Class and Method

This document first introduces “String”, “Array”, and “for statement”. And how to create and use “Class” using these will be introduced.

1. String

Strings, which are widely used in Java programming, are a sequence of characters. In Java, strings are treated as objects.

1.1 Creating Strings

The most direct way to create a string is to use double quotes and write as follows:

```
String greeting = "Hello world! ";
```

Note: Single quotes cannot be used.

1.2 Concatenating Strings

The String class includes a method for concatenating two strings:

```
string1.concat(string2);
```

This returns a new string that is *string1* with *string2* added to it at the end.

Strings are more commonly concatenated with the “+” operator as follows:

```
string1 + string2 + string3
```

This creates a string consisting of three strings arranged in order.

The examples of concatenating strings is shown below.

Example program

```
String string1 = "Information Science";
String string2 = " and ";
String string3 = "Engineering";
System.out.println(string1 + string2 + string3);
```

Output

```
Information Science and Engineering
```

1. 3 String Methods

Table 1 shows the list of frequently used methods supported by String class.

Tabel 1. The list of frequently used methods supported by String class.

Method & Description of String
char charAt(int index) retrurns the character at the specified index.
Boolean endsWith(Sting suffix) tests if this string ends with the specified suffix .
Boolean equals(Object anObject) compares this string to the string handled by the specified object. Note that operator "==" compares objects themselves .
int length() returns the lengh of this string.
String[] split(String regex) splits this string around matches of the given regular expression.
Boolean startWith(Sting prefix) tests if this string starts with the specified prefix beginning a specified index.
String substring(int beginIndex, int endIndex) returns a new string that starts with the specified beginIndex and before the character at index endIndex. The length of the substring is endIndex – beginIndex.

Examples are shown in 5.1.

2. Array

An array is a very common type of data structure wherein elements must be of the same data type. Once defined, the size of an array is fixed and cannot increase to accommodate more elements. The first element of an array starts with index zero.

2. 1 Array Variables

Using an array in the program is a 3 step process.

(A) Declaring an Array

Syntax is as follows.

```
datatype[] arrayName;
```

or

```
datatype arrayName[];
```

We can use primitive data and Object as datatype (See 4.3 of "First Java Program", The first day).

(B) Constructing an Array

```
arrayName = new datatype[arraySize];
```

The *arraySize* is an integer greater than or equal to 1.
Declaration and Construction are combined as follows.

```
datatype[] arrayName = new datatype[arraySize];
```

or

```
datatype arrayName[] = new datatype[arraySize];
```

The "new" keyword is used to create a new Array object.

(C) Initialize an Array

A value can be assigned to each element of array. It is possible to realize all of step (A) to (C) at the same time as follows.

```
datatype[] arrayName = {value0, value1, ..., valueN};
```

An example:

```
int[] arrayInt = {1, 2, 3, 4, 5}
```

2. 2 Multidimensional Arrays

Multidimensional arrays are actually arrays of arrays. To declare a multidimensional array, specify each additional index using another set of square brackets.

two-dimensional array

```
datatype[][] arrayName = new datatype[arraySize1] [arraySize2];
```

three-dimensional array

```
datatype[][][] arrayName =
                new datatype[arraySize1] [arraySize2] [arraySize3];
```

2. 3 Array Length

To find out how many elements an array has, use the "length" property. The following is examples:

Example program

```
String[] cars = {"Toyota", "Honda", "Nissan", "Mazda"};
System.out.println(cars.length);
int[][] data = {{0, 1}, {2, 3}, {4, 5}, {6, 7}, {8, 9}};
System.out.println(data.length);
```

Output

```
4
5
```

3. Java for statement

   When we know exactly how many times we want to loop through a block of code, use the “for” statement. Following is the syntax of for statement.

```
for(Statement1; Statement2; Statement3){
    // code block to be executed
}
```

Statement1 is executed (one time) before the execution of the code block.
Statement2 defines the condition for executing the code block.
Statement3 is executed (every time) after the code block has been executed.

The example below will print the numbers 0 to 4.

Example program

```
for(int i = 0; i < 5; i++) {
    System.out.println(i);
}
```

Output

```
0
1
2
3
4
```

There is also a “for-each” statement, which is used exclusively to loop through elements in an array. Following is the syntax of for-each statement.

```
for(datatype variable: arrayname){
    // code block to be executed
}
```

]

is equivalent to:

```
for(int i = 0; i < arrayname.length; i++){
    datatype variable = arrayname[i];
    // code block to be executed
}
```

The following example outputs all elements int the "cars" array, using a for-each statement:

Example program

```
String[] cars = {"Toyota", "Honda", "Nissan", "Mazda"};
for(String s: cars) {
    System.out.println(s);
}
```

Output

```
Toyota
Honda
Nissan
Mazda
```

4. Class and Constructors

Java is an object-oriented programming language. Everything in Java is associated with classes and objects. An example of Dog class was introduced in Section 5.2 of "First Java Program", The first day. A dog is an object which has data (field), such as breed, age, size, color, and methods, such as sleep, sit, eat, etc.

A Class is like an object constructor, or a "blueprint" for creating object.

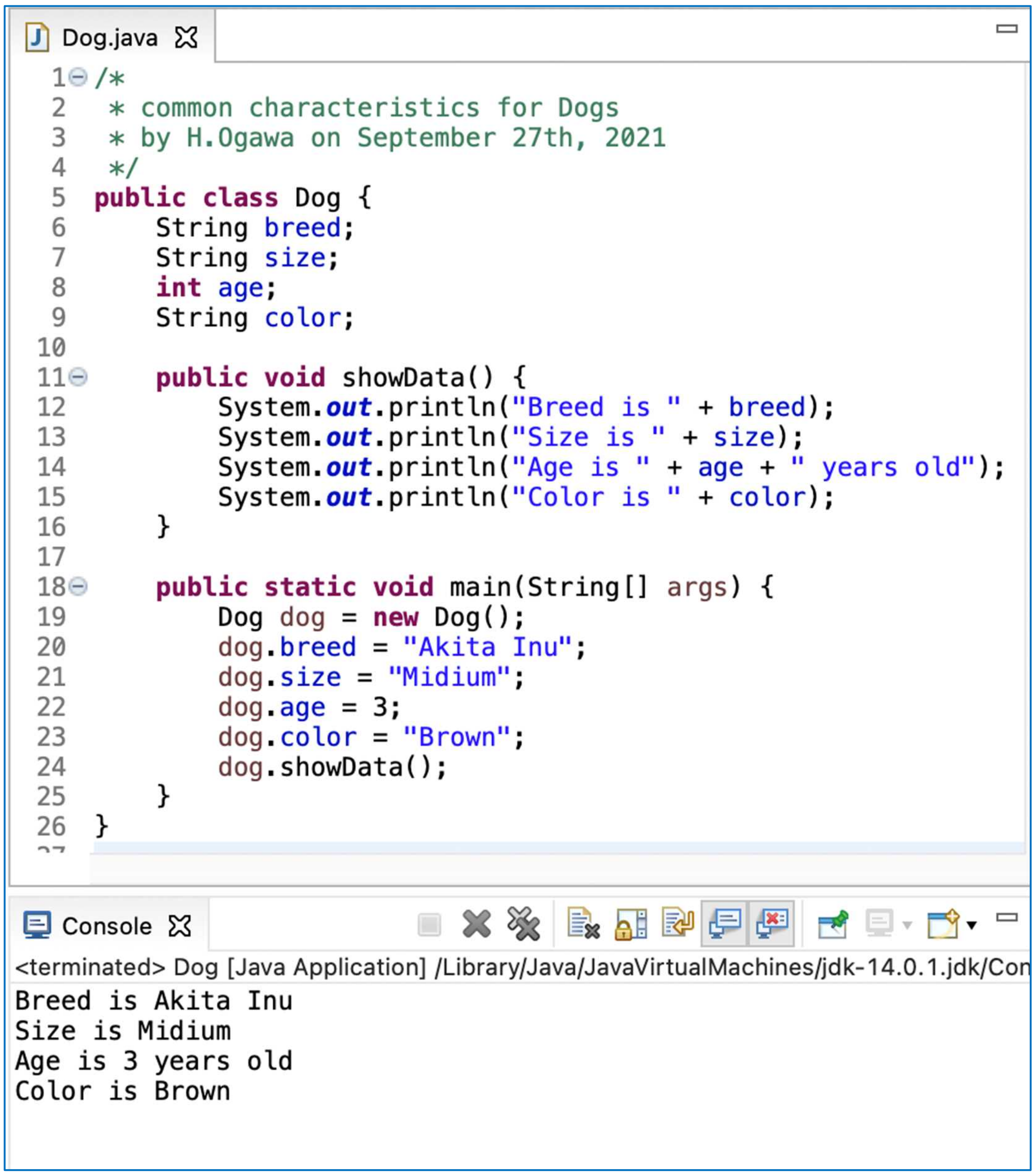
4. 1 Create a Class and an Object

To create a class, use the keyword "class". In principle, put keyword "public" before "class". Keyword "public" is as access modifier used for classes, data (field), methods and constructors, making them accessible by any other class.

In Java, an object is created from a class. If there is the class named "Dog" for example, so now we can use this to create objects. To create an object of "Dog", specify the class name, followed by the object name (variable), and use the keyword "new". The example is shown in Figure 1.

In the main method, an instance (object) of Dog class is created and assigned to the variable dog (line 19). In lines 20 to 23, values are assigned to the variables set to lines 6 to 9. In line 24, the shownData() method is executed.

The "void" in the first line of showData method specifies that the method does not have a return value.



```java
/*
 * common characteristics for Dogs
 * by H.Ogawa on September 27th, 2021
 */
public class Dog {
    String breed;
    String size;
    int age;
    String color;

    public void showData() {
        System.out.println("Breed is " + breed);
        System.out.println("Size is " + size);
        System.out.println("Age is " + age + " years old");
        System.out.println("Color is " + color);
    }

    public static void main(String[] args) {
        Dog dog = new Dog();
        dog.breed = "Akita Inu";
        dog.size = "Midium";
        dog.age = 3;
        dog.color = "Brown";
        dog.showData();
    }
}
```

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

Figure 1. Dog class.

4. 2 Constructors

A **constructor** is a special method that is used to initialize a newly created object. It can be used to initialize the objects to desired values or default values at the time of object creation. It is not mandatory for the coder to write a constructor for a class.

(A) Rules for Constructor

- ✓ Constructor name must be the same as its class name.
- ✓ A Constructor must have no explicit return type.

(B) Types of Constructors

There are three types of constructors: Default constructor, No-argument constructor and Parameterized constructor.

- Default constructor

  If we do not implement any constructor in our class, Java system inserts a default constructor into our code.

- No-argument constructor

  Constructor with no arguments is known as no-argument constructor. The signature is same as default constructor, however body can have any code unlike default constructor where the body of the constructor is empty.

- Parameterized constructor

  Constructor with arguments is known as Parameterized constructor. An example using a constructor for Dog class is Figure 2. Since all data are assigned by the constructor, it is not necessary to set each data of the field individually.

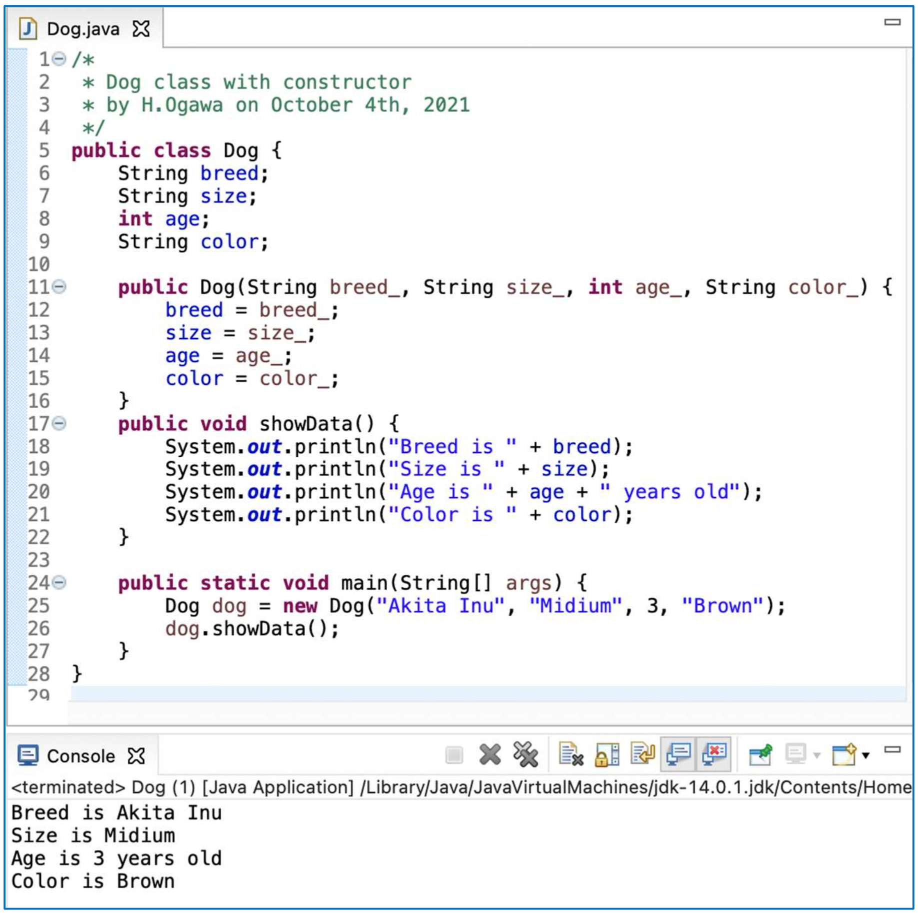
  As shown in Figure 3, the Akita_Inu class can be rewritten in the same way. Don't forget to create "Akita_Inu.java" in the same "default package" as "Dog.java".

(C) Constructor Overloading

Default constructor becomes invalid at the time of specifying any constructor. However, constructor overloading is a technique in Java in which a class can have any number of constructors that differ in parameter list. The Java system differentiates these constructors by taking into account the number of parameters in the list and their type.

Examples of valid constructors for class Dog are:

```
Dog()
Dog(String breed_)
Dog(int age_)
Dog(String breed_, String size_)
```

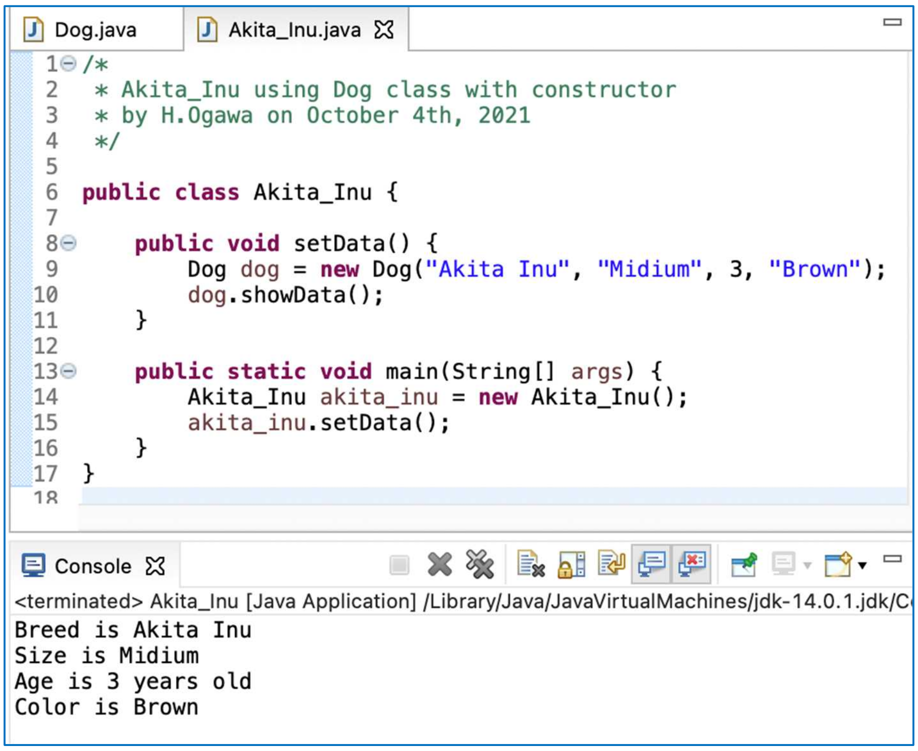

Dog.java

```java
/*
 * Dog class with constructor
 * by H.Ogawa on October 4th, 2021
 */
public class Dog {
    String breed;
    String size;
    int age;
    String color;

    public Dog(String breed_, String size_, int age_, String color_) {
        breed = breed_;
        size = size_;
        age = age_;
        color = color_;
    }
    public void showData() {
        System.out.println("Breed is " + breed);
        System.out.println("Size is " + size);
        System.out.println("Age is " + age + " years old");
        System.out.println("Color is " + color);
    }

    public static void main(String[] args) {
        Dog dog = new Dog("Akita Inu", "Midium", 3, "Brown");
        dog.showData();
    }
}
```

Console

```
<terminated> Dog (1) [Java Application] /Library/Java/JavaVirtualMachines/jdk-14.0.1.jdk/Contents/Home
Breed is Akita Inu
Size is Midium
Age is 3 years old
Color is Brown
```

Figure 2. Dog class with constructor.


Dog.java | Akita_Inu.java

```java
/*
 * Akita_Inu using Dog class with constructor
 * by H.Ogawa on October 4th, 2021
 */

public class Akita_Inu {

    public void setData() {
        Dog dog = new Dog("Akita Inu", "Midium", 3, "Brown");
        dog.showData();
    }

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

Console

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

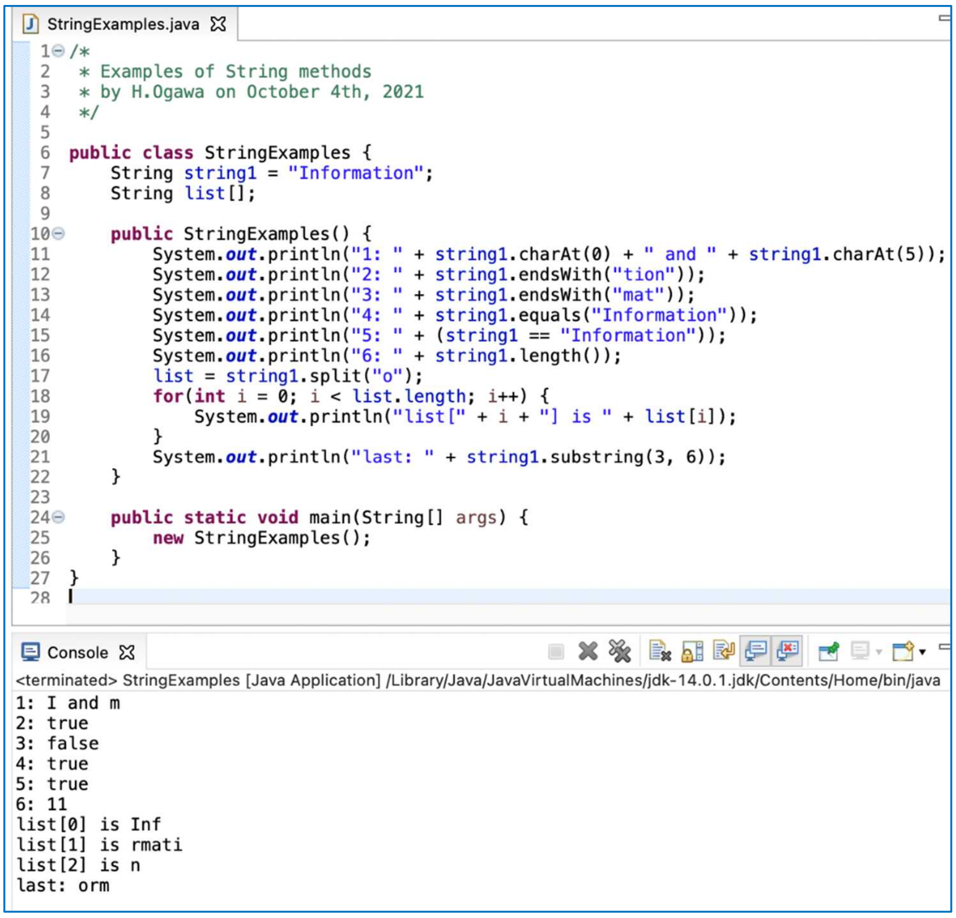
Figure 3. Akita_Inu Class using Dog class with constructor.

5. Examples

5.1 String

Examples using String methods are shown in Figure 4. All processing is described in the constructor.

line 11: Result of extracting the first and sixth characters of the word “Information”.
line 12: The result is “true” because string1 ends with “tion”.
line 13: The result is “false” because the last string in string1 in not “mat”.
line 14: Check whether the content of string1 is “Information”.
line 15: Check whether the content of string1 is “Information”. In this example, the result is “true”. However, the result was “false” in the case of old version of Java.
line 16: Output of the number of charactor in string1.
line 17-20: Output three words obtained by splitting the word “Information” with the letter “o”.
line 21: Obtain from 4th character to 6th character of “Information”.



```
 1  /*
 2   * Examples of String methods
 3   * by H.Ogawa on October 4th, 2021
 4   */
 5
 6  public class StringExamples {
 7      String string1 = "Information";
 8      String list[];
 9
10      public StringExamples() {
11          System.out.println("1: " + string1.charAt(0) + " and " + string1.charAt(5));
12          System.out.println("2: " + string1.endsWith("tion"));
13          System.out.println("3: " + string1.endsWith("mat"));
14          System.out.println("4: " + string1.equals("Information"));
15          System.out.println("5: " + (string1 == "Information"));
16          System.out.println("6: " + string1.length());
17          list = string1.split("o");
18          for(int i = 0; i < list.length; i++) {
19              System.out.println("list[" + i + "] is " + list[i]);
20          }
21          System.out.println("last: " + string1.substring(3, 6));
22      }
23
24      public static void main(String[] args) {
25          new StringExamples();
26      }
27  }
28
```

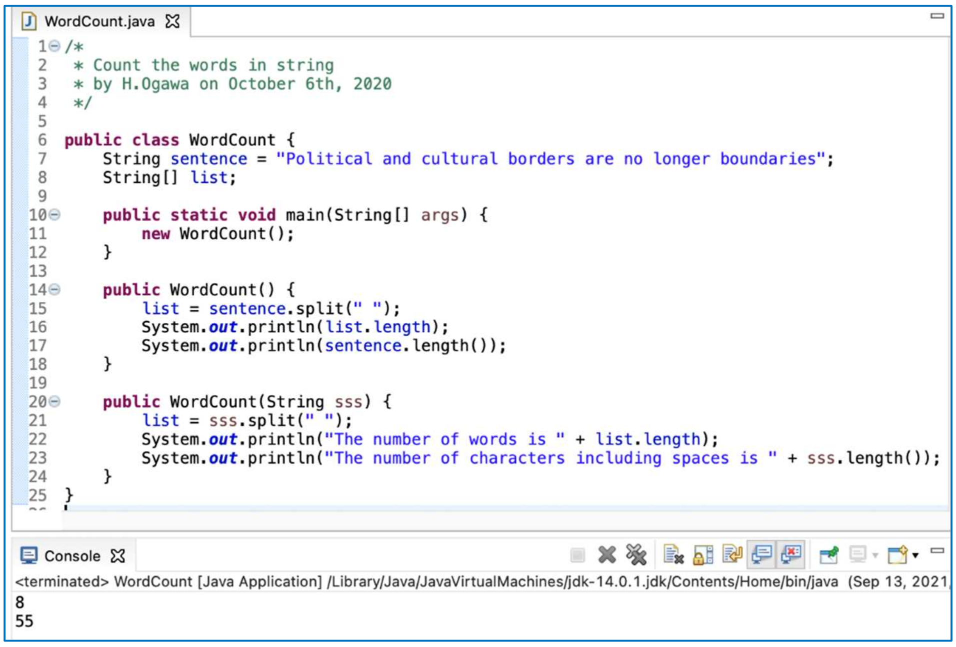
```
<terminated> StringExamples [Java Application] /Library/Java/JavaVirtualMachines/jdk-14.0.1.jdk/Contents/Home/bin/java
1: I and m
2: true
3: false
4: true
5: true
6: 11
list[0] is Inf
list[1] is rmati
list[2] is n
last: orm
```

Figure 4. Examples using String methods.

5. 2 Word count

Figure 5 shows as example of word count. There are 2 constructors. When WordCount is executed, the constructor WordCount() is invoked. The two numbers output are the number of words and the number of characters including spaces of sentence, respectively.

Figure 6 shows Application class which call the constructor WordCount(String sss) of WordCount class. The output also shows the number of words and the number of characters including spaces of sentence, respectively. "Application.java" must be created in the same "default package" as "WordCount.java".

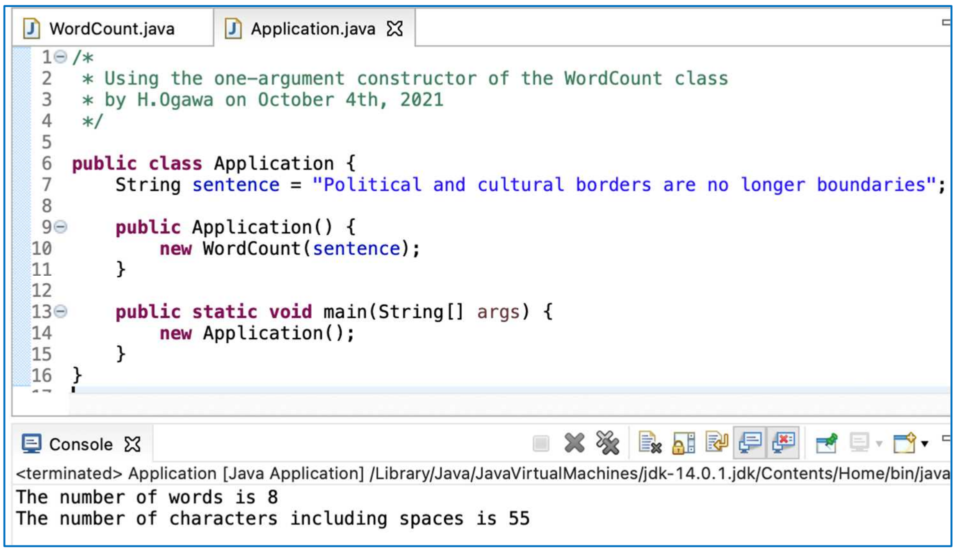

WordCount.java

```java
/*
 * Count the words in string
 * by H.Ogawa on October 6th, 2020
 */

public class WordCount {
    String sentence = "Political and cultural borders are no longer boundaries";
    String[] list;

    public static void main(String[] args) {
        new WordCount();
    }

    public WordCount() {
        list = sentence.split(" ");
        System.out.println(list.length);
        System.out.println(sentence.length());
    }

    public WordCount(String sss) {
        list = sss.split(" ");
        System.out.println("The number of words is " + list.length);
        System.out.println("The number of characters including spaces is " + sss.length());
    }
}
```

Console

```
<terminated> WordCount [Java Application] /Library/Java/JavaVirtualMachines/jdk-14.0.1.jdk/Contents/Home/bin/java  (Sep 13, 2021
8
55
```

Figure 5. Word Count Class


WordCount.java Application.java

```java
/*
 * Using the one-argument constructor of the WordCount class
 * by H.Ogawa on October 4th, 2021
 */

public class Application {
    String sentence = "Political and cultural borders are no longer boundaries";

    public Application() {
        new WordCount(sentence);
    }

    public static void main(String[] args) {
        new Application();
    }
}
```

Console

```
<terminated> Application [Java Application] /Library/Java/JavaVirtualMachines/jdk-14.0.1.jdk/Contents/Home/bin/java
The number of words is 8
The number of characters including spaces is 55
```

Figure 6. Application class which call WordCount class.