# class, 문자열, 배열

yksoon

2020 4 7

## 1. 한 개 파일에 하나의 메인 클래스만 선언 할 경우

- A.java

```
public class A{         // main 클래스
    public A(){
        }
}
```

## 2. 한 개 파일에 메인 클래스와 서브 클래스만 선언 할 경우

- B.java

```
public class B{         // main 클래스
    public B(){
    }
}
class B_sub{                // sub 클래스 (publlic을 붙일 수 없다)
    public B_sub(){
    }
}
```

## 3. 한 개 파일에 메인 클래스와 이너 클래스, 서브 클래스만 선언 할 경우

- C.java

```
public class C{         // main 클래스
    public C(){
    }
    class C_inner{      // inner 클래스
    }
}
class C_sub{                // sub 클래스 (publlic을 붙일 수 없다)
    public C_sub(){
    }
}
```

# 4. 한 개 파일에 메인 클래스와 메서드 클래스, 이너 클래스, 서브 클래스만 선언 할 경우

- D.java

```
public class D{          // main 클래스
    public D(){
    }
    class D_inner{           // inner 클래스
    }
    public void method(){    // 메서드
        class D_method{      // method 클래스
        }
    }
}
class D_sub{                     // sub 클래스 (publlic을 붙일 수 없다)
    public D_sub(){
    }
}
```

# 선언의 예

- TestClass .java

```
// Main Class
public class TestClass {
  public TestClass() {
    System.out.println("Test 생성자 내부");
  }
}
// Inner Class
class Test_inner{   // public 사용 선택
  public Test_inner(){
    System.out.println("Test_inner 생성자 내부");
  }
}

public void method(){
  // inner method Class
  class Test_method{   // public 사용 불가
    public Test_method(){
      System.out.println("Test_method 생성자 내부");
    }
  }
    Test_method t1=new Test_method(){
    }
}

// Sub Class
class Test_sub{   // public 사용 불가
  public Test_sub(){
    System.out.println("Test_sub 생성자 내부");
  }
}
```

# 선언된 클래스 사용의 예

- import com.kkh.array.TestClass.Test_inner;

```
public class MainClass {

  public static void main(String[] args) {
    TestClass t=new TestClass();           // Main Class 객체 생성
    Test_sub t_sub=new Test_sub();        // Sub Class  객체 생성
    Test_inner t_inner=t. new Test_inner();  // Inner Class 객체 생성
    t.method();                                 // 메서드를 호출함으로써 Mathod Class 객체 생성
  }
}



console 결과
Test 생성자 내부
Test_sub 생성자 내부
Test_inner 생성자 내부
Test_method 생성자 내부
```

- private 는 인자나 변수 등이 한 클래스 내에서만 사용가능
- String.substring ( ) 문자열을 인덱스처럼 인식하여 추출하는 방법
    - ex) String s = “Hello”;
    - s.substring (0, 2) => “He”
- Object 로 업캐스팅시 모든 타입을 받고 묶을수 있지만
  클래스 안에서 선언된 변수나 데이터를 사용하려면 다시 다운캐스팅을 해주어야 함.

---

- 배열(array) : 주로 여러개의 값을 저장할 때.
- 배열의 특징 : 동일한 데이터 타입만 저장 가능. -초기 index의 갯수는 수정 불가능. -초기에 index(방)의 갯수를 설정. -예) TestClass[] tArr = new TestClass[3];

---

---

# ArrayList : 외부클래스로 존재(ArrayList.class)

- 반드시 import 가 필요 (import java.util.ArrayList)
- 내부 메서드를 통하여 갯수가 정해지고, 수정 가능
- 따라서 객체 생성이 필수 (new ArrayList() )
- ArrayList는 주로 객체들을 저장할 때 사용.
- 다양한 타입의 데이터를 보관할 수 있다. (Object 타입으로 저장되기 때문에)

예) ArrayList a = new ArrayList();
a.add(new TestClass);
a.add(true);
a.add(1123);
a.add(12.5);
a.add(“문자열”);
a.add(‘abcd’);

단, 저장된 값은 Object 타입으로 저장된다.
따라서 데이터를 사용할 경우, 원래 타입으로 변환 해야 한다.
예) TestClass t = (TestClass)a.get(0);

만약 동일한 타입의 객체들을 저장할 경우에는 미리 타입을 예약한다.
예) ArrayList a = new ArrayList();
a.add(new TestClass()); // TestClass타입으로 저장 됨.
TestClass tt = a.get(0); // 데이터 타입 변환 필요 없음.

# ArrayList의 주요 메서드

- add(객체명); // 맨 마지막 위치에 데이터 추가
- get(인덱스번호); // 지정한 위치의 데이터 추출
- remove(인덱스번호); // 지정한 위치의 데이터를 제거하고 그 데이터를 반환
- size(); // 데이터의 갯수를 반환

- ArrayList에 데이터를 추가하거나 삭제할 경우, size() 는 자동으로 바뀐다.
  예) ArrayList a = new ArrayList();
  a.add(new TestClass); // size() => 1
  a.add(new TestClass); // size() => 2
  a.add(new TestClass); // size() => 3

  a.remove(0); // size() => 2
  이 때 제거된 데이터 위치로 다른 데이터들이 한칸씩 이동 된다.