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

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);
```

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 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

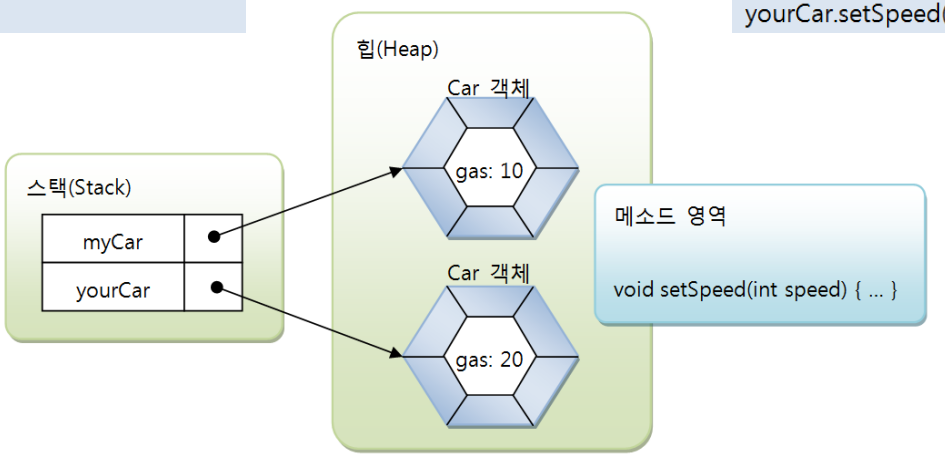
- Fields and Methods that belong to an object
- They can be accessed only through the object variable
  - First create an instance from a class definition (using new keyword)
  - Upon creation of an object, it is located in Heap region

Method Memory Region

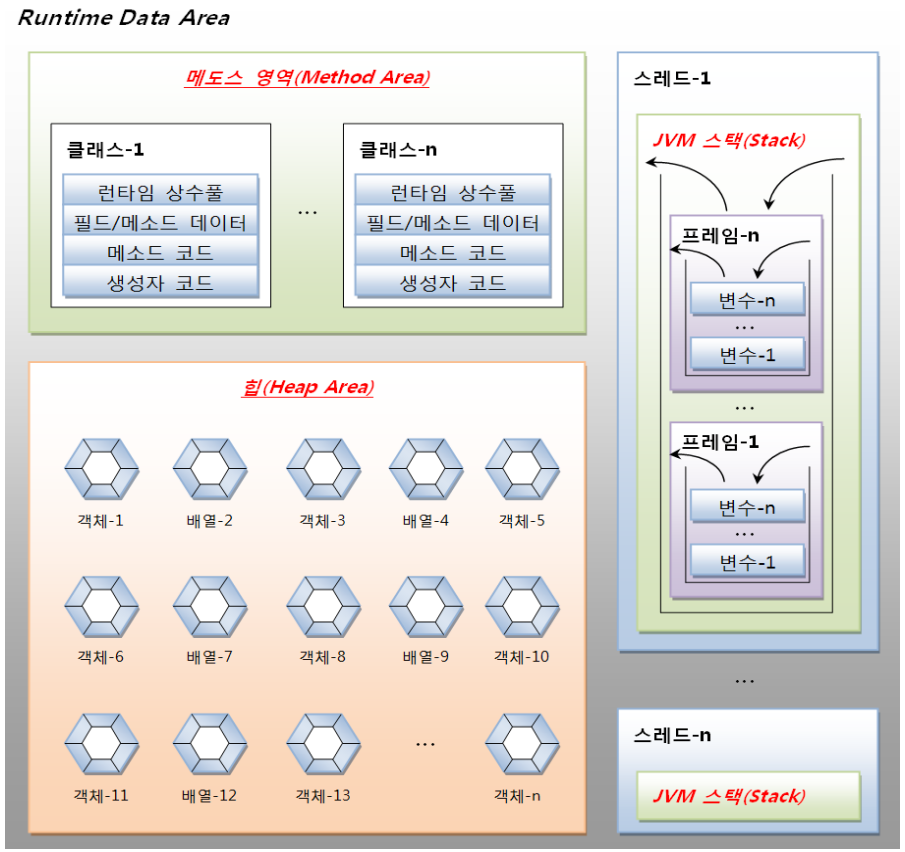
- Method definition in a class is same regardless objects

```
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;
    }
}
```

Java Package

- In reality, a Java application is composed of 100s ~ 1000s of classes that is really hard to manage well in the unit of class
- As we use folder (directory) to better manage multiple files, we use package to manage multiple classes
- Directory is managed in hierarchy
  - Package is also managed in hierarchy
- A same file name can exist in different directories
  - A same class name can exist in different packages

Decide Package Name

- Similar to directory name, a package name is shaped in a hierarchical way
  - Separator of directory : ₩ (Windows) or / (Unix)
  - Separator of package : .
- Generally recommend to use the organization's URL as the package name hierarchy (URL in reverse order)
  - org.apache.
  - kr.ac.kookmin.cs.oop.project
- It is recommend to have the class directory hierarchy same as that of package name

Package Import

- Classes in a same package can use other classes
- To use classes in a different package, we have to declare “import” command in the beginning of a code
  - After package name declaration
  - Before starting class declaration
- In importing a package, classes in lower layer package are not imported
- Package exercise
  - kr.ac.kookmin.cs.calculus and oop with Class – ImportExercise

Static Member

- Meaning of static
  - Not changing (정적)
  - Antonym of dynamic
- In Java, static members belongs to a class, but they can be used without creating an object
- Static field and static method
- Object fields are stored in Heap
- Static fields are stored in the method area
  - Stack, Heap, Method area

Static Member or Instance Member?

- When declaring field/method, do we have to use static or instance member?
- If a variable value has to be different across different objects, it should be declared as instance member
- If a variable value is same across all the object, it can be static member
  - Note the space efficiency for the static member
  - Static member is created only once

Static Method or Instance Method?

- Similar to fields, if a method uses instance fields, declare it as an instance method
- If instance fields are not referenced, declare it as static method

Using Static Member

```
public class StaticMember {
    String color;
    static double pi = 3.14159;

    static int plus(int x, int y) {
        return x + y;
    }

    static int minus(int x, int y) {
        return x - y;
    }

    void setColor(String color) {
        this.color = color;
    }

    String getColor() {
        return color;
    }
}
```

Accessing Static Member

- In order to access static member, it is recommended to use Class Name followed by dot and the member name
  - Though one can use an object variable name directly (not recommended)

```
public class StaticMember {
    String color;
    static double pi = 3.14159;
```

```
StaticMember staticExercise = new StaticMember();
double sizeOfCircle = StaticMember.pi * radius * radius; // recommended
sizeOfCircle = staticExercise.pi * radius * radius; // not recommended
```

Cautions when Using Static Members

- Using instance member in a static method is not feasible
  - Note that there is only one static method but instance is not
  - If one wants to use object, it can be created within a static method

```
public class StaticMember {
    String color;
    static double pi = 3.14159;

    public static void main(String[] args) {
        StaticMember staticExercise = new StaticMember();
        System.out.println(this.color);
        System.out.println(staticExercise.color);
```

Necessities to Declare as static

- Declare field/methods as static whenever you can
- It will not waste your memory space by allocating unnecessary members multiple times
- It lets programmer to understand a function does not use any instance member within a method

References

- 이것이자바다 – 한빛미디어 2015