Object Oriented Programming

Kookmin University

Department of Computer Science

Announcements

- Midterm exam score is posted in your private-notification repository
  - github.com/leeky-courses-2017-02/oop-private-notifications-YOUR_ID
  - Students without the repository
  - https://classroom.github.com/a/PjjgMfCM
- The second live coding test is scheduled on the week 10 Wednesday – Nov. 1st
  - The third one going to happen on the week 13

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 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”;

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

Method Overloading

- Overloading
  - Multiple methods with a same name can be defined with different arguments
  - One of argument types, the number of arguments, the order should be different
  - Return type does not matter
  - It allows reusing a method name with different types

```
int plus(int x, int y) {
    int result = x + y;
    System.out.println("Two int variables sum");
    return result;
}

double plus(double x, double y) {
    double result = x + y;
    System.out.println("Two double variables sum");
    return result;
}
```

```
int sum1 = calculator.plus(3, 10);
double sum2 = calculator.plus(2.5, 5.1);
```

Good Example of Method Overloading

- System.out.println()
  - It receives different types of input variables and print out the value

```
System.out.println("String print");
System.out.println(12345);
System.out.println(12345.678);
System.out.println(true);
```

```
int getRectangleArea(int width, int height) {
    return width * height;
}

int getRectangleArea(int width) {
    return width * width;
}
```

Class Constructor

- A constructor is executed when a ”new” keyword is called
- It is responsible for initialization of an object
  - Field value initialization
  - Calling methods to create another object
- Upon successful execution of constructor, an object is created in a heap region and the address is stored in a stack region
- Default constructor
  - If a constructor is not declared, a default constructor (empty body with no arguments) is executed by default

Constructor Definition

```
public class Car {
    public Car() {}
}
```

- The name of constructor method should follow class name
- In order to set a custom constructor (arguments and body), a constructor method (name with a class name) can be declared

```
public class CarConstructor {
    public CarConstructor(){}
    public CarConstructor(String model, String company, int year) {
        modelName = model;
        manufacturer = company;
        modelYear = year;
    }
    String modelName = "apollo";
    String manufacturer = "KMU";
    int modelYear = 2006;
```

Constructor Overloading

- Similar to method overloading (having multiple methods with a same name but with different arguments), class constructor can also be overloaded
  - Either number of arguments, variable types, or order of variable types should be different

Instance Member

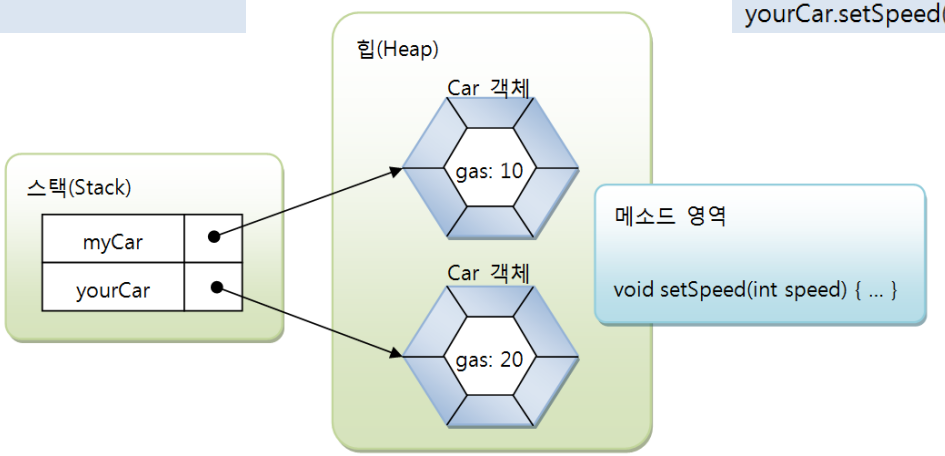
- Fields and Methods that belong to an object
- They can be accessed only through the object variable
  - First create an instance from a class definition (using new keyword)
  - Upon creation of an object, it is located in Heap region

Method Memory Region

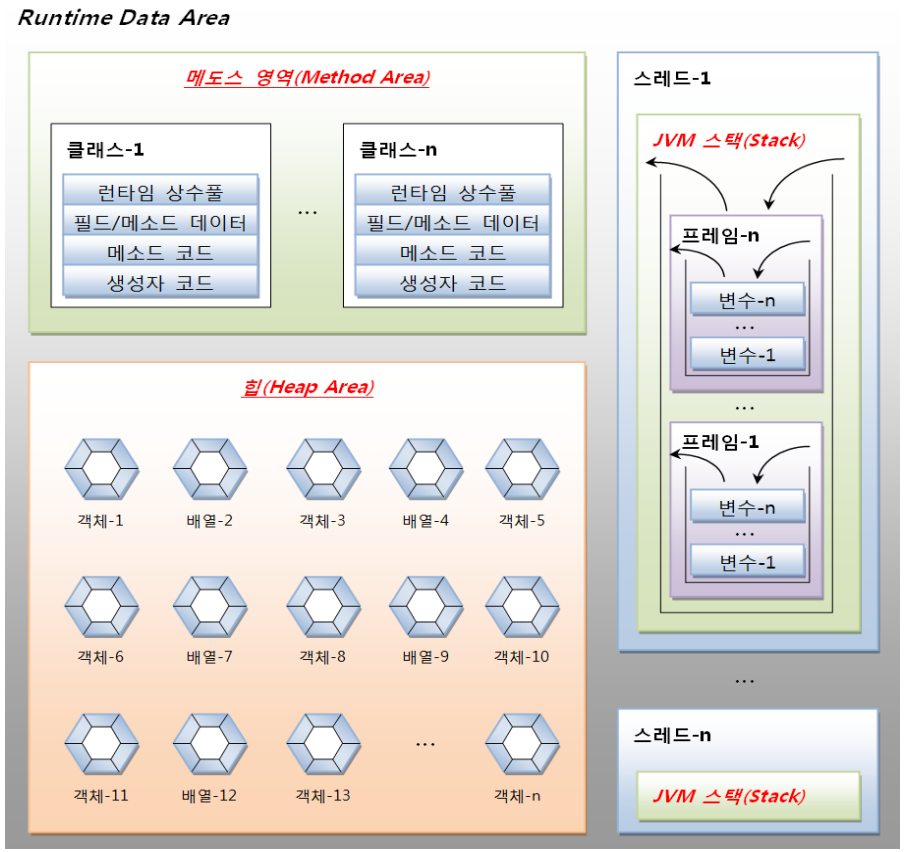
- Method definition in a class is same regardless objects

```
public class Car {
   //필드
   int gas;

   //메소드
   void setSpeed(int speed) { ... }
}
```

```
Car myCar = new Car();
myCar.gas = 10;
myCar.setSpeed(60);

Car yourCar = new Car();
yourCar.gas = 20;
yourCar.setSpeed(80);
```



이것이자바다 P. 234



이것이자바다 P. 140

Access Instance Using this Keyword

- Outside of an object, fields/methods should be accessed by variable name + dot + names
  - EX: myCar.model, myCar.startEngine()
- Within an object itself, it can access own fields/method with this keyword
  - EX. this.model, this.startEngine()
  - Beneficiary when the argument variable name and object field/method name is the same
- In a class constructor, it is used to call another constructor with different arguments (overloading)

This Keyword Example

```
public class InstanceThis {
    String model;
    int speed;

    InstanceThis(String model) {
        this.model = model;
    }

    void setSpeed(int speed) {
        this.speed = speed;
    }
}
```