

# Object Oriented Programming

Kookmin University

Department of Computer Science

# Announcements

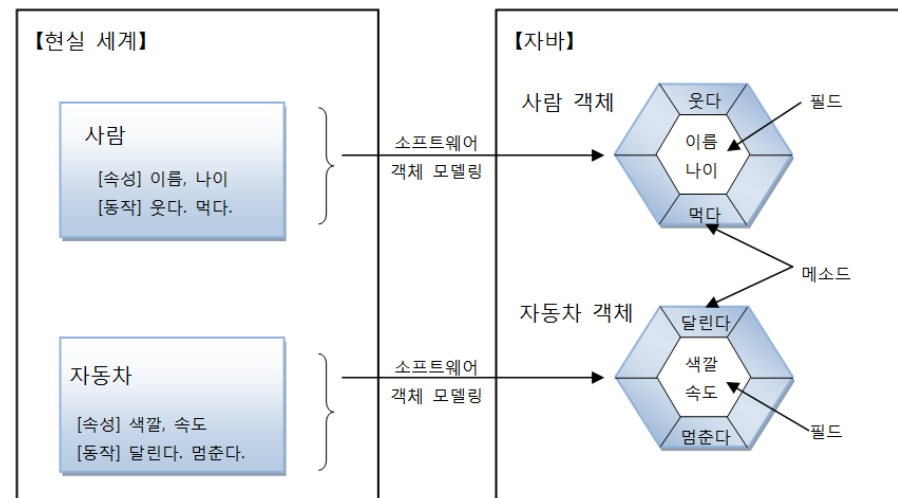
- Midterm exam
  - Oct. 18<sup>th</sup> (**Wednesday**)
  - Location: 7호관 629호
  - Until multi-dimensional array
- A lecture will be given on Oct. 16<sup>th</sup>
  - Homework explanation
- The third homework
  - <https://classroom.github.com/a/hbNWlgTg>
  - Due: Oct. 16<sup>th</sup>
- Review of the previous live coding test

# Object Oriented Programming

- Comparing to a process of making a car
  - Makes components first – engine, tire, door, ...
  - Later, assemble it
  - Some cars shares components – economic and luxurious version
- What is **Object**
  - Things that physically exist – car, book, person
  - Abstract stuffs that have own attributes
    - KMU CS department - # of students, classes, faculty, ...
    - KMU CS department class can be another object – title, timetable, syllabus, ...
- Object Oriented Programming models programming using objects

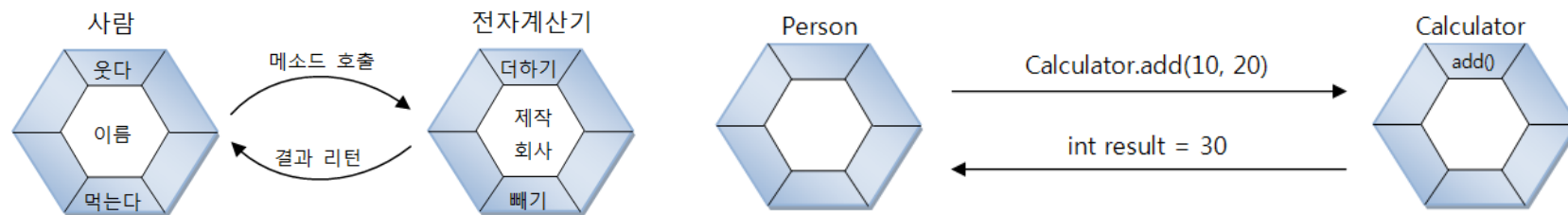
# Object in Java

- Java allows express an object using
  - Field – attributes that express characteristics of an object
    - Person's name, Class hour
  - Method – an action that an object can take
    - Person: walk, class: taking an exam



# Interaction Between Objects

- Each object provides an interface so that other objects can interact with it
  - Method – an interface that an object provide to other projects in order to interact/perform action with an object
  - An output from method is returned to a caller object
  - A way of calling method
    - Object variable name followed by dot and method name/arguments
    - EX: Calculator.add(10, 20);

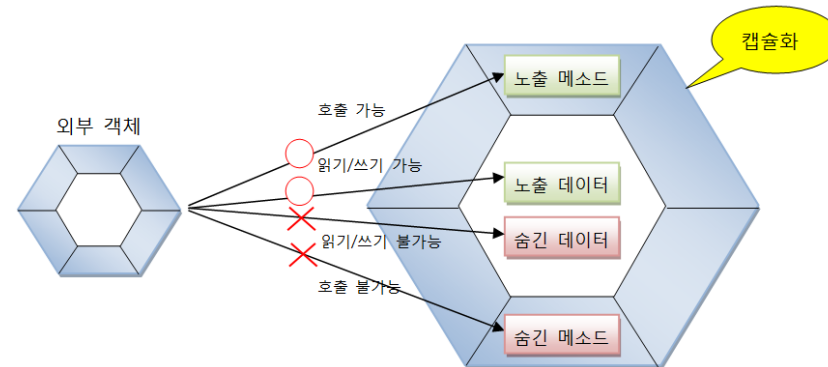


# Method and Arguments

- Argument value (매개값)
  - When calling a method of an object, input data need to be passed to an object that executes the method
  - EX: `Calculator.add(10, 20);`
- Return value
  - After executing a method, the result is returned to caller
  - EX. `int sumResult = Calculator.add(10,20); // sumResult is 30`

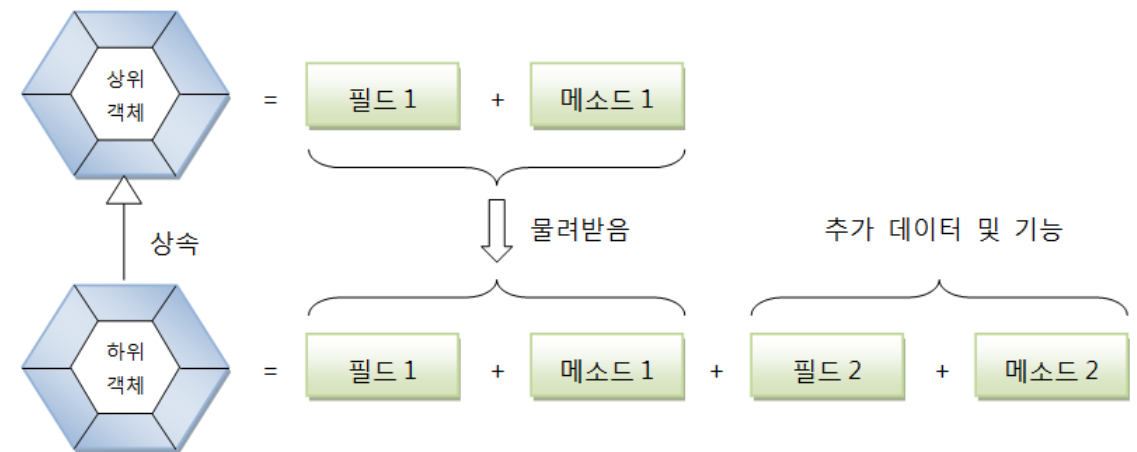
# Characteristics of OOP

- Encapsulation (캡슐화)
  - Hide details of method implementation and field
  - An external object does not know about the details about an method
  - It prevents invalid manipulation of internal data from illegal attempts from external objects
  - Java allows setting a level of exposure by using access modifier
    - Public/Protected/Private



# OOP Characteristics – Inheritance (상속)

- Inheritance in a real-world
  - Parents give legacy to children for free
- In Java programming, the concept is very similar
  - Child class can use methods in parent class for free (without reimplementation)
  - Child class can extend and modify what parent class provides
  - Field and Methods are inherited
- Advantage of Inheritance
  - Remove duplicate code
  - Fast child class implementation
  - Maintenance





# Polymorphism in OOP

- Using various types of objects with a same type
- EX: If a parent class is same, objects created from a child class can be replaced among them
- Advantage
  - Maintenance
  - Each object can be modularized



# Object and Class

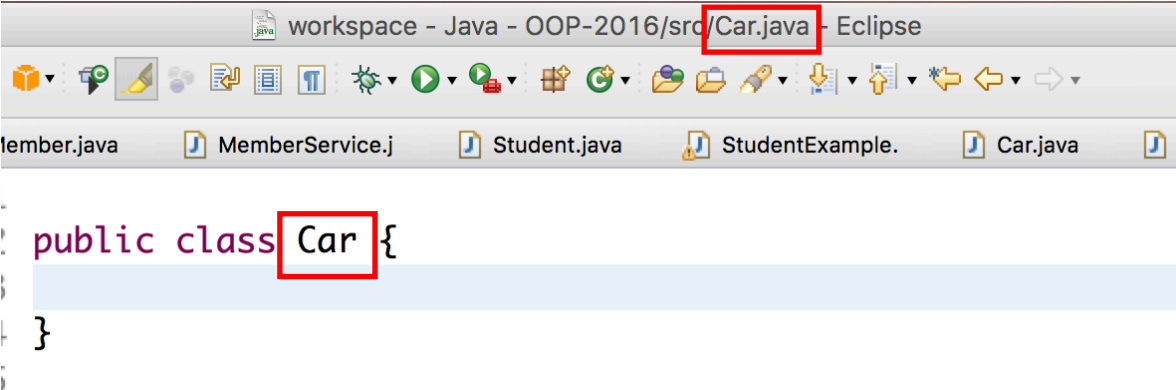
- In Java, an object is created from a class
- A class defines how an object will be shaped
  - Field – defines attributes
  - Method – defines functions
- Instance
  - An object created from a class
- Multiple instances can be created from a class
  - Student class can general many instances per each students

# Sequence of OOP Programming

- Design a class
- Create an instance from the class
- Use the object created from a class
- We will cover how to create a class

# Implementing Class

- Decide class name
  - Encouraged to use English name
    - No number as the first character of class name
    - No special character except \$ and \_
    - No java keyword (int, for, while, ...)
  - Conventional: the first character of a word is capital (WebBrowser)
- Class definition and compile
  - One class in one file
  - Access modifier class class\_name
    - public class Car {...}
  - Code file name: class\_name.java
  - Car.java → public class Car {...}



```
workspace - Java - OOP-2016/src/Car.java - Eclipse

member.java  MemberService.j  Student.java  StudentExample.  Car.java

public class Car {
}
```

# Creating Object from Class

- If an object is created from a class definition, it is stored in the heap region, and the stack region stores the address to the instance (reference type variable)
- new keyword is used to create an instance
  - `Car myCar = new Car();`

# Class Composition

- A class is composed of
  - Constructor – called once in the initialization (setting age to 0)
  - Field – stores data of a class (name, kmu-id, ...)
  - Method – defines an action for a class (eat, walk, ...)

```
public class Car {  
    Car() {  
  
    }  
  
    String modelName;  
  
    void drive() {  
  
    }  
}
```

# Class Field

- A field in a class stores data or the status of an object
- EX: Car object
  - General data – manufactured company, model, color
  - Status – current speed, gas level
  - Components – Car body, engine, tire
- Field needs to be defined within {} of a class – not in method and constructor scope
- Declaration format is very similar to variable declaration
  - Syntax: type name = initialization
  - EX: String company = "Hyundai Motor";

# Accessing Object Field

- An object field can be accessed only when an object is created from a class definition
- The object field is accessed by using a variable name followed by dot and field name
  - `Car myCar = new Car();`
  - `myCar.name`



# Class Method

- Actions that can be executed using class fields
- Method consists of
  - Return type
    - Decides a type of outcome from execution of a method (code body)
  - Method name
    - A name of method where a programmer can reference the method
  - Arguments
    - Passes data to be executed in a method code
  - Code body
    - A set of codes that is executed when a method is called
- Method signature
  - Return type + Method name + Arguments

# Class Method Example

- Return type
  - int (the output is an age of a car)
- Method name
  - GetAge : calculate age of a car
- Argument
  - int currentYear: Using current year, we can calculate the age of a car
- Code body
  - Calculate the age of a car

```
int modelYear = 2006;  
  
int GetAge(int currentYear) {  
    int age = currentYear - modelYear;  
    return age;  
}
```

# Method Signature

- Return type + Method name + Arguments
- Return type
  - A variable type of outcome from the method
  - It can primitive variables/reference type variable (String, array, class objects)
  - If it does not return any outcome, a keyword “void” should be used as a return type

```
void StartEngine() {  
    .....  
}
```

# Method Name

- Method name should be reasonable to understand what the method does
- It should not start with number
- No special characters except \$ and \_
- By convention
  - The method name should be lowercase letters
  - For consecutive words, use capital for a new word (Camel case)

# Method Arguments

- Arguments allow to pass data from the method caller to the method itself
- Arguments can be omitted (input data is not necessary)
- Each argument should contain its type and name that can be accessed within a method body
- Multiple arguments are separated by commas
- Array can be passed as an argument
  - If the length of an array is not known at the time of declaration, we can use "...": `int sum(int ... inputs)`
- Calculator exercise

# References

- 이것이자바다 – 한빛미디어 2015