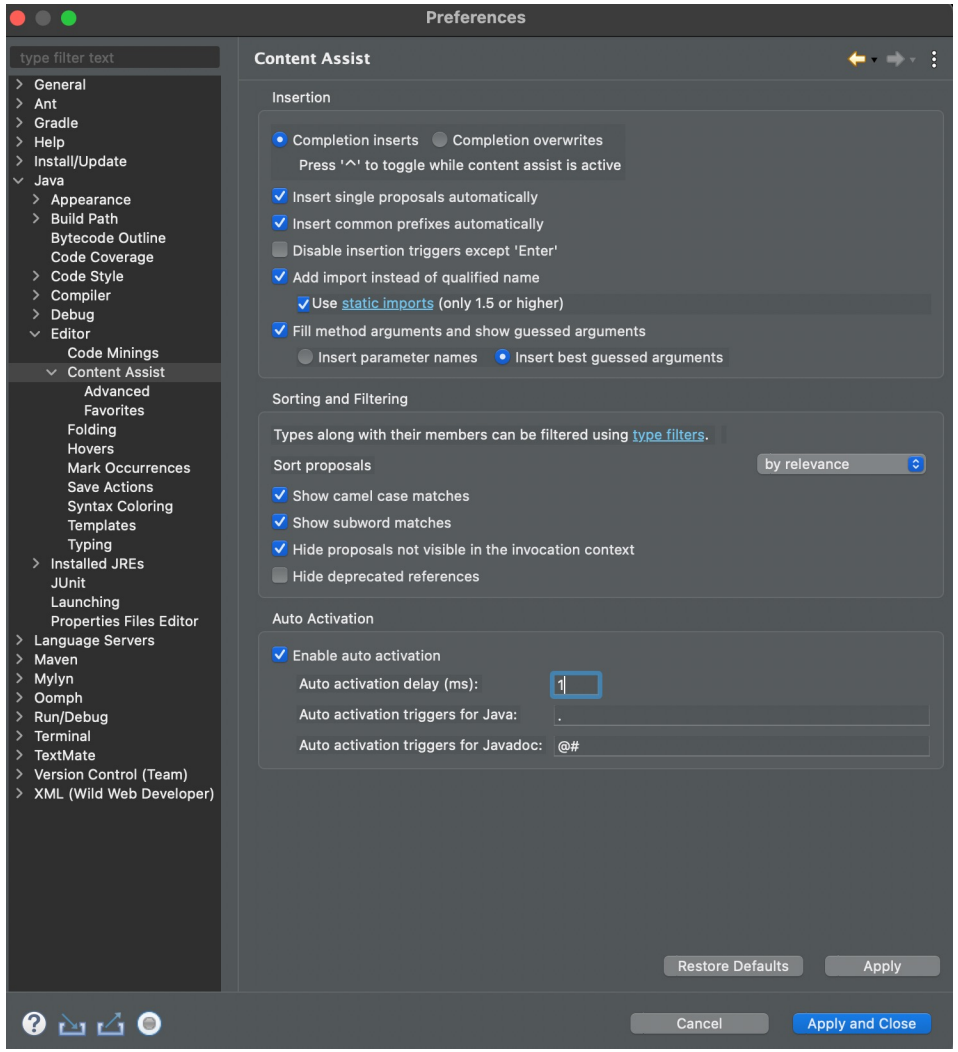
# 6. Eclipse Preferences (cont.)

- Preferences → Content Assist
  → Auto Activation

- Mac Keyboard Shortcuts → Input Sources
  → uncheck both items

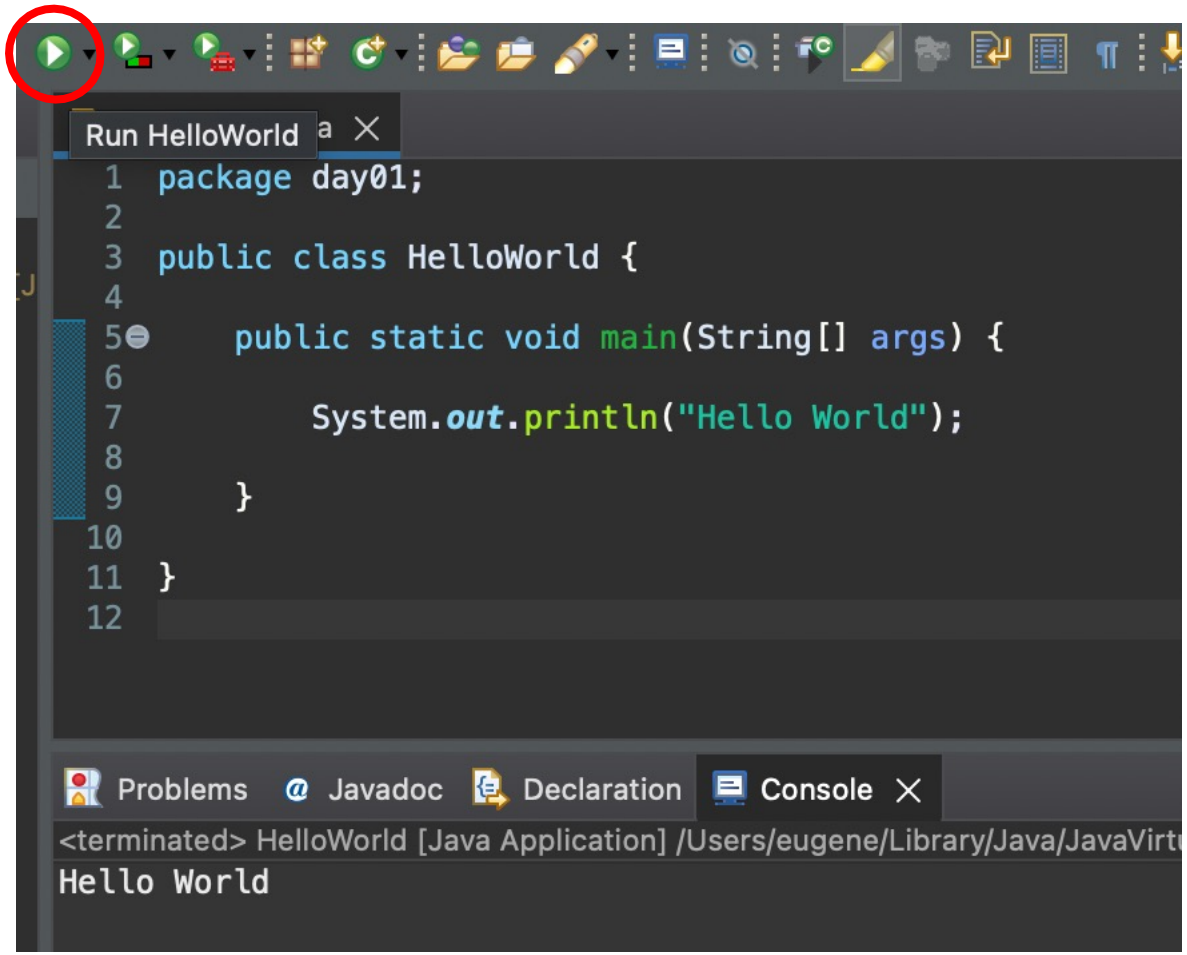



- *control + command + .* : auto-complete



# 7. Run HelloWorld.java

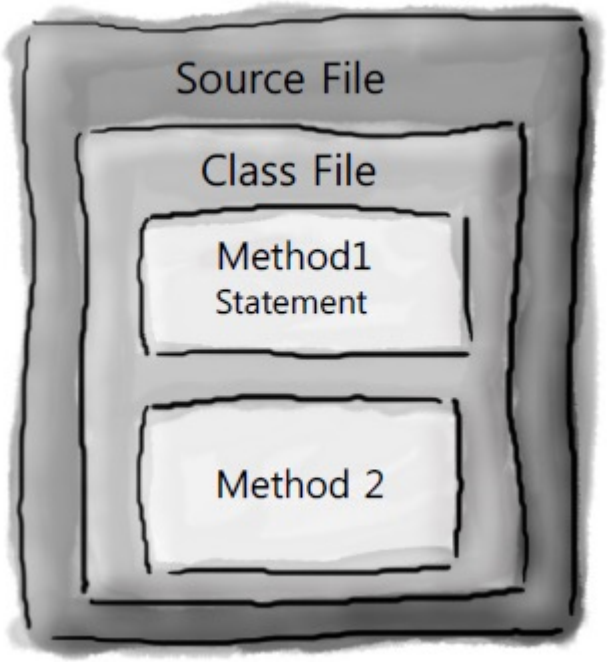
- `System.out.println("some string");` prints the string in the console



# JAVA BASICS

- Java Code Structure
- Anatomy of a Class
- Identifiers
- Keywords
- Naming Conventions
- Variables
- Methods
- Comments
- Print on Console

# Java Code Structure




**class**

```
public class Dog{
        …
}
```

**method**

```
public class Dog{
        void bark() {
        … }
}
```

**statement**

```
public class Dog{
      void bark() {
        System.out.println("…");
      }
}
```

# Anatomy of a Class

- When the JVM starts running, it first looks for the class you give at the command line.
- Next, it starts looking for the **main method**.

```
public static void main (String[ ] args) {
        //statement
}
```

- Next, it runs everything inside the { } of the main method.
- Every Java application has to have **at least one class, and one main method** (not per class, just per application!)
- The main method is where the program starts running.
- Every statement **must** end in a semicolon.

# Identifiers

- It must start with a letter, underscore U, or dollar sign ($).
- You can’t start a name with a number.
- Don’t start it with number.
- After the first character, you can use numbers as well.
- Blank space cannot be included.
- It can be anything you like, just so long as it isn't one of Java's reserved words.

# Keywords

- Keywords cannot be used as variable names.

| | | | | |
|---|---|---|---|---|
| abstract | continue | for | new | switch |
| assert*** | default | goto* | package | synchronized |
| boolean | do | if | private | this |
| break | double | implements | protected | throw |
| byte | else | import | public | throws |
| case | enum**** | instanceof | return | transient |
| catch | extends | int | short | try |
| char | final | interface | static | void |
| class | finally | long | strictfp** | volatile |
| const* | float | native | super | while |

# Naming Conventions

▽ Class / Interface Names

- 명사나 형용사를 서술적으로 연결하여 사용.
- 첫 글자는 대문자로 표기.
- 연결된 단어의 첫 글자도 대문자로 표기. (Camel-Case)
- 나머지 문자는 소문자로 표기.
- "$"(dollar sign)은 내부 클래스에서 특별한 의미가 있기 때문에 사용을 권장하지 않음.

▽ Variables

- 명사적 의미를 갖게 생성.
- 첫 글자는 소문자, 연결 단어의 첫 글자는 대문자로 표기합니다. (Camel-Case)
- 나머지 문자는 소문자로 표기.
- 일반적으로 변수 이름에서는 "_"(underscore character) 를 사용하지 않음.

# Naming Conventions

▽ Methods

- 동사적 의미를 갖게 함.
- 첫 글자는 소문자, 연결 단어의 첫 글자는 대문자로 표기. (Camel-Case)
- 나머지 문자는 소문자로 표기.
- 메서드 이름 뒤에는 한 쌍의 괄호 "( )" 가 뒤따름. ⇒ 인자 표시 (parameters)
- 일반적으로 "_"(underscore character)를 사용하지 않음.

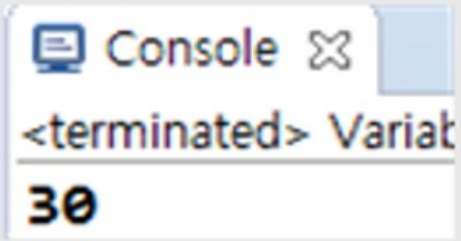
# Variables

*type* variableName;

- type: assigns the data type of the variable.
- variableName: follows the Java naming convention, camel-case.

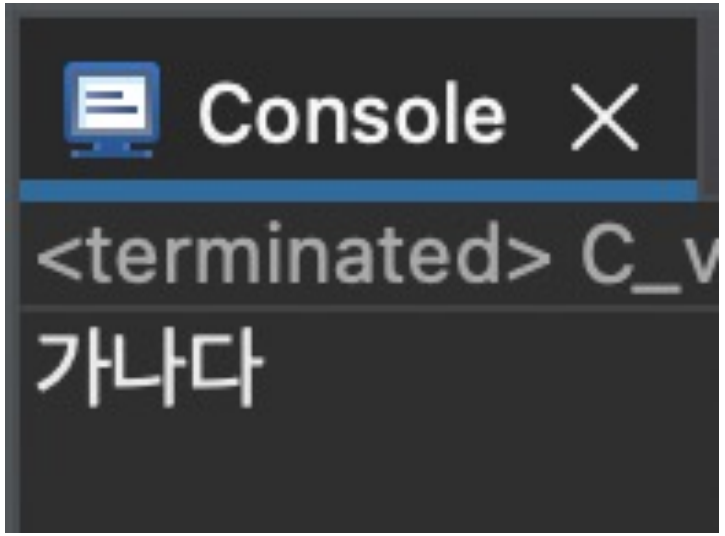
VariableExample.java

```
1:  public class VariableExample {
2:      public static void main(String[] args) {
3:          int numberOne = 10;
4:          int numberTwo = 20;
5:          int result = numberOne + numberTwo;
6:          System.out.println(result);
7:      }
8:  }
```



# Variables – Example

```
int a;
double d;
char c;
String s;

d = 3.14;
c = '가';
s = "가나다";

float f = 10.0f;

int b = 3;
String x = "abc";

System.out.println(s);
```



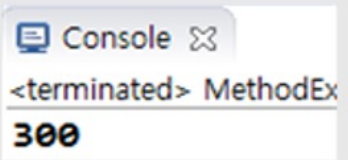
# Methods

```
returnType methodName (parameterList) {
        …
}
```

- returnType: the variable type of the return value.
  (if no data is returned after the method is ran, use `void`)
- parameterList: used to get the actual values of the parameters.
  - ✓ Each parameter variable must include its data type)

MethodExample.java

```
1:  public class MethodExample {
2:      public static void main(String[] args) {
3:          int result = add(100, 200);
4:          System.out.println(result);
5:      }
6:
7:      static int add(int num1, int num2) {
8:          return num1 + num2;
9:      }
10: }
```



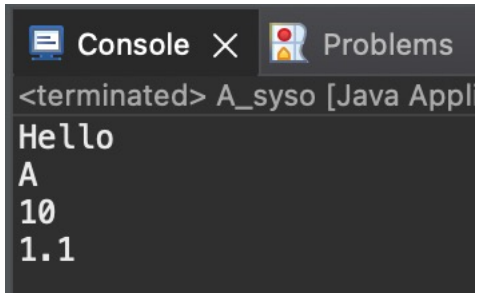
# Comments

- // ; 한 줄 주석 : ctrl + /
- /* ; 문단 주석: ctrl + shift + / */
- 주석 해제: ctrl + shift + ₩

# Print on Console

- `System.out.print();`
  - 콘솔 창에 소괄호 내 데이터 출력
- `System.out.println();`
  - 콘솔 창에 소괄호 내 데이터를 출력
  - 자동으로 줄 바꿈

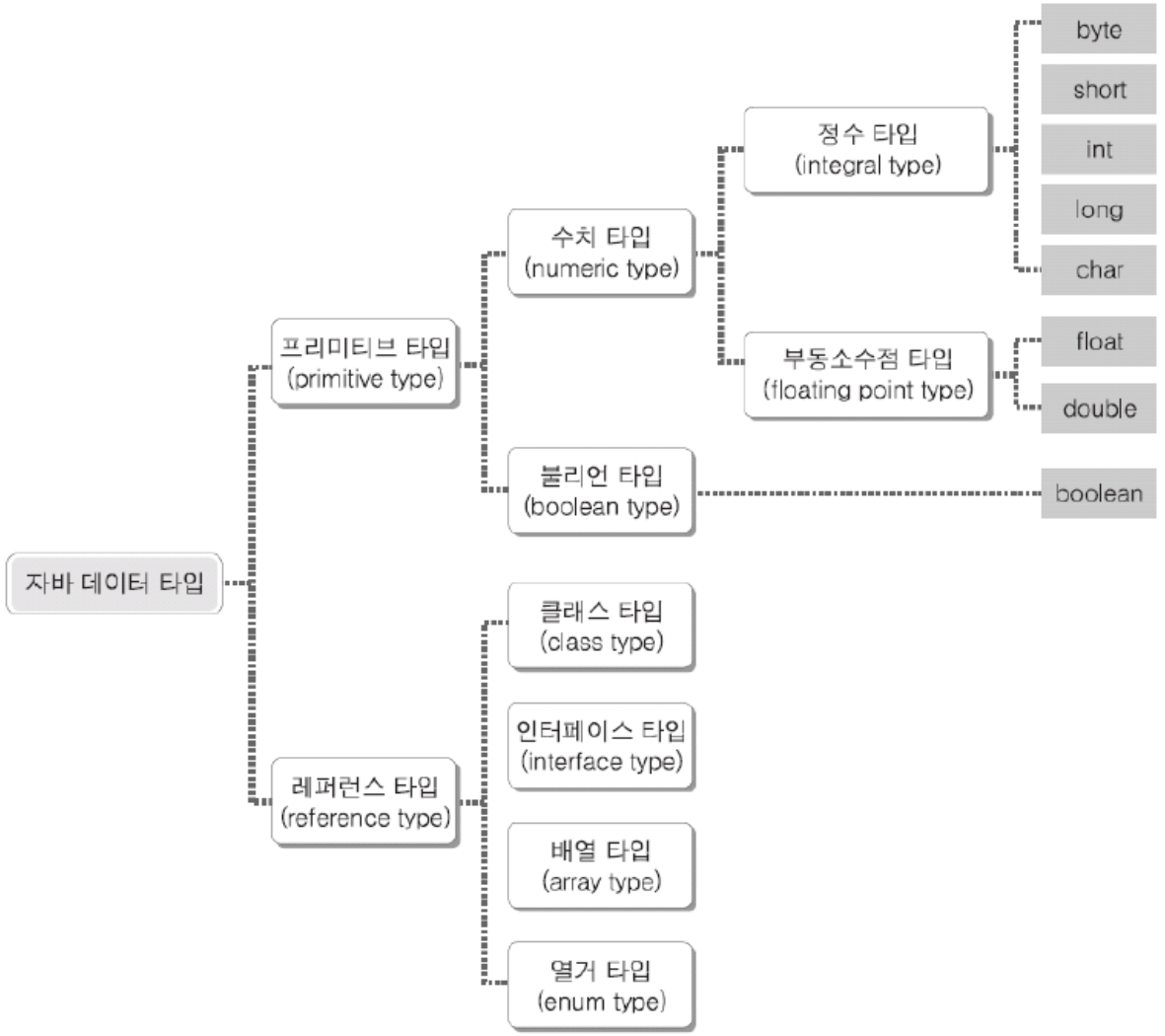
```java
package datatype;

public class A_syso {

    public static void main(String[] args) {

        System.out.println("Hello");
        System.out.println('A');
        System.out.println(10);
        System.out.println(1.1);

    }
}
```



# DATA TYPES

- Data Types
- String
- Operators
- Input
- Array

# Data Types


자바 데이터 타입
프리미티브 타입
(primitive type)
수치 타입
(numeric type)
정수 타입
(integral type)
byte
short
int
long
char
부동소수점 타입
(floating point type)
float
double
불리언 타입
(boolean type)
boolean
레퍼런스 타입
(reference type)
클래스 타입
(class type)
인터페이스 타입
(interface type)
배열 타입
(array type)
열거 타입
(enum type)

# Data Types (cont.)

- Primitive Data Types

| Type | Data Type | Memory | Default Value |
|---|---|---|---|
| Integral type | byte<br>(-128 ~ 127) | 1 byte | 0 |
| | short<br>(-32,768 ~ 32,767) | 2 bytes | 0 |
| | int<br>(-21억 ~ 21억) | 4 bytes | 0 |
| | long<br>(int 범위 벗어나는 더 큰 수 포함) | 8 bytes | 0L |
| Floating Point type | float<br>(±소수점 아래 7자리까지) (오차 범위 커서 잘 안 씀) | 4 bytes | 0.0F |
| | double<br>(±소수점 아래 15자리까지) | 8 bytes | 0.0D |
| Boolean type | boolean | - | false |
| Character type | char<br>(ASCII 코드 / UNICODE) ( ' ' 사용) | 2 bytes | '₩u0000' |

# String

- use class String to declare a variable.
- Default value of String is `null`.

```
String s1 = "Hello";          // declare a variable and assign a String
String s2;                    // default value null, length unknown
String s3 = "";               // null string, length=0
String s4 = " ";              // blank string, length != 0
```

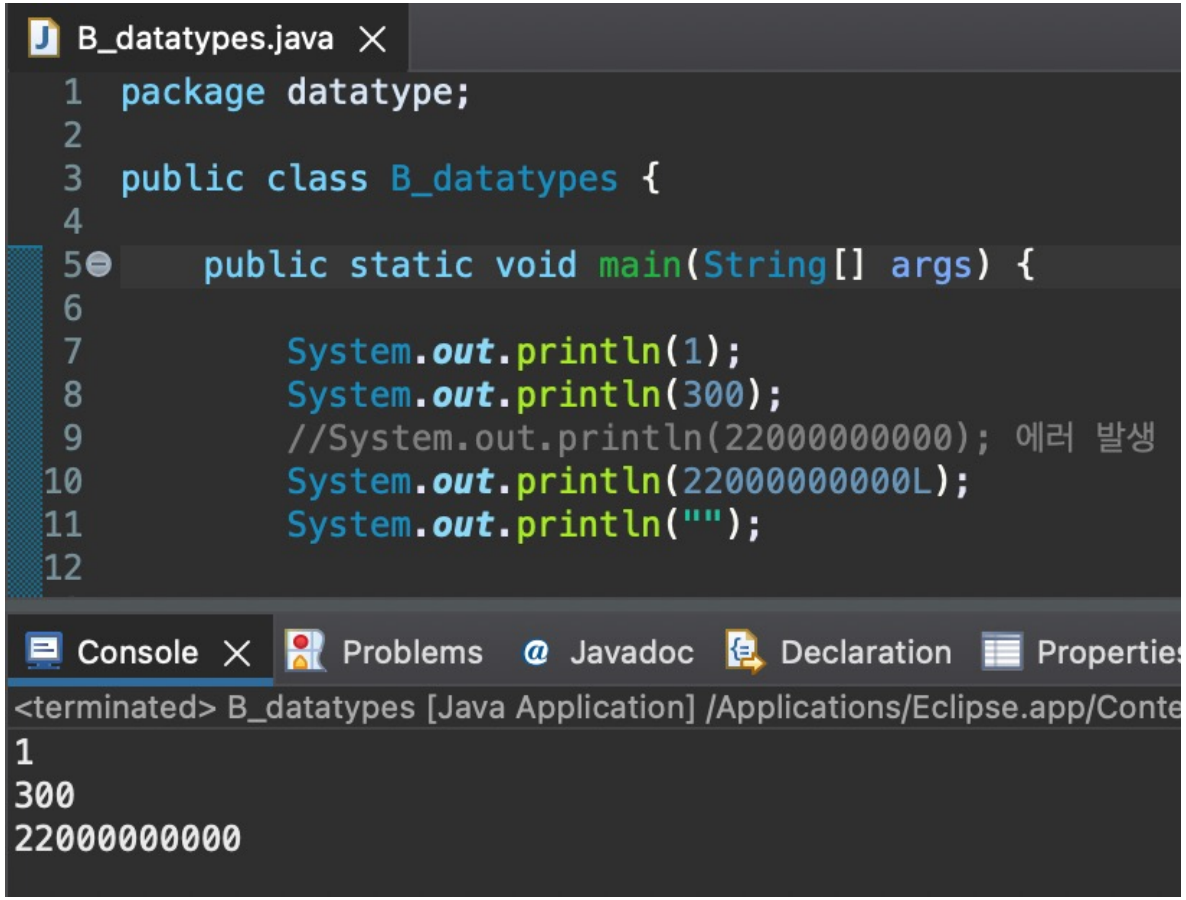
# String (cont.)

- When operator '+' is used with a String and other type, it is converted to String type. ⇒ returning a String.

```
100+200                 // 300
100+"200"               // "100200"
10+20+"Hello"           // "30Hello"
"Hello"+10+20           // "Hello1020"
```

# Data Types – Example

1. Integral type
   1) int
      - 한 정수당 4 byte
      - -21억 ~ 21억
   2) long
      - 한 정수당 8 byte
      - int형 범위 벗어나는 정수는 long 형으로 지정!



# Data Types – Example (cont.)

2. Floating point type

1) double

- 한 실수당 8 byte
- 소숫점 15자리까지 표현

2) float

- 한 실수 당 4 byte
- 소숫점 7자리까지 표현
- 오차 범위가 double보다 커서 잘 안 씀

```
System.out.println(1.0);
System.out.println(3.14342423432);
System.out.println(3.14d); // double
System.out.println(3.14f); // float
System.out.println("");
```

```
1.0
3.14342423432
3.14
3.14
```

# Data Types – Example (cont.)

3. Boolean type
   - String “true” != Boolean true
   - String “false” != Boolean false

```
System.out.println(true);
System.out.println(false);

System.out.println("true");
System.out.println("false");
```

```
true
false
true
false
```

# Data Types – Example (cont.)

4. Character type
   - 문자 1개, ' ' 사용
   - 한 문자당 2 byte
   - ASCII 코드 (1 byte):
     알파벳 대소문자, 0~9, 특수 문자
   - UNICODE (2 byte): 모든 국가의
     문자들, 기타 특수 문자

```
System.out.println('A');
System.out.println('a');
System.out.println('가');
System.out.println('1');
System.out.println((int)'A'); //ASCII 코드
System.out.println((int)'1'); //ASCII 코드
//System.out.println('가나다'); 에러
System.out.println("");
```

```
A
a
가
1
65
49

```

# Data Types – Example (cont.)

5. String type
   - 문자 여러 개, "" 사용
   - 자료형처럼 사용되는 특별 케이스

```
System.out.println("ABCDE");
System.out.println("안녕");
System.out.println("1");
System.out.println("");
```

```
ABCDE
안녕
1

```

# OPERATORS

# Operators

| Type | Operator | Function |
|---|---|---|
| 산술 연산자 | + - * / % | 사칙 연산 및 나머지 계산 |
| 단순 대입 | = | 우변의 값을 좌변의 변수에 대입 |
| 증가/감소 | ++ -- | 변수 값을 1만큼 증가/감소 |
| 수치 크기 비교 | < > <= >= | 수치 크기 비교 |
| 동등 연산자 | == != | 데이터 동일한지 비교 |
| 조건 연산자 | && \|\| | 논리적 and / or 연산 |
| 캐스트 연산자 | (*typeName*) | 타입 강제 변환 |

# Operators – Example

1. 산술 연산자 : +, -, *, /, %
   - / : 몫 알려 줌
     - 두 수 중에 적어도 1개가 실수여야 소숫점까지 나옴
   - % : 나머지 알려 줌
   - - 실수끼리도 가능
   - + : 문자열 덧셈 (결합)

```
System.out.println(5/2);
System.out.println(5.0/2);

System.out.println(10%3.0);
```

```
2
2.5
1.0
```

```
String s1 = "ABC";
String s2 = "DEF";
String s3 = s1 + s2;
String s4 = s1 + 10;

System.out.println(s1 + s2);
System.out.println(s3);
System.out.println(s4);
```

```
ABCDEF
ABCDEF
ABC10
```

# Operators – Example (cont.)

2. 대입 연산자 : =

- = : 왼쪽(공간)에 오른쪽(값)을 저장
- ex) a = 10; a = b;
- a = b 와 b = a는 완전히 다른 의미

```
a = 10;
b = 20;

System.out.println(a + b);
System.out.println("a + b 했을 때의 a : "  + a);
System.out.println(a += b);
System.out.println("a += b 했을 때의 a : "  + a);
```

❖ 복합 대입 연산자

- 연산과 저장을 동시에 수행, 변수의 값 바꿈!
- += , -= , *= , /= , %=

```
30
a + b 했을 때의 a : 10
30
a += b 했을 때의 a : 30
```

# Operators – Example (cont.)

3. 증감 연산자: ++, --

- a++, ++a : a에 +1한 후, 다시 a에 저장
- a--, --a : a에 -1한 후, 다시 a에 저장

```
int a = 10;
a++;
System.out.println(a);
++a;
System.out.println(a);
a--;
System.out.println(a);
--a;
System.out.println(a);
```

```
11
12
11
10
```

❖ 전치 및 후치 연산자

- 전치 연산 ++a : 자신 먼저 증감시킨 뒤, 나머지 연산 수행
- 후치 연산 a++: 나머지 연산 먼저 수행 뒤, 마지막에 자신을 증감시킴*

```
a = 10;
int b;
b = ++a;
System.out.println(b);
```

```
11
```

```
a = 10;
b = a++;
System.out.println(b);
System.out.println(a);
```

```
10
11
```

# Operators – Example (cont.)

❖ 전치 및 후치 연산자

```
int c = 100;
System.out.println("현재 C의 값 : " + c++ );  // print 먼저 수행 후, c +1 더해 줌
System.out.println(c);
System.out.println("현재 C의 값 : " + ++c );   // c +1 먼저 더한 후, print 실행
```

```
현재 C의 값 : 100
101
현재 C의 값 : 102
```

# Operators – Example (cont.)

4. 관계 연산자: < , <= , > , >= , == , !=
   - 결과는 boolean type (true / false)

```
boolean a;
a = (10==20);
System.out.println(a);
```

```
false
```

```
int b = 10;
a = (10 == b++); //먼저 관계 비교 후, b +1 시킴
System.out.println(a); //따라서 a = true
System.out.println(b);
```

```
true
11
```

# Operators – Example (cont.)

5. 논리 연산자 : &&  ,  ||  ,  !

   • 결과는 boolean type (true / false)

```
boolean a;

a = 10 > 20 && 15 < -5;
System.out.println(a);
```

```
false
```

   1) && : 논리곱연산자 (논리 AND 연산자)

      - 양 수식이 모두 참이어야 true
      - 하나라도 거짓이면 false
      - 앞 수식이 '거짓'이면 다음 수식은 무시함

```
long l = 1;
a = l > 10 && ++l == 2;
System.out.println(l);
System.out.println(a);
```

```
1
false
```

   2) || : 논리합연산자 (논리 OR 연산자)

      - 양 수식 중 하나라도 참이면 true
      - 앞 수식이 '참'이면 다음 수식은 무시함

```
a = 10 < 20 || 15 < -5;
System.out.println(a);
```

```
true
```

   3) ! : 논리부정연산자 ( 논리 NOT 연산자)

      - 수식이 참이면 false를, 거짓이면 true 반환

```
boolean b = !(10 == 20);
System.out.println(b);
```

```
true
```

# Operators – Example (cont.)

6. 조건 연산자
   - 조건식 ? 참일 경우 선택 값 : 거짓일 경우 선택

```
int a = 10;
int b = 20;

int max;

max = a > b ? a : b ;
System.out.println(max);
```

```
20
```

- “a가 b보다 크다”가 참이라면 a가 max값, 거짓이라면 b가 max값

# Operators – Example (cont.)

6. 조건 연산자 (실습)

   - Q) age 값을 임의로 저장한 후, 19 이하는 미성년자 / 이상은 성인을 String 타입의 변수에 저장하고 출력하세요.

# Operators – Example (cont.)

7. 캐스트 연산자

- Type Casting: 데이터의 자료형을 바꾸는 것
- 주용도: int형 데이터를 실수(float/double)형으로 바꿀 때
- 강제 형변환: 개발자가 원하는 대로 (강제로) 형변환 => 캐스트 연산자 사용
  - ex) `(castType)data`
- 자동 형변환 : 컴퓨터가 컴파일을 위하여 자동으로 (어쩔 수 없는 상황에서) 형변환

```
double d;
d = 3; //자동 형변환: 3을 3.0으로 형변환 뒤 d에 저장
System.out.println(d);
```

```
3.0
```

```
int i = (int)3.14; //3.14를 int형으로 강제 casting하고, i에 저장
System.out.println(i);
```

```
3
```

```
System.out.println((int)'A'); //A의 정수 값 (ASCII 코드)
```

```
65
```

# Operators – Example (cont.)

7. 캐스트 연산자 (실습)
   - Q) 세 과목의 점수를 임의로 지정하고, 조건 연산자를 사용하여 평균이 60.2 이상이면 합격! 그렇지 않으면 불합격!을 출력하세요.

# Operators – Example (cont.)

- `Math.random()`: 랜덤 수 (난수) 생성
  - 0.0 ~ 0.99999999 범위 내의 실수가 랜덤하게 나옴

```
System.out.println(Math.random());
System.out.println(Math.random()*10);

/*0~9까지 정수를 랜덤하게 출력*/
System.out.println((int)(Math.random()*10));

/*1~10까지 수를 랜덤하게 출력*/
System.out.println((int)(Math.random()*10) + 1);

/*1~6 까지 수를 랜덤하게 출력*/
System.out.println((int)(Math.random()*6) + 1);
```

```
0.8460873229281287
3.7784676857826502
6
8
5
```

```
0.5520373110072481
5.387841329358007
0
7
4
```

# Operators – Example (cont.)

- `Math.random()` (실습)
  - Q) 2~9단 구구단을 랜덤하게 출력하세요.

```
2 x 8 = 16
```

```
3 x 9 = 27
```

```
4 x 6 = 24
```

# INPUT

# Input

- Can use class `Scanner` to get user's input.

- java.util.Scanner
  - inside `util` package
  - text scanner
  - parse primitive types and string
  - needs to be imported at the beginning of the code file

# Input – Scanner Example

- `java.util.Scanner` is imported.
- Scanner object is made (`sc`).
- sc has `next()` method for taking a String for user input.
- sc has `nextInt()` method for taking a int for user input.

```java
package input;
import java.util.Scanner;

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

        Scanner sc = new Scanner(System.in);

        System.out.print("숫자를 입력하세요: ");
        int a = sc.nextInt();
        System.out.println(a);

        System.out.print("문자를 입력하세요: ");
        String b = sc.next();
        System.out.println(b);

    }
}
```

```
숫자를 입력하세요: 12
12
문자를 입력하세요: hello
hello
```

# Input – Scanner Example

- Q) 클래스에서 나이, 이름, 성별(f/m) 을 사용자에게 입력받아서 출력하는 Person 클래스를 만들어 보세요.

```java
package input;
import java.util.Scanner;

public class Person {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        int age;
        String name;
        String gender;
```

```
나이를 입력하세요: 29
이름을 입력하세요: Choco
성별을 입력하세요: F
================
나이: 29
이름: Choco
성별: F
```

# Input (cont.)

- Can use class `JOptionPane` to get user's input.

- javax.swing.Joption
  - inside `javax` package
  - pop up a standard dialog box that prompts user for a value or informs them of something

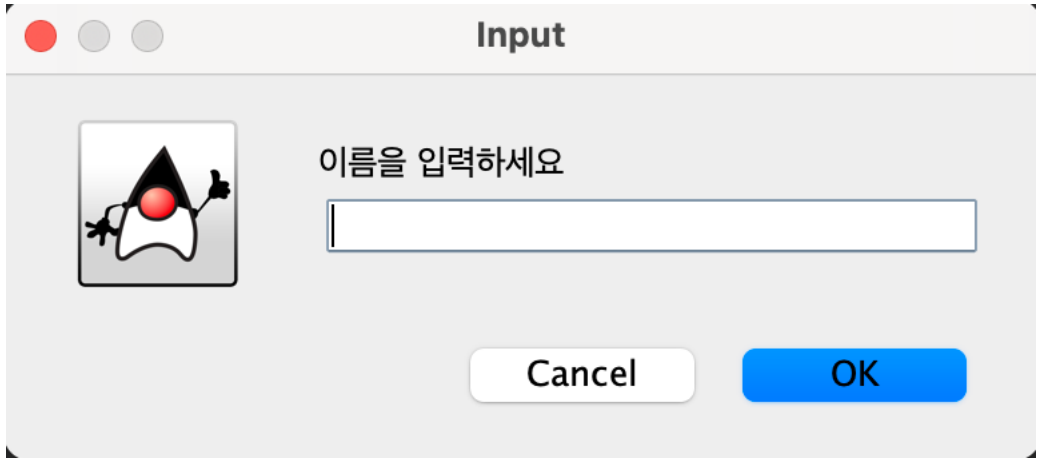
| | |
|---|---|
| `showConfirmDialog` | Asks a confirming question, like yes/no/cancel. |
| `showInputDialog` | Prompt for some input. |
| `showMessageDialog` | Tell the user about something that has happened. |
| `showOptionDialog` | The Grand Unification of the above three. |

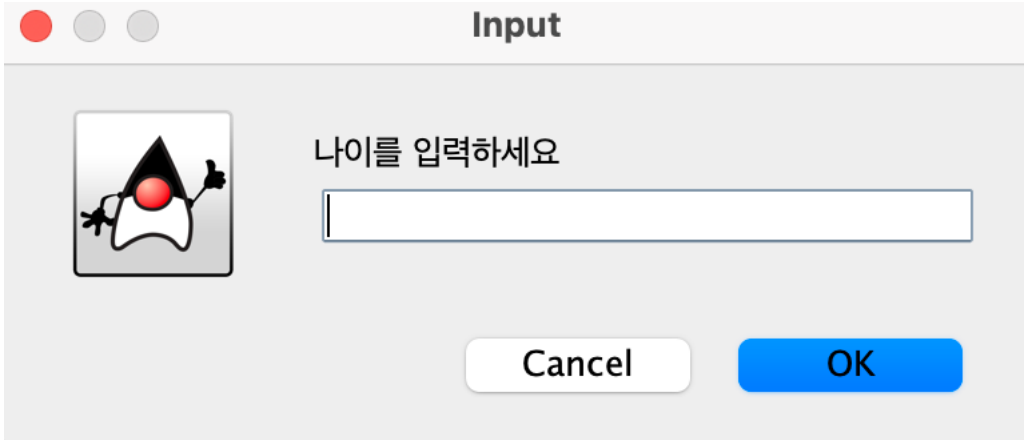
# Input – JOptionPane Example

- Class `javax.swing.JOptionPane` needs to be imported.
  - `showInputDialog("…")`
  - `showMessageDialog(Component parentComponent,`
    `Object message, String title, int messageType)`
  - `showConfirmDialog(Component parentComponent,`
    `Object message)`

# Input – JOptionPane Example

```
String name = JOptionPane.showInputDialog("이름을 입력하세요");
String tmp = JOptionPane.showInputDialog("나이를 입력하세요");
int age = Integer.parseInt(tmp);
```



# Input – JOptionPane Example

```
System.out.println(name);
System.out.println(age);
JOptionPane.showMessageDialog(null, "이름:" + name + " 나이:" + age, "DOG INFO", 2);
JOptionPane.showConfirmDialog(null, "강아지 정보 잘 보셨나요?");
```

```
Choco
3
```