CSE2016 Programming Methodology

Week 2: Simple Java Program

Instructor: Jinyoung Han (jinyounghan@hanyang.ac.kr)

HANYANG UNIVERSITY

Contents

HANYANG UNIVERSITY

Today's Schedule

1. An Application and its Architecture
2. How to Build and Execute an Application
3. How the Application Works
4. How One Object Constructs Another
5. Syntax and Semantics
6. Java Package
7. Summary

01. An Application and its Architecture

HANYANG UNIVERSITY

Java Programming

- Design
  - Draw a class diagram
  - Either by hand or computer software
- Implementation
  - Write a detailed instruction
- Execution
  - Compile and execute the Java file
  - Class is stored in a memory and becomes an “object”

HANYANG UNIVERSITY

Main Method

- A method that starts the program
  - "main"
- After an object is created in the memory, the "main" method is called and its instructions are executed
- Java application has a "main" method

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

HANYANG UNIVERSITY

A Simple Example

- Let's write a Java program that shows the following result:

```
Hello World!
100
```

02. How to Build and Execute an Application

HANYANG UNIVERSITY

An Execution Step

1. class “Hello” must be typed and saved in the file, Hello.java.

2. The program's spelling and grammar must be checked by the Java compiler; that is, the program must be compiled.

3. The program must be started (executed).

01. An Application and its Architecture

HANYANG UNIVERSITY

Hello.java

class name

```
/* Class Hello */
public class Hello
{
    public static void main(String[] args)
    {
        System.out.println("Hello World!");
        System.out.println(100);
    }
}
```

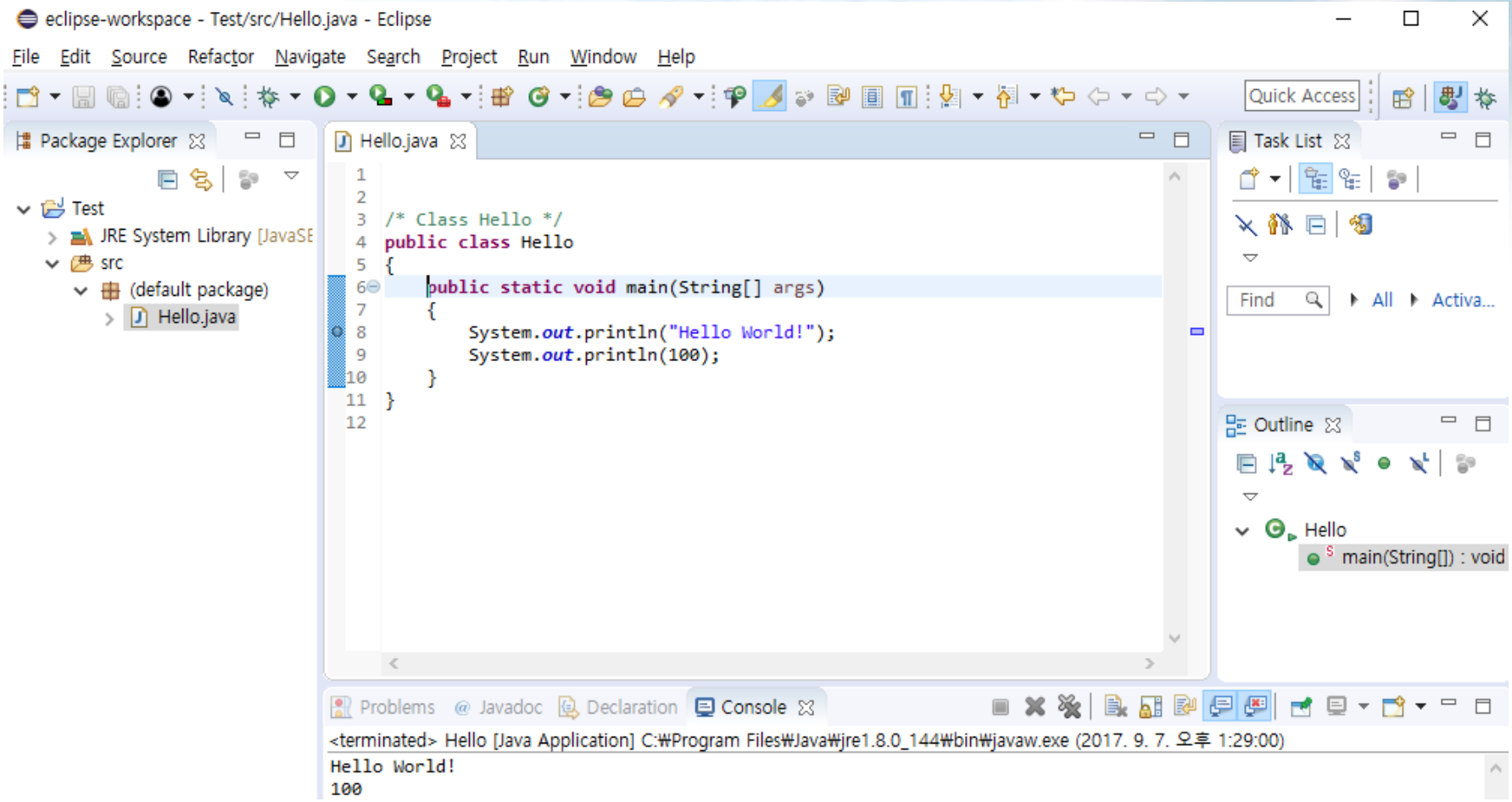
standard output

02. How to Build and Execute an Application

HANYANG UNIVERSITY

In Eclipse

- Project creation >> Class creation >> Run

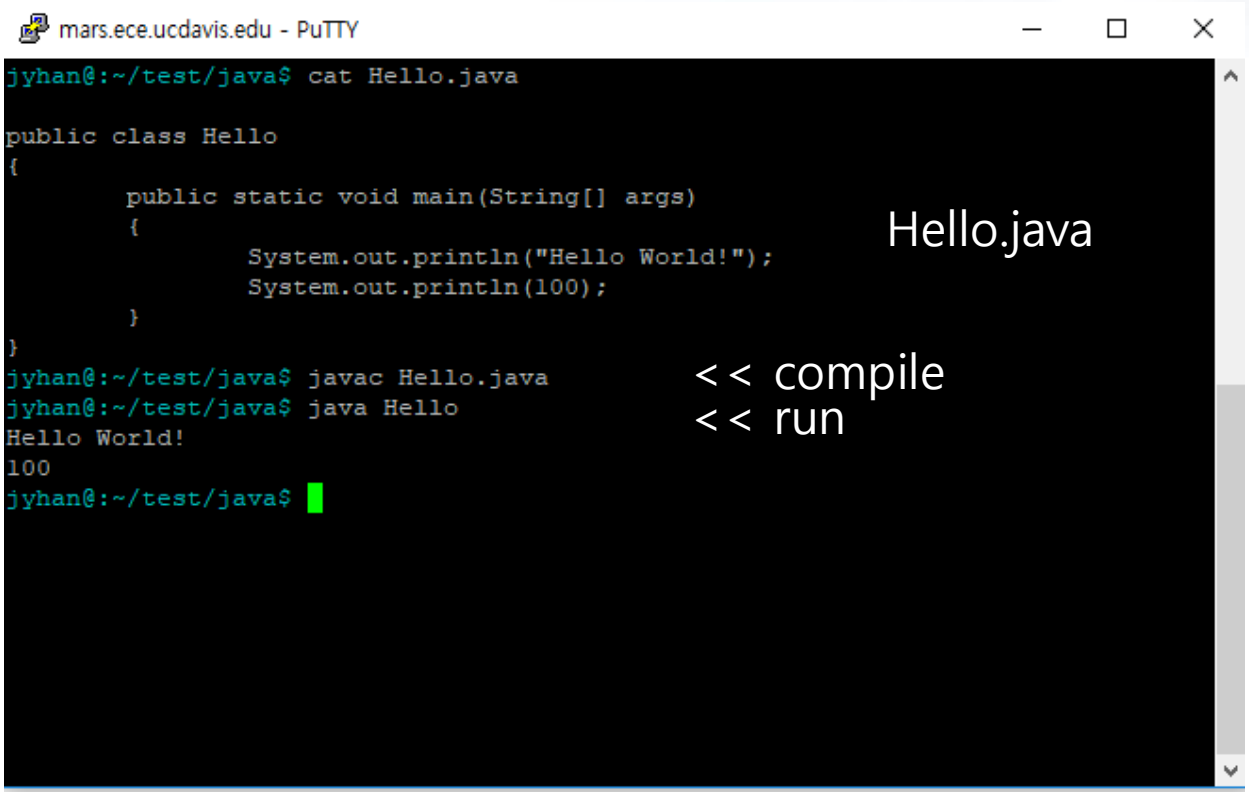


02. How to Build and Execute an Application

HANYANG UNIVERSITY

If you use a terminal

- Write Hello.java
- Compile: javac Hello.java
- Run: java Hello



An Execution Trace

- JVM (Java Virtual Machine) runs
  - Basic objects such as “System.out” are in the memory
- Hello.java -> Hello.class through the compile process (javac)
- When “java Hello” is executed, “Hello” object is in the memory
- JVM calls the main method of “Hello”

03. How the Application Works

HANYANG UNIVERSITY

An Execution Trace: Hello

① Staring main method

Hello.java

```
/* Class Hello */
public class Hello
{
    public static void main(String[] args)
    {
        System.out.println("Hello World!");
        System.out.println(100);
    }
}
```

System.out

```
println(x)
{
        printing x
        in the display
}
```

03. How the Application Works

An Execution Trace: Hello

① Staring main method

Hello

```
/* Class Hello */
public class Hello
{
    public static void main(String[] args)
    {
     ② System.out.println("Hello World!");
        System.out.println(100);
    }
}
```

System.out

```
println(x)
{
        printing x
        in the display
}
```

03. How the Application Works

HANYANG UNIVERSITY

An Execution Trace: Hello

① Staring main method

Hello

```
/* Class Hello */
public class Hello
{
    public static void main(String[] args)
    {
     ② System.out.println("Hello World!");
        System.out.println(100);
    }
}
```

③ run println method

System.out

```
println(x)
{
        printing x
        in the display
}
```

03. How the Application Works

HANYANG UNIVERSITY

An Execution Trace: Hello

① Staring main method

Hello

```
/* Class Hello */
public class Hello
{
    public static void main(String[] args)
    {
     ② System.out.println("Hello World!");
       System.out.println(100);
    }
}
```

④ wait until the end of method

③ run println method

System.out

④ run

```
println(x)
{
        printing x
        in the display
}
```

03. How the Application Works

HANYANG UNIVERSITY

An Execution Trace: Hello

① Staring main method

Hello

```
/* Class Hello */
public class Hello
{
    public static void main(String[] args)
    {
      ② System.out.println("Hello World!");
      ⑤ System.out.println(100);
    }
}
```

③ run println method

⑥ run println

System.out

④ run

```
println(x)
{
        printing x
        in the display
}
```

03. How the Application Works

An Execution Trace: Hello

① Staring main method

Hello

```
/* Class Hello */
public class Hello
{
    public static void main(String[] args)
    {
      ② System.out.println("Hello World!");
      ⑤ System.out.println(100);
    }
}   ⑦ the program ends
```

③ run println method

⑥ run println

System.out

④ run

```
println(x)
{
        printing x
        in the display
}
```

04. How One Object Constructs Another

HANYANG UNIVERSITY

Name and Date

- Previous example – sending a message to an existing object
- This example – sending a message to a ***newly created object***

- How?
  - Let's learn from a new example: Name and Date

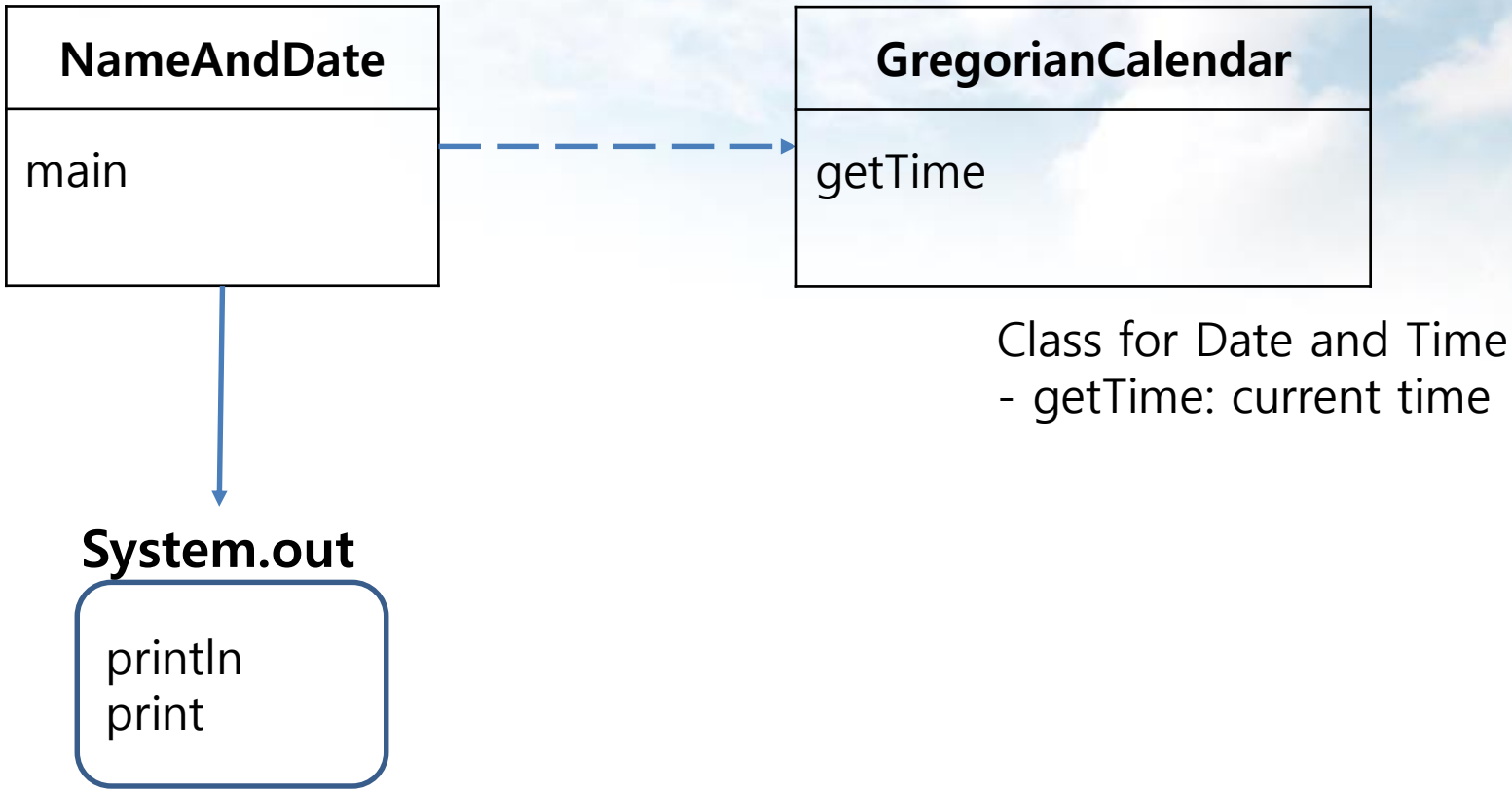
- "NameAndDate"

```
Name and Date Class.
Hanyang ICT --- Fri Sep 07 10:18:08 KST 2018
Finished.
```

04. How One Object Constructs Another

HANYANG UNIVERSITY

Class Diagram



Class for Date and Time
- getTime: current time

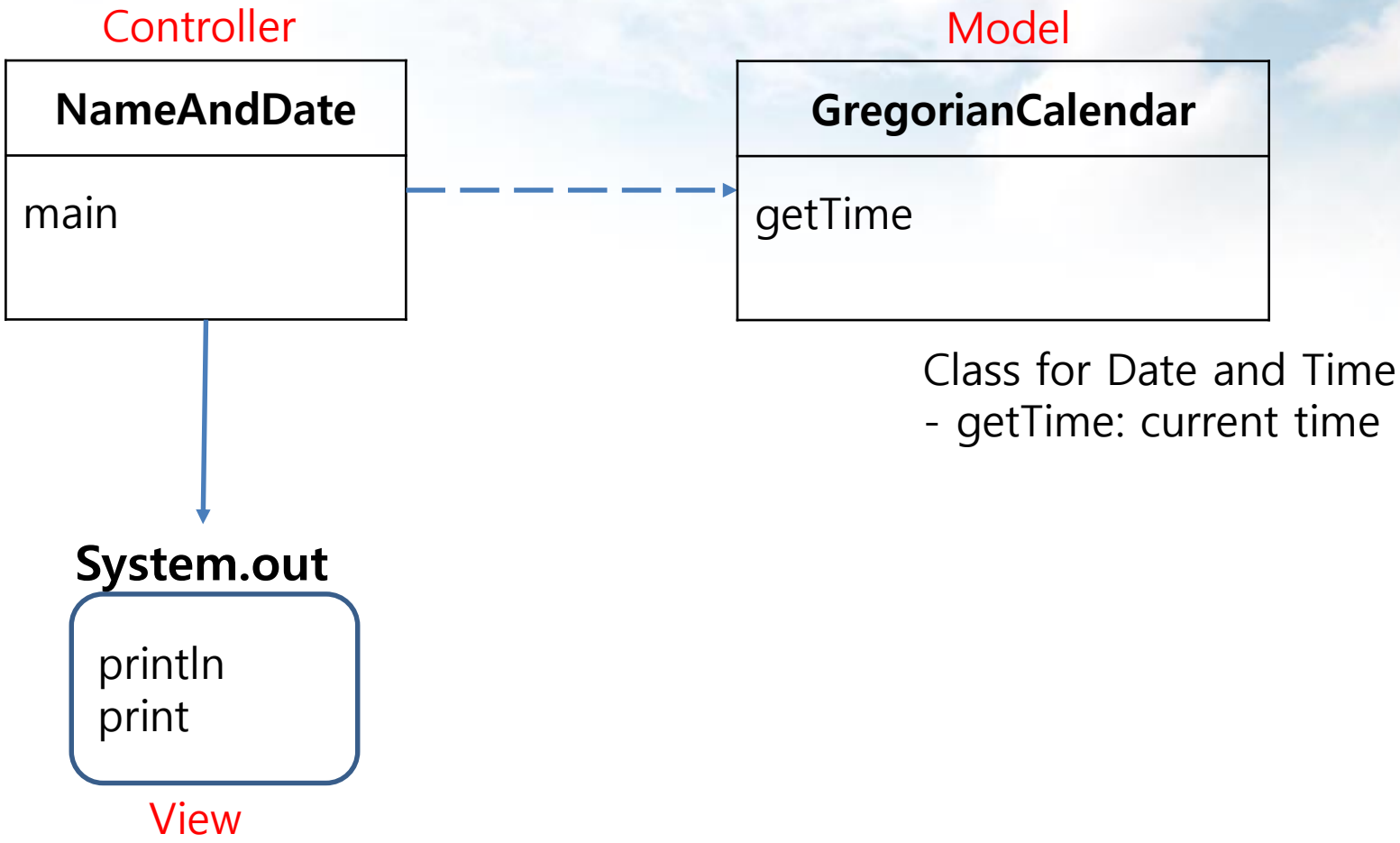
Remind - MVC

- Model-View-Controller
  - View: Interaction with users
  - Controller: Management of information exchanges
  - Model: Calculation

04. How One Object Constructs Another

HANYANG UNIVERSITY

Class Diagram



04. How One Object Constructs Another

HANYANG UNIVERSITY

NamdAndData.java

```
import java.util.*; // Java package for Gregorian Calendar
/** NameAndDate class*/
public class NameAndDate
{
    public static void main(String[] args)
    {
        System.out.println("Name and Date Class.");
        System.out.print("Hanyang ICT --- ");
        // create an object:
        GregorianCalendar c = new GregorianCalendar();
        // call getTime method
        System.out.println(c.getTime());
        System.out.println("Finished.");
    }
}
```

Object Creation by "new"

04. How One Object Constructs Another

HANYANG UNIVERSITY

NamdAndData.java

- `import java.util.*;`
  - import: for using pre-defined packages
  - GregorianCalendar ∈ java.util
- `new GregorianCalendar();`
  - “new” creates an object from the GregorianCalendar class
  - We can add some parameters in ()
- `GregorianCalendar c = new GregorianCalendar();`
  - Name a newly created object as “c”
- `c.getTime()`
  - Call the getTime method of the “c” object
- `System.out.println(c.getTime());`
  - Print the result from the method “c.getTime()”

04. How One Object Constructs Another

HANYANG UNIVERSITY

An Execution Trace

① Staring main method

NameAndDate

```
main
{
  System.out.print("Hanyang ICT --- ");
  GregorianCalendar c = new GregorianCalendar();
  System.out.println(c.getTime());
  System.out.println("Finished.");
}
```

System.out

```
print(x)
{
  printing x
}
println(x)
{
  printing x + "\n"
}
```

04. How One Object Constructs Another

HANYANG UNIVERSITY

An Execution Trace

① Staring main method

NameAndDate

```
main
{
  System.out.print("Hanyang ICT --- ");
  GregorianCalendar c = new GregorianCalendar();
  System.out.println(c.getTime());
  System.out.println("Finished.");
}
```

② Object Creation

System.out

```
print(x)
{
  printing x
}
println(x)
{
  printing x + "\n"
}
```

??: GregorianCalendar

```
getTime()
{
  return system time
}
```

04. How One Object Constructs Another

HANYANG UNIVERSITY

An Execution Trace

① Staring main method

NameAndDate

```
main
{
  System.out.print("Hanyang ICT --- ");
  GregorianCalendar c = new GregorianCalendar();
  System.out.println(c.getTime());
  System.out.println("Finished.");
}
```

② Object Creation

③ Name the object as "c"

System.out

```
print(x)
{
  printing x
}
println(x)
{
  printing x + "\n"
}
```

c: GregorianCalendar

```
getTime()
{
  return system time
}
```

04. How One Object Constructs Another

HANYANG UNIVERSITY

An Execution Trace

① Staring main method

NameAndDate

```
main
{
  System.out.print("Hanyang ICT --- ");
  GregorianCalendar c = new GregorianCalendar();
  System.out.println(c.getTime());
  System.out.println("Finished.");
}
```

② Object Creation

③ Name the object as "c"

④ Call getTime

c: GregorianCalendar

```
getTime()
{
  return system time
}
```

System.out

```
print(x)
{
  printing x
}
println(x)
{
  printing x + "\n"
}
```

04. How One Object Constructs Another

HANYANG UNIVERSITY

An Execution Trace

① Staring main method

NameAndDate

```
main
{
  System.out.print("Hanyang ICT --- ");
  GregorianCalendar c = new GregorianCalendar();
  System.out.println(c.getTime());
  System.out.println("Finished.");
}
```

② Object Creation

③ Name the object as "c"

④ Call getTime

c: GregorianCalendar

```
getTime()
{
  return system time
}
```

System.out

```
print(x)
{
  printing x
}
println(x)
{
  printing x + "\n"
}
```

⑤ Call println

Syntax vs. Semantics

- Syntax
  - "Appearance"
  - E.g., new <class name>(<parameters, …>)
  - Syntax error – compiler can find the syntax error
- Semantics
  - "Meaning"
  - E.g., new C() -> An object of the class "C" is created in the memory
  - Semantic error – occur in running
    - Partially detected during run-time, e.g., type error

Java Package

- Classes can belong to a package
  - E.g., in erica.util, MyMap is a class

- Java API (Application Programming Interface)
  - Basic packages
  - E.g., java.lang, java.util
  - Some basic packages do not require “import”, e.g., System.out
  - http://docs.oracle.com/javase/8/docs/api/

Summary

- Class definition: public class Hello { ... }
- Main method: public static void main (String[] args) { ... }
- Methods usage: System.out.println(...), c.getTime()
- Object creation: new GregorianCalendar()
- Name variable: GregorianCalendar c = ...;

Thanks

Week 2: Simple Java Program

Instructor: Jinyoung Han (jinyounghan@hanyang.ac.kr)