First Java Program

1. Java Overview

We have already learned “Processing” and “Python.”

Processing is an open-source graphical library and integrated development environment (IDE). The programming part which performs detailed setting is excluded for the content creation work by the artist. It is suitable for beginners to learn programming.

Python is a general purpose programming language. The code is designed to be simple and easy to handle. There are features such that various programs are easy to understand and can be written with a small number of lines of code. Python features a dynamic type system and automatic memory management. Python has libraries of mathematics and statistical analysis, and it is suitable for use in academic fields such as computer science, data analysis, bioinformatics.

In Processing and Python, it is difficult to understand the operation in the computer because it is processing well in the language system. That is, it may be difficult to decide a method corresponding to the generated error.

Java is a programming language and a computing platform for application development. It was first released by Sun Microsystem in 1995 and later acquired by Oracle Corporation. Java programming is statically typed means that one has to explicitly mention the data type of variable. So, if data type (int, float, double, char) does not mention then the error will occur in program. Therefore, Java is suitable for clearly creating programs. Java has high versatility and is often used when creating large systems.

To compare the three programming languages, let’s review how the program is executed.

(A) Processing

We use Processing Development Environment (PDE) to write Processing programs and execute the programs. Basically, program source which is a sequence of instructions is executed from top to bottom. However, using the functions titled “setup()” and “draw()”, we can create interactive programs. The “setup()” block runs once, and the draw() block runs repeatedly.

(B) Python

Python programs are also executed from top to bottom. However, when the program is used as a module (imported by another program), each instruction group must be defined in the form of a function. If it is a program with only function definition, there is no way to start it at the beginning. So, we use "if __name__ == '__main__':" to start the program.

(C) Java

In Java, it is always executed from the main method. A method is a function indicating an action. In principle, the instructions are executed from top to bottom in the method.

2. First Java Program

If the Java system or Eclipse is not available on your MacBook (or PC), please refer to Chapters 1 and 2 of "Python2" which is the material of "Introduction to Programming" to build a programming environment.

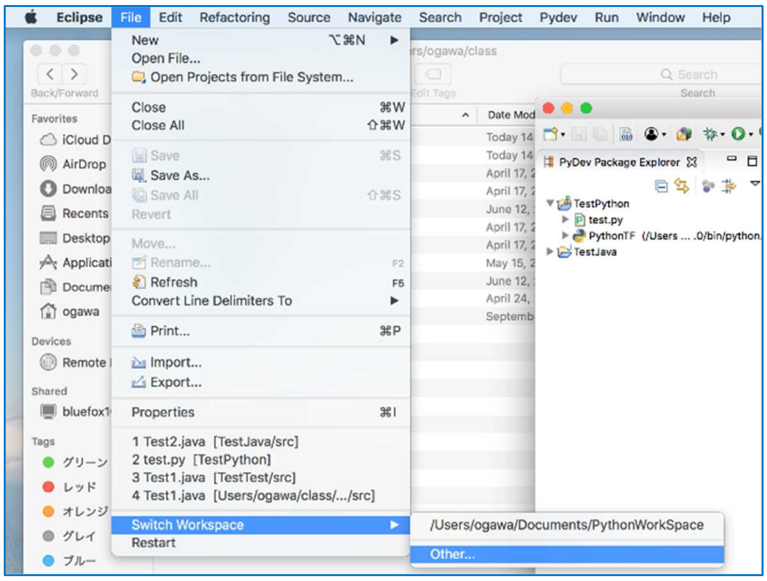
How to use eclipse to create a Java program will be explained step by step as follows:

Step 0. Changing Workspace (Option)

If you want to use a different workspace than the python program, you can change the workspace as follows:

(a) Prepare a new folder for workspace.
Example: /Users/ogawa/class/ProgrammingLanguage/workspace

(b) Click "File" on the menu bar, select "Switch Workspace", then click "Other..." as below.



Eclipse File Edit Refactoring Source Navigate Search Project Pydev Run Window Help
New
Open File...
Open Projects from File System...
Close
Close All
Save
Save As...
Save All
Revert
Move...
Rename...
Refresh
Convert Line Delimiters To
Print...
Import...
Export...
Properties
1 Test2.java [TestJava/src]
2 test.py [TestPython]
3 Test1.java [TestTest/src]
4 Test1.java [Users/ogawa/class/.../src]
Switch Workspace
Restart
/Users/ogawa/Documents/PythonWorkSpace
Other...
PyDev Package Explorer
TestPython
test.py
PythonTF
TestJava

(c) Click "Browse..." in Eclipse Launcher window. And select the workspace directory to use. Then click "launch" button in Eclipse Launcher window. Eclipse restarts and the specified workspace becomes available.

Step 1. Java Perspective

In order to create Java programs in eclipse, we need to select "Java Perspective." Click (Open Perspective) at the upper right of eclipse window, select Java (default) , then click the "Open" button.

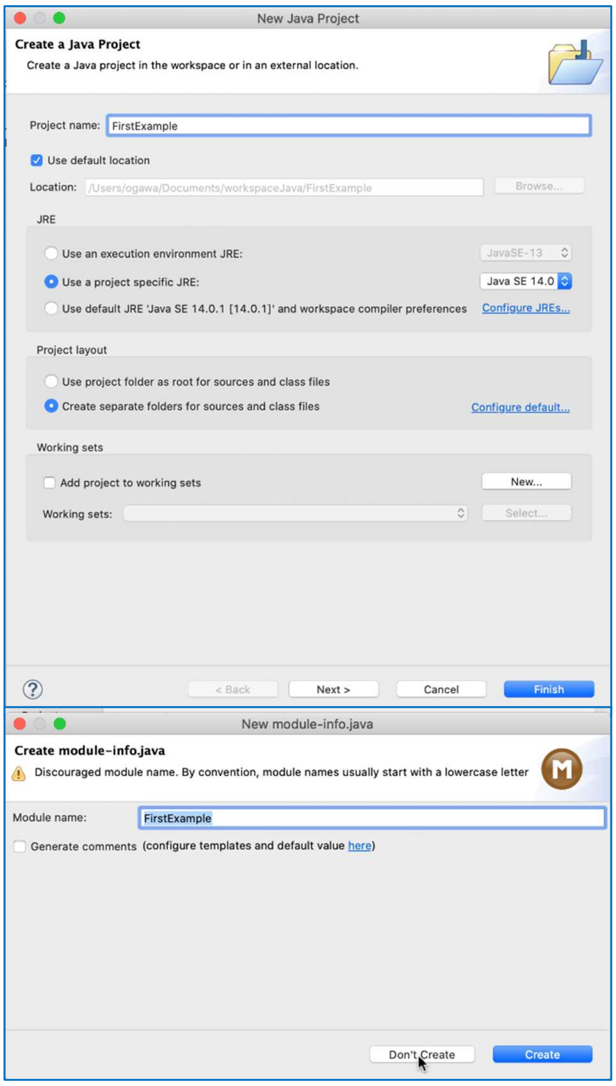
Mark appears at the upper right of eclipse window.

Step 2. Project

To create a new project, click "File" on the menu bar and select "new," then select "Java Project." Enter "Project name" and select the JRE to use.
Example: the project name is "FirstExample", JRE is "JavaSE-14.0"

When the "Finish" button is clicked, "New module-info.java" window will appear. Click the "Don't Create" button.

If you click the "Create" button, the file named "module-info.java" will be created in the project. In this case, the concept of "package" is needed to create programs within this project. If you have created it, delete it.


New Java Project

Create a Java Project
Create a Java project in the workspace or in an external location.

Project name: FirstExample
[✓] Use default location
Location: /Users/ogawa/Documents/workspaceJava/FirstExample

JRE
Use an execution environment JRE: JavaSE-13
[✓ Use a project specific JRE: Java SE 14.0]
Use default JRE 'Java SE 14.0.1 [14.0.1]' and workspace compiler preferences

Project layout
Use project folder as root for sources and class files
[✓ Create separate folders for sources and class files]

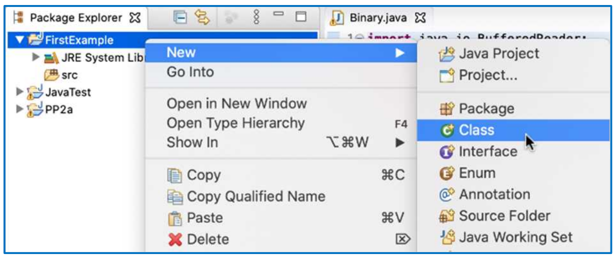
Working sets
Add project to working sets
Working sets:

New module-info.java

Create module-info.java
Discouraged module name. By convention, module names usually start with a lowercase letter
Module name: FirstExample
Generate comments (configure templates and default value here)

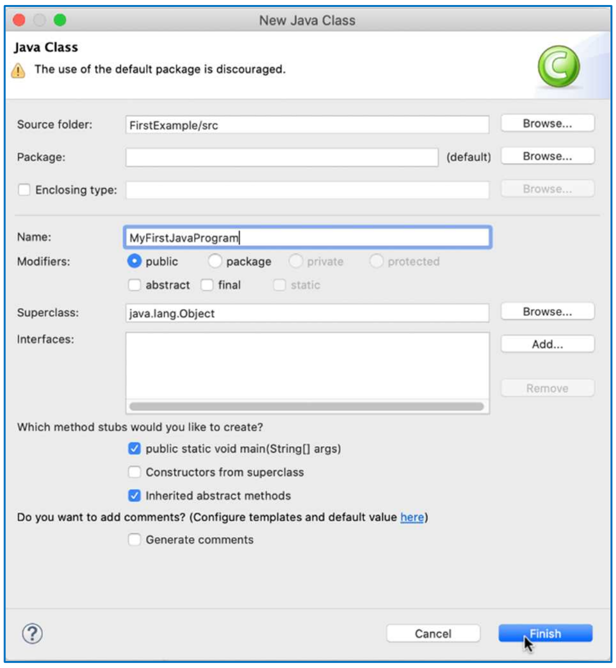
Step 3. Class

To create a new program, right click the Project name (or control + click, or click or tap with two fingers), and select "new," then select "class."



In the "New Java Class" window, enter program name to Name: text field. Please do not define "package" to simplify processing. Please check box if you want to use main method.

In the example on the right, package does not input anything, the program name is "MyFirstJavaProgram," and the main method is set up.


New Java Class

Java Class

The use of the default package is discouraged.

Source folder: FirstExample/src

Package: (default)

Enclosing type:

Name: MyFirstJavaProgram

Modifiers: [✓ public] package private protected

abstract final static

Superclass: java.lang.Object

Interfaces:

Which method stubs would you like to create?

[✓ public static void main(String[] args)]

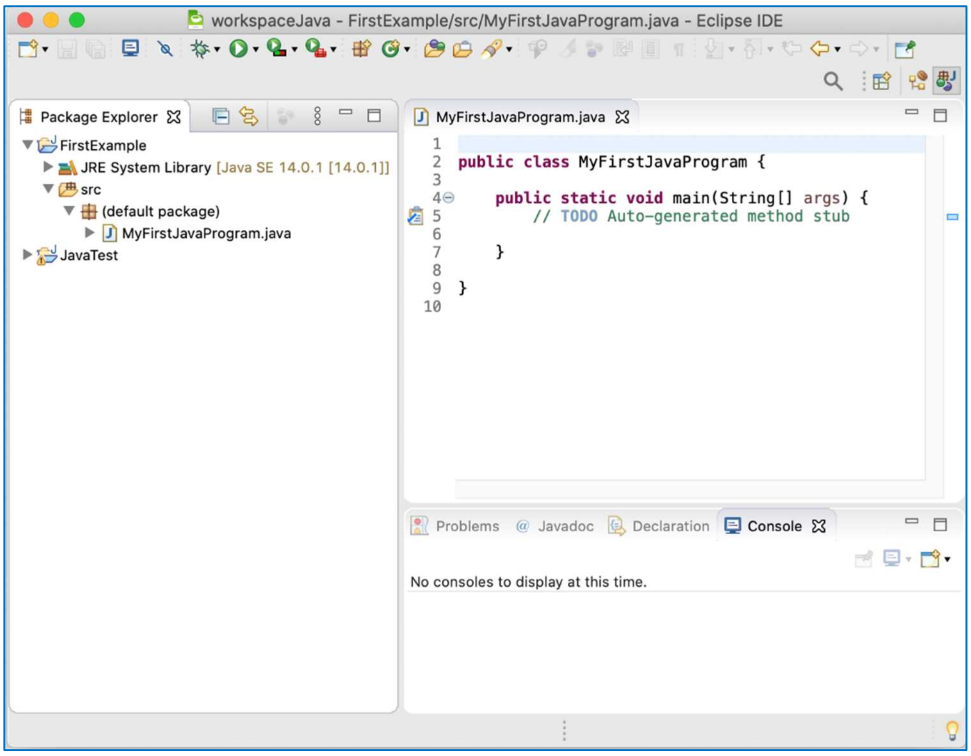
Constructors from superclass

[✓ Inherited abstract methods]

Do you want to add comments? (Configure templates and default value here)

Generate comments

When the "Finish" button is clicked, Eclipse changes to be able to create a program as shown in Figure 1.


workspaceJava - FirstExample/src/MyFirstJavaProgram.java - Eclipse IDE

```

public class MyFirstJavaProgram {

    public static void main(String[] args) {
        // TODO Auto-generated method stub

    }

}

```

No consoles to display at this time.

Figure 1. Initial state of programming in Eclipse.

Step 4.　Creating program

As shown in Figure 2, a program is filled in the editor window of MyFirstJavaProgram. There is the comment describing the purpose, creator, and date of the program in the first part. This program has a greeting method and a main method.

Step 5.　Execution of program

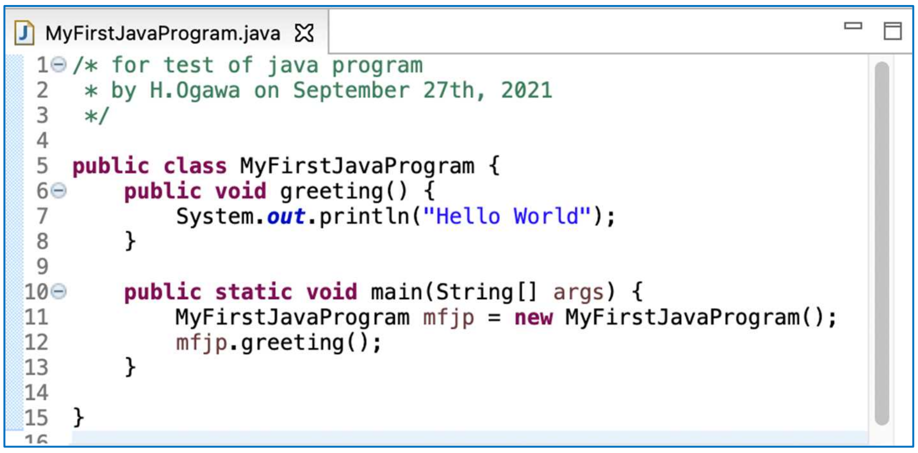
To run the program, click “RUN” on the menu bar, and select “RUN” from the pull-down menu. The result is shown in Figure 3.

In this example, we created a class. A class is an entity that determines how an object will behave and what the object will contain. In other words, it is a blueprint or prototype that defines the variables and the methods (functions).

In a Java system, only objects can be executed. However, the main method is a special method that works even if no object is created.

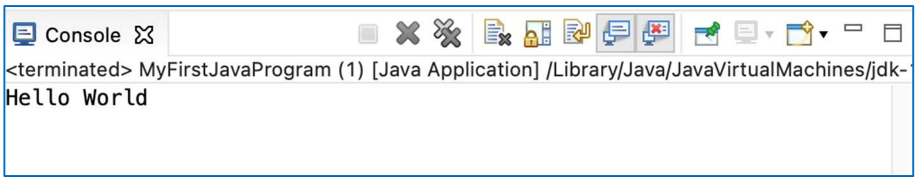
The object of “MyFirstJavaProgram” was created in the main method (line 11), and the greeting method of “MyFirstJavaProgram” was executed (line 12).

“System.out.println” is a Java statement that prints the argument, which is passed into the “System.out” (which is generally standard output). The standard output is displayed in the “Console” tab in Eclipse. In this example, “Hello World” is displayed



```
 1 /* for test of java program
 2  * by H.Ogawa on September 27th, 2021
 3  */
 4
 5 public class MyFirstJavaProgram {
 6     public void greeting() {
 7         System.out.println("Hello World");
 8     }
 9
10     public static void main(String[] args) {
11         MyFirstJavaProgram mfjp = new MyFirstJavaProgram();
12         mfjp.greeting();
13     }
14
15 }
```

Figure 2. Example of java program.



```
<terminated> MyFirstJavaProgram (1) [Java Application] /Library/Java/JavaVirtualMachines/jdk-
Hello World
```

Figure 3. The result of “MyFirstJavaProgram.java”.

3. Basic Syntax of Java

3. 1 Basic Points

- ✓ **Case Sensitivity:** Java is case sensitive, which means identifier **Hello** and **hello** would have different meaning in Java.
- ✓ **Class Names:** For all class names the first letter should be in Upper Case. If several words are used to form a name of the class, each inner word's first letter should be in Upper Case.
  Example: *class MyFirstJavaClass*
- ✓ **Method Names:** All method names should start with a Lower Case letter. If several words are used to form the name of the method, then each inner word's first letter should be in Upper Case.
  Example: *public void myMethodName()*
- ✓ **Program File Name:** Name of the program file should exactly match the class name. When saving the file, you should save it using the class name (Remember Java is case sensitive) and append '.java' to the end of the name (if the file name and the class name do not match, your program will not compile).

  Example: Assume 'MyFirstJavaProgram' is the class name. Then the file should be saved as *'MyFirstJavaProgram.java'*
- ✓ **public static void main(String args[]):** Java program processing starts from the main() method which is a mandatory part of every Java program.

3. 2 Java Identifiers

All Java components require names. Names used for classes, variables, and methods are called **identifiers**. In Java, there are several points to remember about identifiers. They are as follows:

- ✓ All identifiers should begin with a letter (A to Z or a to z), currency character ($) or an underscore (_).
- ✓ After the first character, identifiers can have any combination of characters.
- ✓ A key word cannot be used as an identifier.
- ✓ Most importantly, identifiers are case sensitive.
- ✓ Examples of legal identifiers: age, $salary, _value, __1_value.
- ✓ Examples of illegal identifiers: 123abc, -salary.

[Java Keywords]

The following list shows the reserved words in Java. These reserved words may not be used as constant or variable or any other identifier names.

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

3. 3 Comments in Java

The Java language supports three kinds of comments:

// text

The compiler ignores everything from // to the end of the line.

/* text */

The compiler ignores everything from /* to */.

/ documentation */**

This indicates a documentation comment. The JDK Javadoc tool uses this. For detail information of Javadoc, see "https://docs.oracle.com/javase/10/"

4. Variables and Data Types

A variable can be thought of as a container which holds value for us. Every variable is assigned a data type which designates the type and quantity of value it can hold.

4. 1 Variable Declaration and Initialization

We must declare all variables before they can be used. There was no need to declare variables in Python. However, in C language and Java, it is necessary to declare variables in advance in order to confirm the correctness of the program in the source code.

There are 2 steps to use a variable. Following is the basic form of a variable declaration:

```
data_type varibable;
```

Please refer 4.3 for *data_type*. Basically, *variables* should, in principle, have unique names. To declare more than one variable of the specified type, we can use a comma-separated list.

To initialize a variable, we must assign it a valid value. The basic form is as follows:

```
varibable = value;
```

We can combine variable declaration and initialization. Examples is shown below:

```
int a, b, c;              // Declares three integers a, b, and c.
int a = 10, b = 10;   // Example of declaration and initialization
byte d = 22;      // initialized a byte type variable d.
double pi = 3.14159;     // declares and assigns a value of PI.
```

4. 2 Types of variables

There are three types of variables in Java.

(A) Local Variables

Local variables are declared in methods, constructors, or blocks. Local variables are visible only within the declared method, constructor, or block. There is no default value for local variables, so local variables should be declared and an initial value should be assigned before the first use.

The arguments of method are also local variables.

(B) Instance Variables

Instance variables (which is called field) are declared in a class, but outside a method, constructor or any block. Instance variables are defined without the static keyword. They are called so because their values are instance (object) specific and are not stared among instances (objects).

The instance variables are visible for all methods, constructors and block in the class. Instance variables have default values. For numbers, the default value is 0, for Booleans it is false, and for object references it is null.

(C) Class/Static Variables

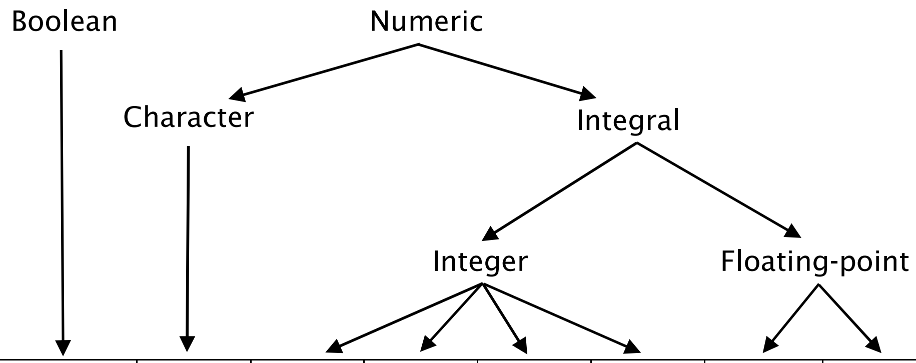
Class variables alse known as static variables are declared with the static keyword in a class, but outside a method, constructor or a block. There would only be one copy of each class variable per class, regardless of how many instances (objects) are created from it. Visibility is similar to instance variables. Default values are same as instance variables.

4. 3 Data Types in Java

Data types specify the different sizes and values that can be stored in the variable. There are two types of data types in Java.

- Primitive data (e.g., number, character)
- Object data (Non-primitive data: programmer created typed)

There are eight primitive datatypes supported by Java shown in Figure 3. Primitive datatypes are predefined by the language and named by a keyword.



	Boolean	char	byte	short	int	long	float	double
Default Value	false	null	0	0	0	0L	0.0f	0.0d
Default Side	1 bit	2 bytes	1 byte	2 bytes	4 bytes	8 bytes	4 bytes	8 bytes

Figure 3. Primitive Data Types

4. 4 Type Conversion and Type Casting

A variable of one type can receive the value of another type. There are 2caces.

Case 1) Variable of smaller capacity is assigned to another variable of bigger capacity.

Example:
```
double d;
int i = 10;
d = i;
```
This process is automatic, and non-explicit is known as **conversion**.

Case 2) Variable of larger capacity is assigned to another variable of smaller capacity.

Example:
```
double d = 10;
int i;
i = (int)d;
```
In such cases, we have to explicitly specify the **type cast operator**.

5. Class and Object

A class is an entity that determines how an object will behave and what the object will contain. In other words, it is a blueprint or a set of instruction to build a specific type of object.

The form is class as follows:

```
class class_name {
      field
      method
}
```

Here, *class_name* is a name of class. The class_name as shown in Figure 1 is "MyFirstJavaProgram." The *field* describes the information the class has. Mainly instance variables and class variables are used. The *method* describes the functions of the class. Methods to realize the functions are written.

An object is nothing but a self-contained component which consists of methods and properties to make particular types of data useful. Software objects are often used to model real-world objects we find in everyday life.

5. 1 Software Object

If we consider the real-world, we can find many objects around us, cars, dogs, humans, etc. All these objects have a state and a behavior.

If we consider a dog, then its state is name, breed, color, and the behavior is barking, wagging the tail, running. If we compare the software object with a real-world object, they have very similar characteristics.

Software objects also have a state and a behavior. A software object's state is stored in *field* and behavior is shown in *method*.

5. 2 Classes in Java

Figure 5 shows illustrations of three different breeds of dogs. Some of the differences we might have listed out maybe breed, age, size, color, etc. These differences are also some common characteristics shared by these dogs. These characteristics (breed, age, size, color) can form data members for your object. Next, we can list out the common behaviors of these dogs like sleep, sit, eat, etc. So, these will be the actions of our software objects.

So far we have defined following things:

- ✓ Class: Dog
- ✓ Data (field): breed, size, age and color.
- ✓ Method: eat, sleep, sit and run.

Figure 5. Three different breeds of dogs

We will get different dog object as follows:

Breed = Akita Inu

Size = Midium

Age = 3 years

Color = Brown

Breed = Dobermann

Size = Large

Age = 5 years

Color = Black

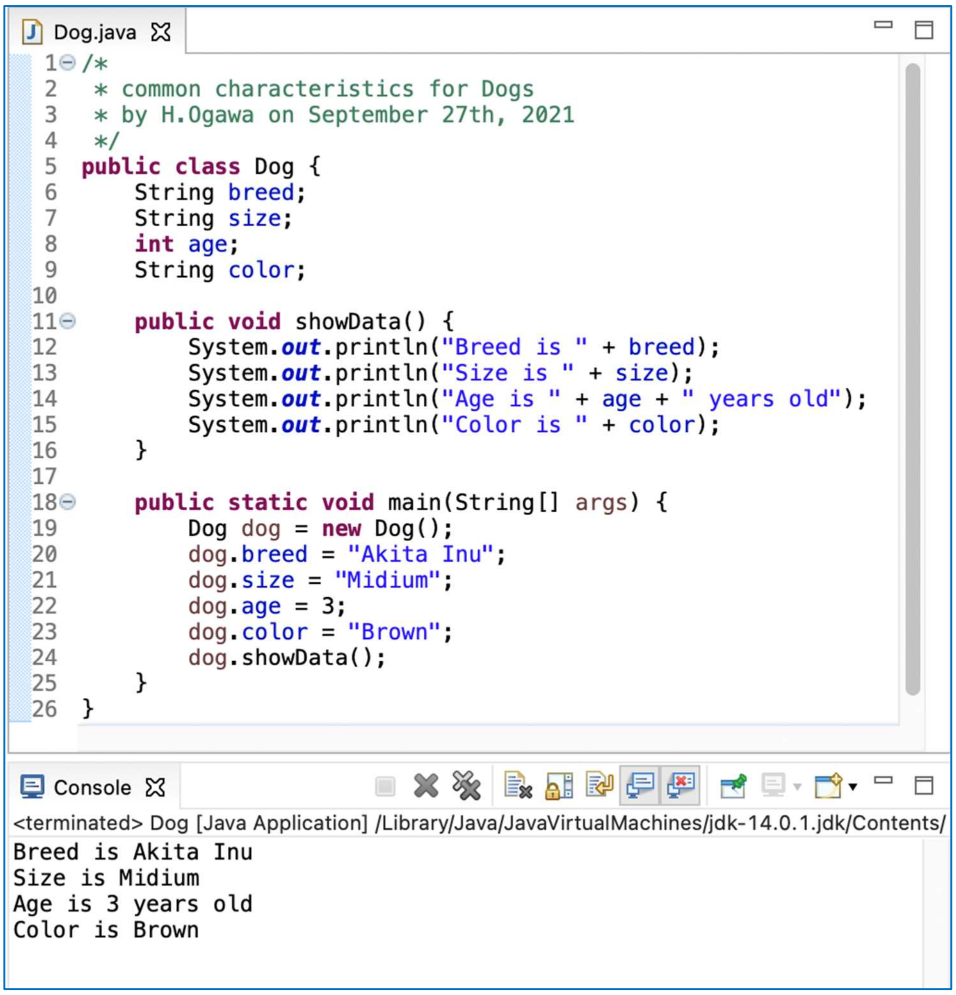
Breed = Maltese

Size = Small

Age = 2 years

Color = White

Example program for Dog is shown as below. The common characteristics are displayed on the console. This program uses the data from Akita Inu.



Dog.java

```
 1 /*
 2  * common characteristics for Dogs
 3  * by H.Ogawa on September 27th, 2021
 4  */
 5 public class Dog {
 6     String breed;
 7     String size;
 8     int age;
 9     String color;
10
11     public void showData() {
12         System.out.println("Breed is " + breed);
13         System.out.println("Size is " + size);
14         System.out.println("Age is " + age + " years old");
15         System.out.println("Color is " + color);
16     }
17
18     public static void main(String[] args) {
19         Dog dog = new Dog();
20         dog.breed = "Akita Inu";
21         dog.size = "Midium";
22         dog.age = 3;
23         dog.color = "Brown";
24         dog.showData();
25     }
26 }
```

Console

```
<terminated> Dog [Java Application] /Library/Java/JavaVirtualMachines/jdk-14.0.1.jdk/Contents/
Breed is Akita Inu
Size is Midium
Age is 3 years old
Color is Brown
```

When we run the class, the main method is executed first. In Dog class, an instance (object) of Dog class is created and assigned to the variable dog (line 19).

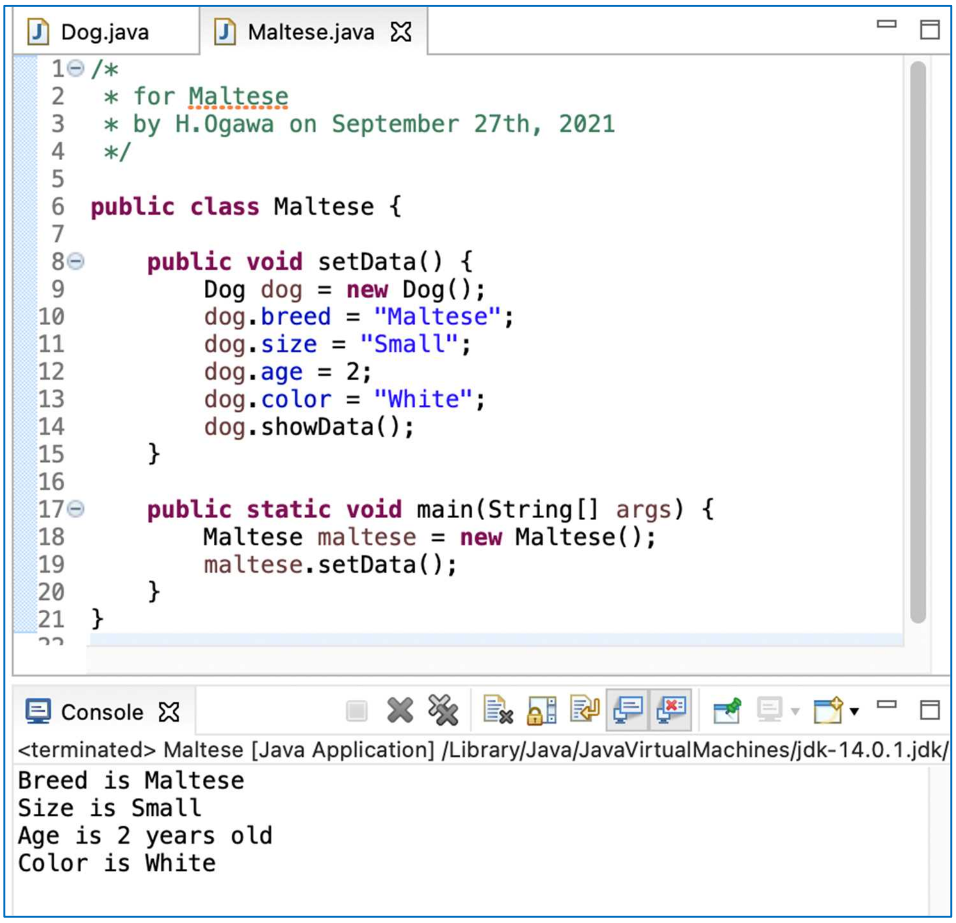
To reference the variables or execute the methods of the instance, use this form:

variable_of_class.variable_name	,or	variable_of_class.method(arguments

In lines 20 to 23, values are assigned to the variables set in lines 6 to 9. In line 24, the showData() method is executed. Akita Inu's information was displayed in the Console window. The symbol "+" in "System.out.println" is an operator that combines strings.

In previous program, the information of Akita Inu was set and used in the main method of Dog class. However, since there are other breeds of dogs, we create other classes using instances of Doc class. An example of Maltese is shown below.

"Maltese.java" must be created in the same "default package" as "Dog.java".



```java
/*
 * for Maltese
 * by H.Ogawa on September 27th, 2021
 */

public class Maltese {

    public void setData() {
        Dog dog = new Dog();
        dog.breed = "Maltese";
        dog.size = "Small";
        dog.age = 2;
        dog.color = "White";
        dog.showData();
    }

    public static void main(String[] args) {
        Maltese maltese = new Maltese();
        maltese.setData();
    }
}
```

```
<terminated> Maltese [Java Application] /Library/Java/JavaVirtualMachines/jdk-14.0.1.jdk/
Breed is Maltese
Size is Small
Age is 2 years old
Color is White
```

In main method of Maltese class, the instance of Maltese class was created and setData() method of Maltese class was executed. Note that the contents of setData() method are the same as those of the main method of Dog class. There is no variable declaration and description of showData() method in Maltese class. By using an instance of Dog class, both the variable and showData() method in Dog class can be used.