자바 클래스2

자바 생성자

자바 생성자

- Java의 생성자는 객체를 초기화하는 데 사용되는 특수 메서드 입니다.
- 생성자는 클래스의 객체가 생성될 때 호출됩니다.
- 객체 속성의 초기 값을 설정하는 데 사용할 수 있습니다.

```
public class Main {
  int x;  // 클래스의 속성 x 생성

  // 생성자 생성
  public Main() {
    x = 5;  // 속성 x 5로 초기화
  }

  public static void main(String[] args) {
    Main myObj = new Main(); // Create an object of class Main (This will call the
constructor)
    System.out.println(myObj.x); // Print the value of x
  }
}

결과 : 5
```

- 생성자 이름은 클래스 이름과 일치 해야 하며 반환 유형을 가질 수 없습니다
- 또한 객체가 생성될 때 생성자가 호출된다는 점에 유의하세요.
- 모든 클래스에는 기본적으로 생성자가 있습니다.
- 클래스 생성자를 직접 생성하지 않으면 Java가 생성자를 생성합니다.
- 그러나 객체 속성의 초기 값을 설정할 수 없습니다.

생성자 매개변수

> 생성자는 속성을 초기화하는 데 사용되는 매개변수를 사용할 수도 있습니다.

다음 예제에서는 생성자에 int y 매개변수를 추가합니다. 생성자 내에서 x를 y(x=y)로 설정합니다. 생성자를 호출할 때 x 값을 5로 설정 생성자(5)에 매개 변수를 전달합니다.

```
public class Main {
  int x;  // x = 5

  public Main(int y) {  // 매개변수가 있는 생성자 , y=5
    x = y;    // x = y 초기화 , x = 5
  }
```

```
  public static void main(String[] args) {
    Main myObj = new Main(5);  // 객체 생성하며 Main() 호출 , y = 5
    System.out.println(myObj.x);  // 5
  }
}
```

결과 : 5

> 원하는 만큼 매개변수를 가질 수 있습니다.

```
public class Main {
  int modelYear;
  String modelName;

  public Main(int year, String name) {
    modelYear = year;
    modelName = name;
  }

  public static void main(String[] args) {
    Main myCar = new Main(1969, "Mustang");
    System.out.println(myCar.modelYear + " " + myCar.modelName);
  }
}
```

공백

결과 : 1969 Mustang

수정자 접근자

수정자

거의 모든 예제에 나타나는 public키워드에 대해 꽤 잘 알고 계실 것입니다

```
public class Main
```

- public 은 가장 일반적인 접근 제어자입니다.
- public 으로 선언된 멤버 변수, 메소드는 어떠한 자바 프로그램에서든지 제한 없이 사용이 가능합니다

수정자를 두 그룹으로 나눕니다

- 접근 수정자 - 접근 수준을 제어합니다.
- 비접근 수정자 - 접근 수준을 제어하지 않지만 다른 기능을 제공합니다.

접근 수정자

> 클래스의 경우 `public` 또는 `default`를 사용할 수 있습니다.

- public : 해당 클래스는 다른 클래스에서 액세스할 수 있습니다.

- default : 클래스는 동일한 패키지에 있는 클래스에서만 액세스할 수 있습니다. 수정자를 지정하지 않을 때 사용됩니다.

> 속성, 메소드 및 생성자 의 경우 다음 중 하나를 사용할 수 있습니다.

- public : 모든 클래스에 접근할 수 있습니다
- private : 선언된 클래스 내에서만 액세스할 수 있습니다.
- default : 동일한 패키지에서만 액세스할 수 있습니다.수정자를 지정하지 않을 때 사용됩니다.
- protected : 동일한 패키지 및 하위 클래스에서 액세스할 수 있습니다.

비접근 수정자

> 클래스 의 경우 `final` 또는 `abstract`를 사용할 수 있습니다 .

- final: 상속할 수 없습니다
- abstract: 객체를 생성할 수 없습니다, 추상 클래스에 접근하려면 다른 클래스에서 상속받아야 합니다.

> 속성 및 메소드 의 경우 다음 중 하나를 사용할 수 있습니다.

- final : 속성과 메소드는 재정의/수정될 수 없습니다.
- static : 속성과 메소드는 객체가 아닌 클래스에 속합니다.
- abstract : 추상 클래스에서만 사용할 수 있으며 메서드에서만 사용할 수 있습니다. 메서드에 본문이 없습니다, 본문은 하위 클래스(상속됨)에 의해 제공됩니다.
- transient :
- synchronized :
- volatile :

final

> 기존 속성 값을 재정의하는 기능을 원하지 않으면 final속성을 다음과 같이 선언하세요.

```
public class Main {
  final int x = 10;
  final double PI = 3.14;

  public static void main(String[] args) {
    Main myObj = new Main();
    myObj.x = 50; // error
    myObj.PI = 25; // error
    System.out.println(myObj.x); // error
  }
}
```

static

> 클래스의 객체를 생성하지 않고도 액세스할 수 있음을 의미합니다

static와 public 방법의 차이점을 보여주는 예 :

```
public class Main {
  // Static method
  static void myStaticMethod() {
    System.out.println("Static 메소드를 호출해서 생성");
  }

  // Public method
  public void myPublicMethod() {
    System.out.println("Public 메소드를 호출해서 생성");
  }

  // Main method
  public static void main(String[ ] args) {
    myStaticMethod(); // static 메소드 호출
    // myPublicMethod(); error

    Main myObj = new Main(); // Main 클래스를 사용하여 myObj 객체 생성
    myObj.myPublicMethod(); // public 메소드 호출
  }
}

결과:
Static 메소드를 호출해서 생성
Public 메소드를 호출해서 생성
```

추상적인

abstract 메소드는 abstract에 클래스 속하며 본문이 없습니다. 본문은 자식클래스에서 제공됩니다.

```
// Main.java
// 추상 클래스 abstract class
abstract class Main {
  public String fname = "홍길동";
  public int age = 24;
  public abstract void study(); // 추상 메소드 abstract method
}

// 하위클래스 ( Main 함수를 상속받음)
class Student extends Main {
  public int graduationYear = 2025;
  public void study() { // 추상메소드를 여기서 생성, 재정의
    System.out.println("웹공부는 평생~~!!");
  }
}
// End Main.java

// Second.java
class Second {
  public static void main(String[] args) {
    Student myObj = new Student();
   // myObj객체를 생성하면서 Student 클래스와 상속받은 Main클래스를 다 사용
```

```
    System.out.println("이름: " + myObj.fname);
    System.out.println("나이: " + myObj.age);
    System.out.println("졸업년도: " + myObj.graduationYear);
    myObj.study(); // 호출
  }
}

결과 :
이름: 홍길동
나이: 24
졸업년도: 2025
웹공부는 평생~~!!
```

자바 캡슐화

> 캡슐화의 의미는 "민감한" 데이터가 사용자에게 숨겨지도록 하는 것입니다.

- 클래스 변수/속성을 다음과 같이 private 선언합니다.
- get 및 set 메소드 private 변수 값에 접근하고 업데이트하기 제공

가져오기 및 설정

- private변수는 동일한 클래스 내에서만 접근할 수 있다는 것을 배웠습니다(외부 클래스는 변수에 접근할 수 없습니다).
- 그러나 공개 get 및 set 메소드를 제공하면 접근이 가능합니다 .
- 메서드 get는 변수 값을 반환하고 set메서드는 값을 설정합니다.

get 또는 set 둘 다의 구문은 로 시작하고 그 뒤에 변수 이름이 오고 첫 글자는 대문자입니다.

```
public class Person {
  private String name; // private = 제한된 접근

  // Getter
  public String getName() {
    return name;
  }

  // Setter
  public void setName(String newName) {
    this.name = newName;
  }
}
```

get메소드는 name변수의 값을 반환합니다 .

set메서드는 매개변수(newName)를 가져와 이를 name변수에 할당합니다. 키워드 this는 현재 개체를 참조하는 데 사용됩니다.

그러나 name변수가 private로 선언되었으므로 이 클래스 외부에서는 해당 변수에 액세스 할 수 없습니다 .

```
public class Main {
  public static void main(String[] args) {
    Person myObj = new Person();
    myObj.name = "홍길동";  // error
    System.out.println(myObj.name); // error
  }
}
```

> private 변수에 접근하려고 하면 다음과 같은 오류가 발생합니다.

```
MyClass.java:4: error: name has private access in Person
    myObj.name = "홍길동";
         ^
MyClass.java:5: error: name has private access in Person
    System.out.println(myObj.name);
                       ^
2 errors
```

> 대신 getName()및 setName()메소드를 사용하여 변수에 액세스하고 업데이트합니다.

```
public class Main {
  public static void main(String[] args) {
    Person myObj = new Person();
    myObj.setName("홍길동"); // name 변수의 값을 "홍길동"으로 설정합니다.
    System.out.println(myObj.getName());
  }
}

결과 : 홍길동
```

왜 캡슐화하는가?

- 클래스 속성 및 메소드에 대한 더 나은 제어
- 클래스 속성은 읽기 전용 (메서드만 사용하는 경우 get) 으로 설정할 수 있습니다.
- 쓰기 전용 (메서드만 사용하는 경우 set) 으로 설정할 수 있습니다.
- 유연성: 프로그래머는 다른 부분에 영향을 주지 않고 코드의 한 부분을 변경할 수 있습니다.
- 데이터 보안 강화

자바 패키지/API

자바 패키지

Java의 패키지는 관련 클래스를 그룹화하는 데 사용됩니다. 파일 디렉토리의 폴더로 생각하십시오 . 우리는 이름 충돌을 피하고 더 나은 유지 관리가 가능한 코드를 작성하기 위해 패키지를 사용합니다. 패키지는 두 가지 범주로 나뉩니다.

- 내장 패키지(Java API의 패키지)
- 사용자 정의 패키지(자신만의 패키지 생성)

내장 패키지

Java API는 Java 개발 환경에 포함되어 무료로 사용할 수 있는 미리 작성된 클래스 라이브러리입니다.

라이브러리에는 입력 관리, 데이터베이스 프로그래밍 등을 위한 구성 요소가 포함되어 있습니다. 전체 목록은 Oracle 웹사이트(https://docs.oracle.com/javase/8/docs/api/)에서 확인할 수 있습니다 .

라이브러리는 패키지와 클래스로 구분됩니다 . 즉,단일 클래스(메소드 및 속성과 함께)를 가져오거나 지정된 패키지에 속하는 모든 클래스를 포함하는 전체 패키지를 가져올 수 있습니다.

라이브러리의 클래스나 패키지를 사용하려면 다음 import키워드를 사용해야 합니다 .

```
import package.name.Class;   // Import a single class
import package.name.*;   // Import the whole package
```

클래스 가져오기

예를 들어 사용자 입력을 얻는 데 사용되는Scanner 클래스와 같이 사용하려는 클래스를 찾으면 다음 코드를 작성하십시오.

```
import java.util.Scanner;
```

위의 예에서 java.util는 패키지이고 Scanner는 java.util 패키지의 클래스입니다.

클래스를 사용하려면 클래스의 객체를 생성하고 Scanner 클래스 문서에 있는 사용 가능한 메서드 중 하나를 사용합니다. 이 예에서는 nextLine()전체 줄을 읽는 데 사용되는 메서드를 사용합니다.

```
import java.util.Scanner;

class MyClass {
  public static void main(String[] args) {
    Scanner myObj = new Scanner(System.in);
    System.out.println("이름 입력");

    String userName = myObj.nextLine();
    System.out.println("이름은: " + userName);
  }
}
```

패키지 가져오기

선택할 수 있는 패키지가 많이 있습니다. 이전 예제에서는 패키지 Scanner의 클래스를 사용했습니다 java.util. 이 패키지에는 날짜 및 시간 기능, 난수 생성기 및 기타 유틸리티 클래스도 포함되어 있습니다.

전체 패키지를 가져오려면 문장을 별표(*)로 끝냅니다. 다음 예에서는 java.util패키지의 모든 클래스를 가져옵니다.

```
import java.util.*;
```

사용자 정의 패키지

자신만의 패키지를 만들려면 Java가 파일 시스템 디렉터리를 사용하여 패키지를 저장한다는 점을 이해해야 합니다.

컴퓨터의 폴더와 마찬가지로:

```
└── root
  └── mypack
    └── MyPackageClass.java
```

패키지를 생성하려면 package키워드를 사용하십시오.

```
MyPackageClass.java

package mypack;
class MyPackageClass {
  public static void main(String[] args) {
    System.out.println("This is my package!");
  }
}

결과 :
This is my package!
```

파일을 MyPackageClass.java 로 저장 하고 컴파일합니다.

```
C:\Users\Your Name>javac MyPackageClass.java
```

그런 다음 패키지를 컴파일합니다.

```
C:\Users\Your Name>javac -d . MyPackageClass.java
```

이렇게 하면 컴파일러가 "mypack" 패키지를 생성하게 됩니다.

키워드 -d는 클래스 파일을 저장할 대상을 지정합니다

참고: 패키지 이름은 클래스 이름과 충돌하지 않도록 소문자로 작성해야 합니다.
위의 예에서 패키지를 컴파일하면 "mypack"이라는 새 폴더가 생성되었습니다.

MyPackageClass.java 파일을 실행하려면 다음을 작성하십시오.

```
C:\Users\Your Name>java mypack.MyPackageClass
```

출력

```
This is my package!
```