CSE3040 Java Language

Lecture #02

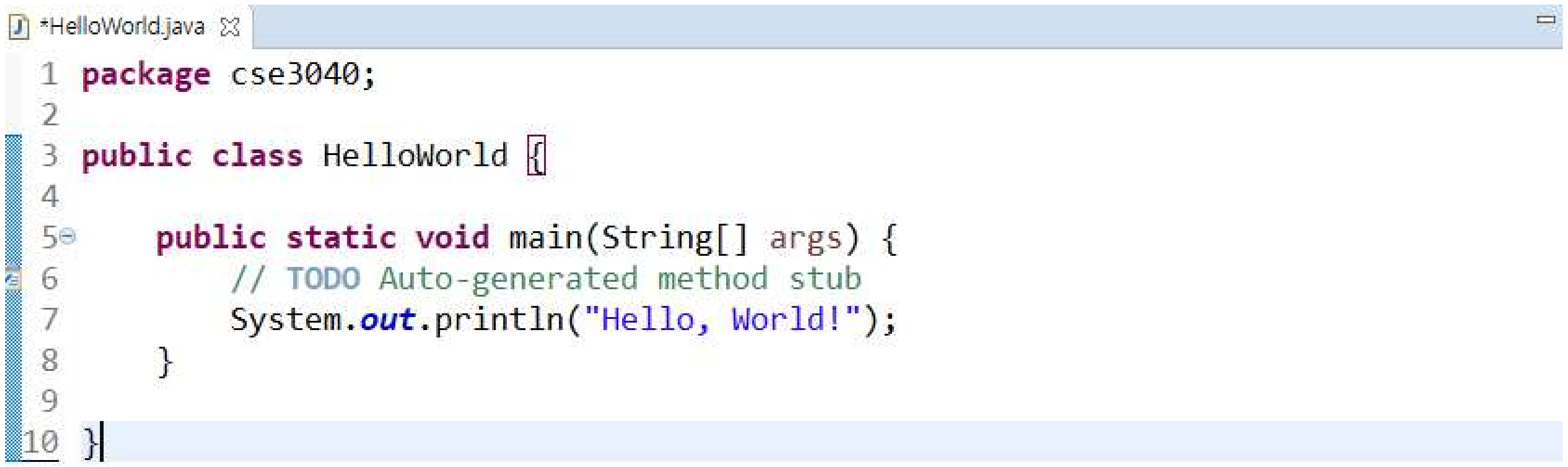
Dept. of Computer Engineering,

Sogang University

Java Programming Basics

1.1. A "Hello, World!" Program

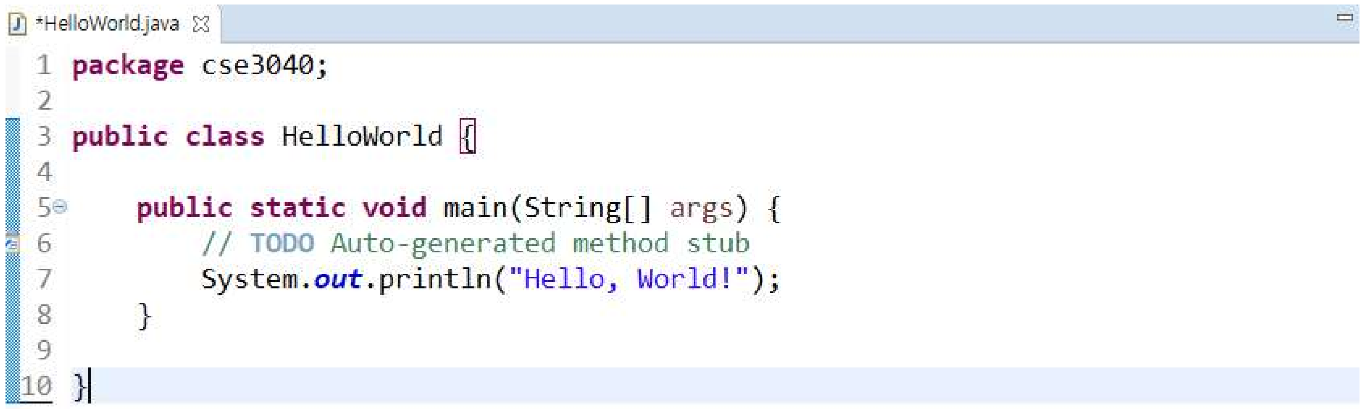
- Let us write our first program.



```java
package cse3040;

public class HelloWorld {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        System.out.println("Hello, World!");
    }

}
```

A "Hello, World!" Program

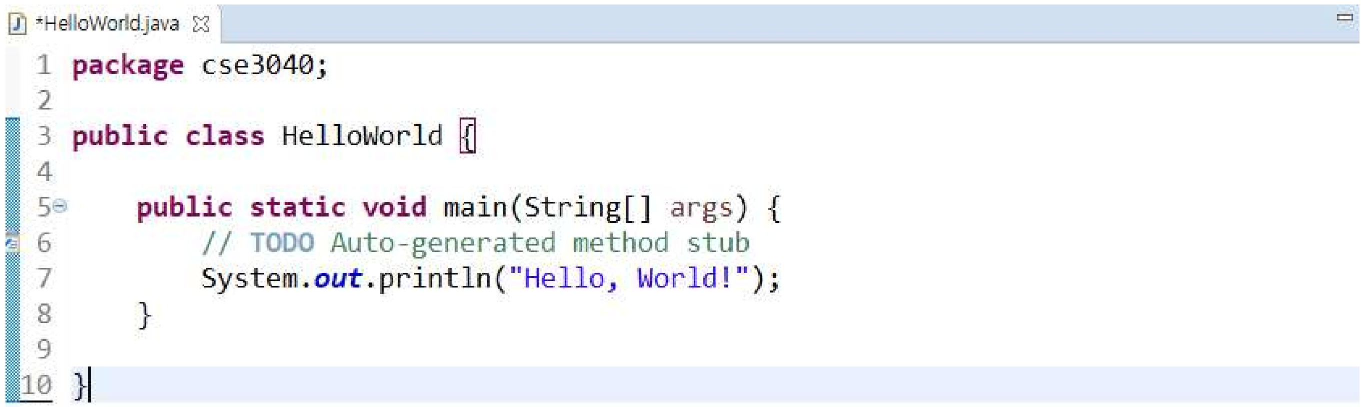
- In Java, everything is an object.
- An object is an instance of a class.
- This program has a single class: HelloWorld



```java
package cse3040;

public class HelloWorld {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        System.out.println("Hello, World!");
    }

}
```

A "Hello, World!" Program

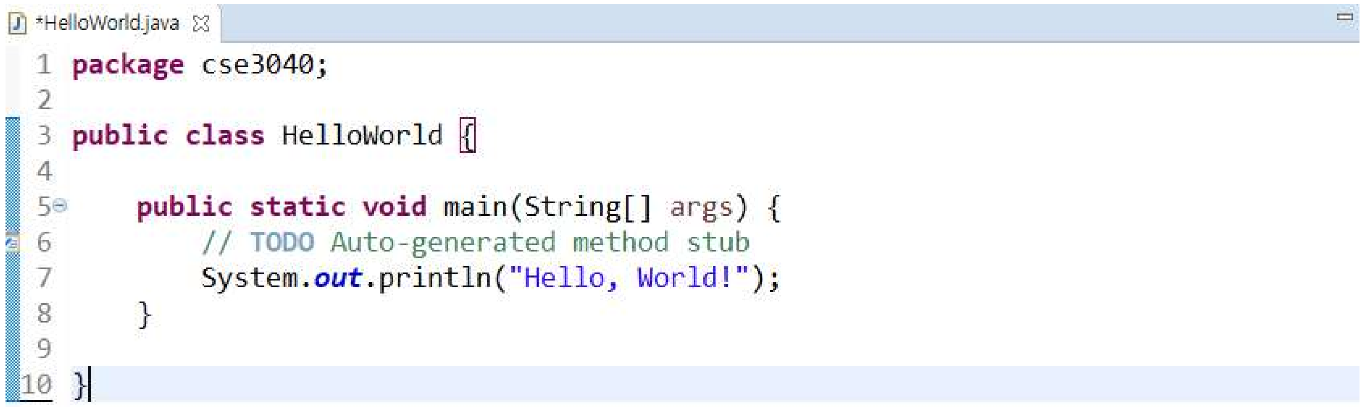
- A class may contain methods. The HelloWorld class has one method: main.
- The main method is the entry point (starting point) of a program.
- It is declared as static, which means this method does not require an instance.
- The method's return type is void, which means the method returns nothing.
- The method takes an array of Strings as input arguments.
  - Datatypes and arrays explained later.



```
package cse3040;

public class HelloWorld {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        System.out.println("Hello, World!");
    }

}
```

A "Hello, World!" Program

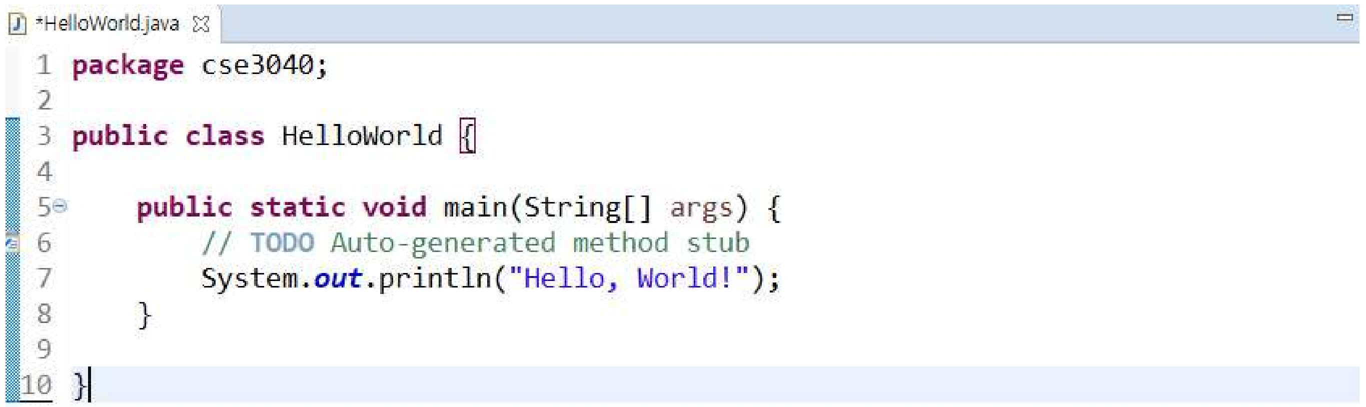
- We can see the keyword public in front of the class and the method.
  - The opposite of public is private.
  - They are called access modifier.
    - There are four access modifiers: public, protected, private, and (nothing).
    - The meaning of public and private will be explained later.



```java
package cse3040;

public class HelloWorld {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        System.out.println("Hello, World!");
    }

}
```

A "Hello, World!" Program

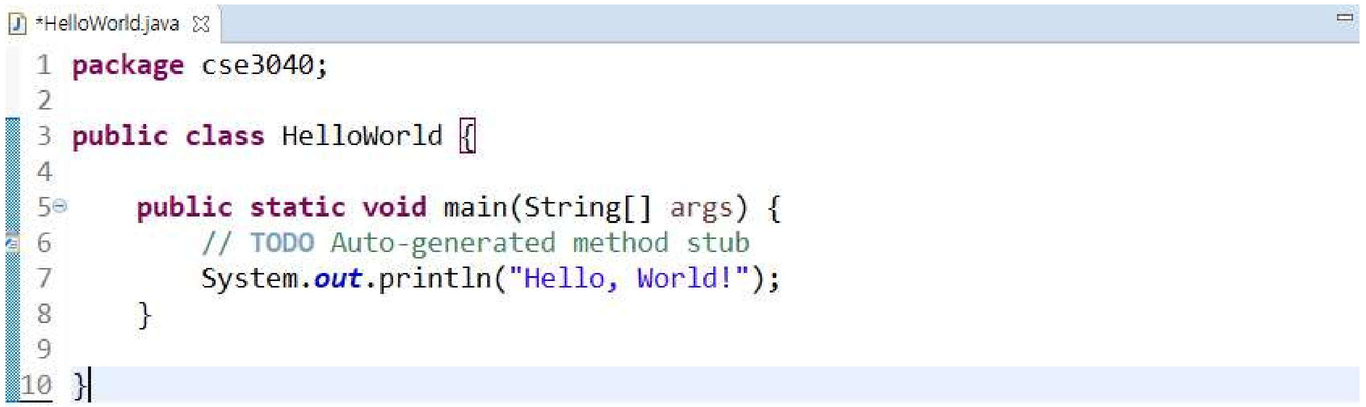
- Package
  - The first line says "package cse3040;"
  - You can put multiple related classes in a package.
    - Just like putting related files in a folder
  - You can use hierarchical packages
    - E.g.) package sogang.cse.cse3040;



```java
package cse3040;

public class HelloWorld {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        System.out.println("Hello, World!");
    }

}
```

A "Hello, World!" Program

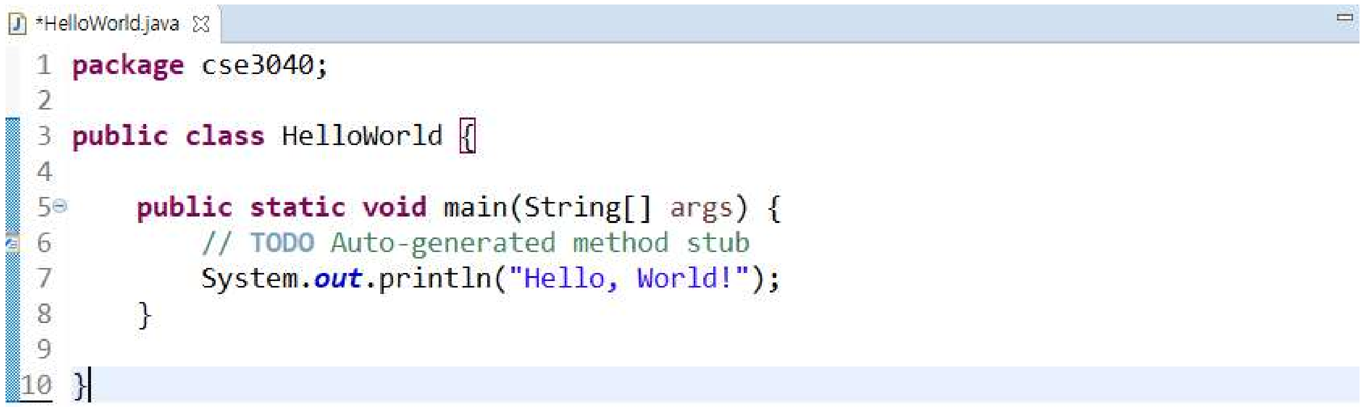
- Comment
  - All characters after “//” until the end of the line are comments and will be ignored by the compiler.
  - You can also use “/* comment */”. Everything between “/*” and “*/” are ignored (regardless of lines.)



```java
package cse3040;

public class HelloWorld {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        System.out.println("Hello, World!");
    }

}
```

A "Hello, World!" Program

- Contents of the method
  - The "main" method has a single line.
    - System.out.println("Hello, World!");
  - This is a frequently used Java library method for printing messages on the standard output.
    - System.out is an object which indicates the standard output.



```java
package cse3040;

public class HelloWorld {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        System.out.println("Hello, World!");
    }

}
```

Calling Methods in Java

- Look at the following line carefully.

```
System.out.println("Hello, World!");
```

- In this line, System.out is an instance of class PrintStream.
  - PrintStream and System.out, they are all from libraries.
  - Libraries: classes and methods written by other people that you can use for your convenience.

- println is a method defined in the class PrintStream. This method is called "instance method", because it can be called using the instance of a class.

- When calling an instance method, we use "dot notation" like the following:
  - *Object.methodName(arguments)*
  - In this example, there is a single argument, "Hello, World!".

Calling Methods in Java

- "Hello, World!" is an instance of class String.
- Class String has an instance method called length.
- So you can call the method length like this:

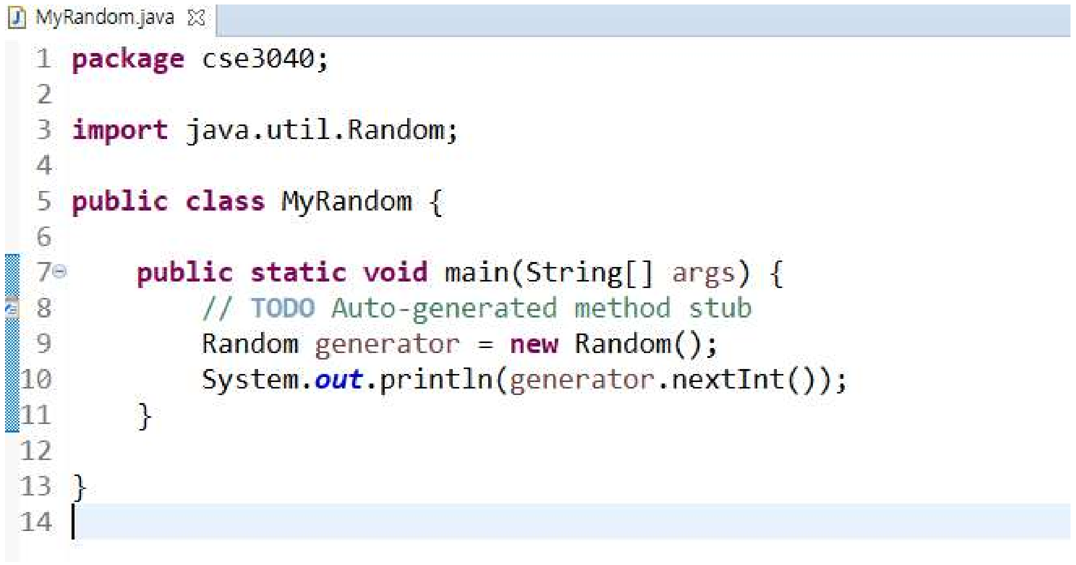
```
"Hello, World!".length()
```

- This will return the length of the string.
- Try this:

```
System.out.println("Hello, World!".length());
```

Calling Methods in Java

- System.out or "Hello, World!", they are instances of classes.
- Usually, we create an instance of a class using the keyword new.
- For example, we can create an instance of class Random, which is a class defined in a library.

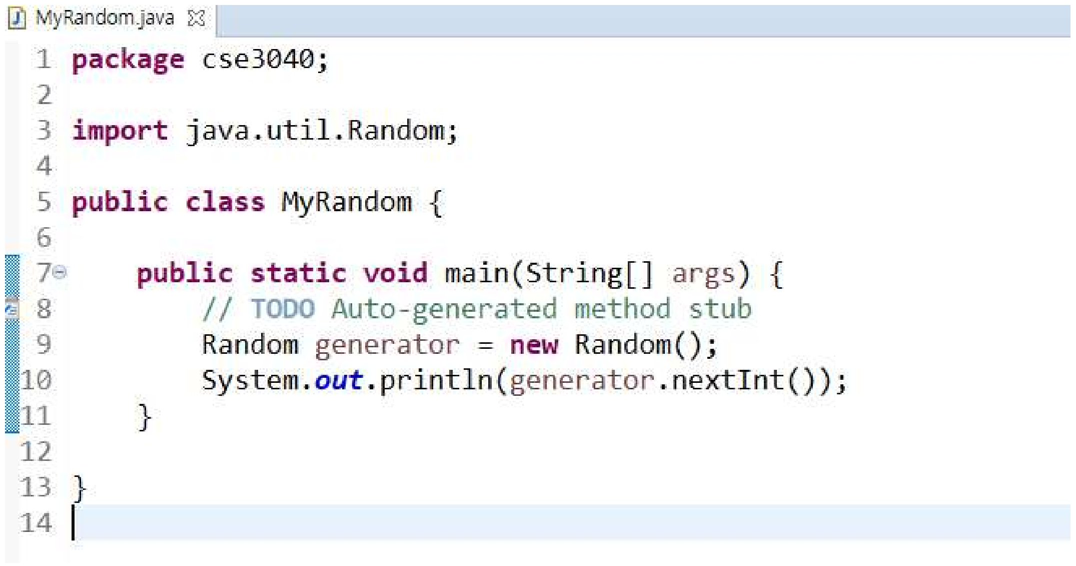


```java
package cse3040;

import java.util.Random;

public class MyRandom {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Random generator = new Random();
        System.out.println(generator.nextInt());
    }

}
```

Calling Methods in Java

- import is used to declare that this program will be using a certain library.
  - In this case, we are using library java.util.Random, which contains class Random.

- Random generator = new Random();
  - generator becomes an instance of class Random.
  - Class Random has an instance method nextInt, which returns a random integer.
  - Try running this code multiple times.



MyRandom.java

```java
package cse3040;

import java.util.Random;

public class MyRandom {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Random generator = new Random();
        System.out.println(generator.nextInt());
    }

}

```

1.2. Primitive Types

- In Java, most variables are objects. (= instance of a class)
- Still, Java has primitive types.
  - Integer types
    - byte (1 byte): -128 ~ 127
    - short (2 bytes): -32,768 ~ 32,767
    - int (4 bytes): -2,147,483,648 ~ 2,147,483,647
    - long (8 bytes): -9,223,372,036,854,775,808 ~ 9,223,372,036,854,775,807
  - If you write an integer, Java thinks it is of type int.
  - If you want to write a long type integer, put L at the end.
    - 400000 → int type integer
    - 400000L → long type integer
  - You can write hexadecimal numbers
    - 0xCAFEBABE
  - You can also write binary numbers
    - 0b1001

1.2. Primitive Types

```
class dtByte {
   public static void main(String[] args)
     {
     // print values from 125 to 134
     byte   b=125;                   // range of byte type : -128 ~ 127

     for (int i=1 ; i<=10 ; i++)     // i : integer type
         System.out.print(" " + b++);
      }
}
```

1.2. Primitive Types

- In Java, most variables are objects. (= instance of a class)
- Still, Java has primitive types.
  - Floating-point types
    - float (4 bytes): 6~7-digit precision
    - double (8 bytes): 15-digit precision
  - If you are writing floating point numbers, Java thinks it is a double type number.
    - 3.14 → double type
    - 3.14E5 → double type (e: exponent) = 3.14 x 10^5
    - 3.14F → float type
  - char type
    - character type
    - In Java, String is used much more frequently than char.
  - boolean type
    - true, false
    - Note that they are not numbers (true and false are not 1 and 0.)
    - You cannot convert boolean to integers

1.2. Primitive Types

```
public static void main(String[] args)
     {
     float  f=0;
     double d=0;

     for (int i=1 ; i<=100000 ; i++)      // i : integer type
         {
         f += 100000;
         d += 100000;
         }

      System.out.println("float  : " + f/100000.0);
      System.out.println("double : " + d/100000.0);
      }
```

1.2. Primitive Types

```
class dtBoolean {
   public static void main(String[] args)
      {
      boolean b1, b2;
      b1 = (5 == 3);          // false
      b2 = (5 > 3);           // true
      System.out.println("5 == 3          : " + b1);
      System.out.println("5 > 3           : " + b2);
      System.out.println("5 == 3 && 5 > 3 : " + (b1 && b2));
      System.out.println("5 == 3 || 5 > 3 : " + (b1 || b2));
      }
}
```

1.2. Primitive Types

```
class dtChar {
   public static void main(String[] args)
      {
      char  c;

      // 0 ~ 255 char print in the Unicode encoding scheme
      for (c=0 ; c<256 ; c++)
          System.out.print(" " + c);
      System.out.println("\n\n");

      c = (char)('A'+3);    // c='D'
      System.out.println(c);

      c -= 2;               // c='B'
      System.out.println(c);

      c = (char)('A' + '5');// Unicode(ASCII) 118->'v'
      System.out.println(c);
      }
}
```

1.3. Variables

- The basic syntax of Java is very similar to C/C++.

- Variable definition
  - int total;
  - int total = 0;   // with initialization
  - int total = 0, count;   // multiple definitions in one line
  - Random generator = new Random();   // definition + class instantiation

- Variable name
  - Must start with a letter. (A number is not allowed.)
  - The variable name can contain letters, numbers, _ and $.
  - Variable name is case-sensitive.

1.3. Variables

- Variable initialization
  - Before using, a variable must be initialized.
  - The following code is wrong, because count is used without initialization.

```
int count;
count++;
```

- Constants
  - To declare a constant, use keyword final.
  - You cannot modify a variable declared as final.
  - According to naming convention, a constant variable should be named using upper-case letters.

```
final int DAYS_PER_WEEK = 7;
```

Programming Lab #02

02-1. Length of a String

- Write a Java program that prints the following on the screen.
  - The first line should print **Hello, World!**
  - The second line should print the length of the string, "Hello, World!".

02-1. Length of a String

- Write a Java program that prints the following on the screen.
  - The first line should print **Hello, World!**
  - The second line should print the length of the string, "Hello, World!".

```
public class Ex02_1 {
  public static void main(String[] args) {
    System.out.println("Hello, World!");
    System.out.println("Hello, World!".length());
  }
}
```

02-2. Random Number Generation

- Write a Java program that prints 5 random integers on the screen.

02-2. Random Number Generation

- Write a Java program that prints 5 random integers on the screen.

```
import java.util.Random;

public class Ex02_2 {
  public static void main(String[] args) {
    // print 5 random integers
    Random generator = new Random();
    System.out.println(generator.nextInt());
  }
}
```

02-3. Primitive Types

- The following Java code produces unexpected result. Modify the code so that it produces the result as expected (print values from 125 to 134.)

```
public class Ex02_3 {
  public static void main(String[] args) {
    // print values from 125 to 134
    byte b = 125;
    for(int i=1; i<=10; i++)
      System.out.print(" " + b++);
    System.out.println();
  }
}
```

02-3. Primitive Types

- The following Java code produces unexpected result. Modify the code so that it produces the result as expected (print values from 125 to 134.)

```
public class Ex02_3 {
  public static void main(String[] args) {
    // print values from 125 to 134
    int b = 125;
    for(int i=1; i<=10; i++)
      System.out.print(" " + b++);
    System.out.println();
  }
}
```

02-4. float vs. double

- Compare the two results and understand why the results are produced that way.

```
public class Ex02_4 {
  public static void main(String[] args) {
    float f = 0;
    double d = 0;
    for(int i=1; i<=100000; i++) {
      f += 100000;
      d += 100000;
    }
    System.out.println("float  : " + f / 100000.0);
    System.out.println("double : " + d / 100000.0);
  }
}
```

02-5. boolean

- Type and execute the following Java program and understand the results.
- Try changing the predicates and observe how the results change.

```
public class Ex02_5 {
  public static void main(String[] args) {
    boolean b1, b2;
    b1 = (5 == 3);
    b2 = (5 > 3);
    System.out.println("5 == 3           : " + b1);
    System.out.println("5 > 3            : " + b2);
    System.out.println("5 == 3 && 5 > 3 : " + (b1 && b2));
    System.out.println("5 == 3 || 5 > 3 : " + (b1 || b2));
  }
}
```

02-6. char

- Type and execute the following Java program and understand the results.
- Try changing the predicates and observe how the results change.

```
public class Ex02_6 {
  public static void main(String[] args) {
    char c;
    for(c = 0; c < 256; c++)
      System.out.print(" " + c);
    System.out.println("\n");
    c = (char)('A' + 3);
    System.out.println(c);
    c -= 2;
    System.out.println(c);
    c = (char)('A' + '5');
    System.out.println(c);
  }
}
```

End of Class

Instructor office: AS818A

Email: jso1@sogang.ac.kr