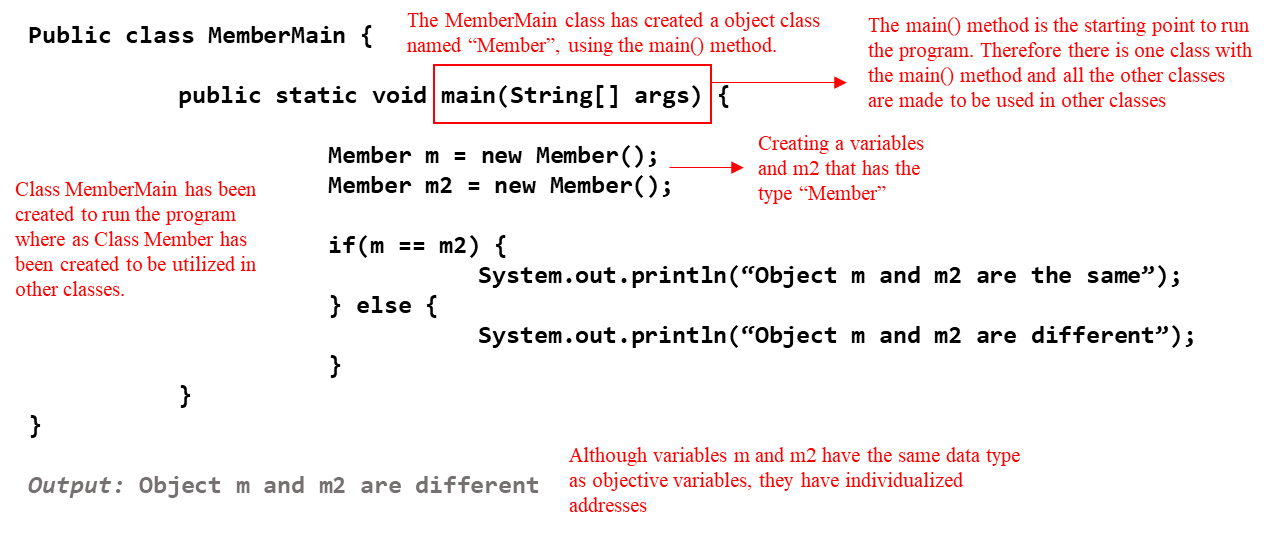
**- OBJECT ORIENTATED PROGRAMMING (객체지항) –**

A programming paradigm that relies on the concept of classes and objects.

**4 fundamental concepts of OOP:**

1. **Inheritance (상속):** When the child class inherits the properties and functions of the adult class, avoiding the need to copy the same codes. Saves maintenance and developing time.
2. **Encapsulation (캡슐화):** A mechanism of concealing the data (variables) as a single unit which are made accessible according to permissions (access modifier).
3. **Polymorphism (다형성)**: The ability of an object to take many forms and therefore perform the same action in many different ways.
4. **Data abstraction (추상화):** The process of reducing the object to its essence so that only the necessary characteristics are exposed to the users. Abstraction defines an object in terms of its properties (attributes), behaviour (methods), and interfaces (means of communicating with other objects).



\

|  |
| --- |
| **Package:** It is a concept used to bind classes that can functionally affect multiple classes in programming, and to effectively call them within the access range. Java(JDK) have already constructed classes (approx. 3000) and it can be found here:  <https://docs.oracle.com/javase/8/docs/api/index.html>   * Importing: If you want to create a package and then use classes from different packages after you create a class, you must use the import keyword. * Classes in the same package do not require import. |

|  |
| --- |
| **Object:** Consists of data and method of the same type |

1. **CLASS**

The process of using a ***class*** (the plan - 설계도) for the ***objec****t* that is created using this class,(creation – 피조물) is called a ***instance*** (클래스 이용해 객체 생성).

* Just as two cars of the same model, share the same plans and parts but is still two ‘difference cars’, two objects (in java) that share the same class are considered two difference objects. A new class is made using:

**ClassName variableName = new ClassName();**

* 1. **FIELD (필드):** Place where the object’s characteristics or specific values are saved. It is the variable within the class.
* A class variable is defined within the block of the class by using the word ‘static’ in front of the variable’s type. (This variable is shared with the entire object).

**ClassName.ClassVariableName**

* An instance variable is also defined within the block of the class but the word static is not added. Since each object has its own address, each object has its own value for this variable

**ObjectName.InstanceVairableName**

|  |  |
| --- | --- |
| EXAMPLE | |
| **Class ‘Andante’ under main class** |  |
| **Main**  **class** |  |

1. **CONSTRUCTOR (생성자):** A method with **no return value** which every class **NEEDS** to have one as default.
   * + - Its name and the class name need to be equivalent.
       - It can have several types of constructors: constructors without a parameter or with one or more parameters.
       - If there is no constructor, the default constructor is automatically created at the compiler stage.

Member (), that is added after the ‘new’ operator is a constructor

**Member member = new Member();**

|  |  |
| --- | --- |
| EXAMPLE | |
| **Method class** |  |
| **Main class** |  |
| **Result** | **Anna**  **4**  **Economics** |

1. **METHOD (메서드):** A logic that is created separately. If given the appropriate data it will return the result (It is an action that an object is able to perform).

* If the method has a return value, after the executable command, must return a value.
* If the method has no return value, before the method name, must write void (**void MethodName()**)**.**

**Void methodName(int … x) {**

**int sum = 0;**

**For (int i=0 ; i<x.length ; i++) {**

**Sum == x[i];**

* If the method starts with ‘static’, use this method without making an object:

|  |  |
| --- | --- |
| EXAMPLE (with static) | |
| **Method class** | **Static void printA() {**  **System.out.println(“A”)** |
| **Main class** | **Method.printName()** |
| **Result** | **A** |

|  |  |
| --- | --- |
| EXAMPLE2 (without static) | |
| **Method class** | **void printB() {**  **System.out.println(“B”)** |
| **Main class** | **Method m = new Method();**  **m.printB();** |
| **Result** | **B** |

|  |
| --- |
| **Overloading:** When there are multiple methods of the same name **within the same class** with at least one of the following different: method variable’s type, its count or order. |

|  |
| --- |
| **Overriding:** Saving multiple methods of the same name **in another class**. |

**Class Operator {**

**int multiply(intx, int y) {**

**Return x \* y;**

Printing 4 types of the “multiply” method that have different object variable types

**}**

**Double multiply (double x, double y) {**

**Return x \* y;**

**}**

**Double multiply (int x, double y) {**

**Return x \* y;**

|  |
| --- |
| **this:** Keyword used when using an operator of the same class. It is also used as an object of a reference variable.   * “this” is used to distinguish a declared variable with an instance variable and save the instance variable, when they share the same name, within the same method * by adding a “this”, the program looks for the variable made in its current class, and then if not found, it looks for the variable in the parent class. |



|  |
| --- |
| **static:** Used when the value is supposed to be shared across all objects   * **Static (public) class variables:** Variable that belongs to a type itself, rather than to an object (instance) of that type. We can access them directly using class name and without object initialization. (E.g. Mom pouch) * **Static methods:** Used to perform an operation that is not dependent upon instance creation and can be called without creating the object of the class in which they reside. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Access modifier:** Can be specified for Class, Method/constructor and variables | **Access Modifier Range** | | | |
| **All the Classes** | **Inheritance Relationship** | **Same Package** | **Same Class** |
| **Public** | ✓ | ✓ | ✓ | ✓ |
| **Protected**  (Available only when the class is in the same package as the class or is an inherited class) | ｘ | ✓ | ✓ | ✓ |
| **(Default) –** If an access modifier is not stated, then the program reads it as default | ｘ | ｘ | ✓ | ✓ |
| **Private** – Only within same class | ｘ | ｘ | ｘ | ✓ |