**DEV #5 – 2020.04.21**

>> 세계 최초의 웹사이트를 찾으시오.

>> <https://opentutorials.org/course/1223> 학습하세요

-. 최신버전 Eclipse IDE for Enterprise Java Developers 압축파일(zip)다운로드 (64bit)

-. OpenJDK 1.8.0 Window 버전 다운로드 (zip)

-. Apache Tomcat 8.5 다운로드 (zip)

-. Apache Maven Download (zip)

-. 개발환경을 구성하세요.

>> Java 필수학습내용

-. 유효범위

-유효범위(scope)는 변수 내 지역변수가 존재하지 않는다면 소속된 클래스의 전역변수를 사용하는데 메서드 내에 존재하는 같은 이름의 변수의 경우 메서드 내 존재하는 변수의 우선순위가 높아 클래스의 변수에 접근하기 위해서는 this. 메서드를 사용해서 클래스의 변수에 접근할 수 있다

class scope{

int test\_a;

public void TestMethod(int a){

int test\_a = a;

this.test\_a = a;

}

}.

위 클래스에서 int test\_a =a; 는 TestMethod 내에 존재하는 test\_a 라는 변수에 변수에 입력된 a 라는 값을 넣어준 것이고 this.test\_\_a =a; 는 scope 라는 클래스에 존재하는 test\_a 라는 변수에 TestMethod에 입력받은 값을 넣은것이다 .

이 두개의 차이는 변수를 메서드 내에서만 쓸것이냐 클래스 내에서 전체적으로 사용할 것이냐의 차이이다

-. abstract (추상class)

-abstract 는 추상 class 를 만들때 사용하는 메서드로 abstract 로 선언된 경우 실체를 가진 자식클래스에서 정의를 해 주어야 사용이 가능하다.//abstract 선언시 상속받은 자식클래스에서 무조건 정의해주어야 함

추상class 의 경우 변수나 메서드가 구체적으로 정의되어있지 않기 때문에 객체를 생성할 수가 없다

abstract 를 사용하여 class 를 만드는 이유는 강제성을 띄는 변수나 메서드등을 부여하여 자식클래스들의 변수와 메서드들의 통일성을 유지하여 유지보수시 편리함을 제공한다.

**public** **abstract** **class** Animal {

**public** String kind;

**public** **void** breath(){

        System.out.println("숨 쉰다.");

    }

    //추상메서드

**public** **abstract** **void** sound();//구체적인 구현부는 없음!

}

**public** **class** Dog **extends** Animal{

**public** Dog(){

**this**.kind = "포유류";

    }

    @Override

**public** **void** sound() {

        // TODO Auto-generated method stub

        System.out.println("왈왈!");

    }

}

위 코드의 경우 Animal class 를 abstract 로 선언하여 추상 class 를 만들었고 Dog 라는 클래스에 Animal 클래스를 상속받아 abstract로 선언된 sound 메서드를 재정의하여 Dog 클래스의 sound는 왈왈! 이라는 문자열을 반환하는 메서드로 재정의하였다

같은 방법으로 cat,cow 등의 클래스를 정의하여 “야옹” 이나 “음머” 등의 울음소리를 내는 동물들의 클래스도 정의하여 사용할 수 있다

이러한 추상클래스의 경우 다형성을위하여 좀더 빠른 코딩을 위하여 사용한다

-. interface (인터페이스)

-interface는 추상메드를 모아둔 것으로 interface 사용시 이러한 추상메서드를 구현하지 않는다면 컴파일 에러가 발생한다.

Interface 의 경우 abstract와는 다르게 다중상속이 가능한데

public interface Walkable {

void walk();

}

public interface Flyable {

void fly();

}

public interface Moveable extends Walkable, Flyable {

}

위와같이 Moveable 클래스에 Walkable, Flyable 를 상속하여 Movealbe 클래스에서는 walk(); 와 fly(); 라는 기능을 한번에 제공한다.

interface의 경우 선언시 무조건 public으로 선언하여야하며 변수의경우

public static final로 선언해야한다(생략시 default)

interface도 abstract와 같이 다형성을 위하여 사용을 한다

-. 제네릭

제네릭은 실행시 타입에러가 나는것보단 컴파일시 미리 타입을 체크해 에러를 사전에 방지하기 위해서 사용하는 것으로 메서드를 사용할 때 불핑요한 형변환을 줄이게 만들어 주어 프로그램 성능을 좋게만들기 위해서 사용한다

List list = new ArrraList();

list.add(“hello”);

String str = **(String)**list.ger(0); //형변환 필요

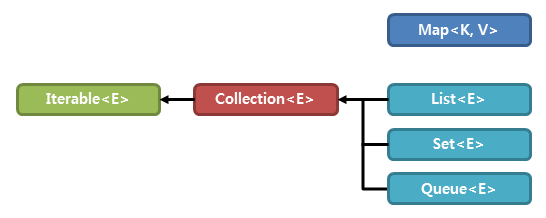
List**<String>** list = new ArrraList();

list.add(“hello”);

String str = list.ger(0); //형변환 불필요

제네릭 타입은 클래스나 인터페이스에 <>부호를 사용하여 여러종류의 타입이 올 수 있게만드는 것으로 <?> 를 사용시(?는 임의의 알파벳 한 글자를 일반적으로 사용)Object 타입(모든종류)객체를 저장할 수 있지만 저장과 불러오기시 형변환이 발생하기 때문에 <? extents 타입> 을 사용하여 지정되는 구체적인 타입을 제한한다. 이러한 것을 와일드카드라고 하는데 제네릭타입 <?> 은 모든 클래스나 인터페이스 타입이 올 수 있음을 의미하고 extends라는 키워드 를 사용하면 상위클래스를 제한, super라는 키워드를 사용하면 하위 클래스를 제한할 수 있다

-. Collection Framework



자바에서

collection framework 를 구성하고 있는 인터페이스 간의 상속관계는 위 그림과 같다. 위 그림의 <E>,<K>,<V>는 collection framework를 구성하는 모든 클래스가 제네릭으로 표현되어 있음을 나타내는 것이다

List 는 순서가 있는 데이터들의 집합으로 ,데이터를 의중복을 허용하는 것으로

Vector, ArrayList, LinkedList, Stack, Queue 등이 있고

Set 은 순서가 없는 데이터의 집합으로 ,데이터들의 중복을 허용하지 않는 것으로

HashSet, TreeSet 등이 있다.

Map은 키와 값의 한 쌍으로 이루어지는 데이터의 집합으로, 순서사 없다.이떄 키는 중복을 허용하지 않지만 ,값(value)는 중복 될 수있다

HashMap, TreeMap, Hashtable, Properties 등이 존재

import java.util.ArrayList; //ArrayList 를 사용할 수 있게 impoet 해줌 (없을시 에러 발생)

-import java.util.\* 을 사용하면 List,Hash,Map 모두 사용가능

public class ArrayListDemo {

public static void main(String[] args) {

String[] arrayObj = new String[2];

arrayObj[0] = "one";

arrayObj[1] = "two";

arrayObj[2] = "three"; //배열크기를 벗어난 입력임으로 에러가 발생함

for(int i=0; i<arrayObj.length; i++){

System.out.println(arrayObj[i]);

}

ArrayList al = new ArrayList();

al.add("one");

al.add("two");

al.add("three");

for(int i=0; i<al.size(); i++){

System.out.println(al.get(i));

}

}

}

위의 코드를 살펴보자면 기본 배열타입 선언시 배열의 크기를 제한하기 때문에 배열의 크기보다 더 많은 값이 들어오게 된다면 에러를 발생시키지만 ArrayList 로 벼열 선언시 동적으로 배열의 크기를 관리해준다.

컬렉션을 사용할 때는 데이터 타입은 가급적으로 해당 컬렉션을 대표하는 인터페이스를 사용하여주어야 좋다.

각 컬렉션들은 그림과 같이 인터페이스를 공유하는 부분이 존재하기 때문에

Map<String, Integer> a = new HashMap<String, Integer>();

Map<String, Integer> a = new Hashtable<String, Integer>();

HashMap 을 Hashtable 로 변경하고 싶다면 위와같이 변경하여도 에러가 발생하지 않는다.

-iterator

iterator 라는 메서드는 interface Collection 에 정의되어 있다.

Collections framework 사용시 iterator 를 통해 데이터를 for문을 사용한것과 같이 사용할 수 있다.

hashNext() : 반복할 데이터가 더 있으면 true.없으면 false를 리턴해준다

next() : hashNext 가 true 라는 것은 next 가 리턴할 데이터가 존재함.

Iterator hi = A.iterator();

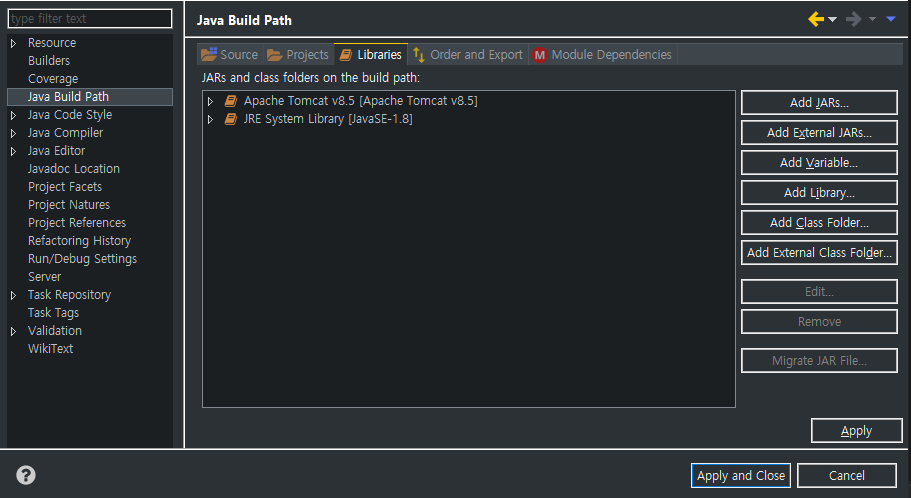
**while**(hi.hasNext()){

System.out.println(hi.next());

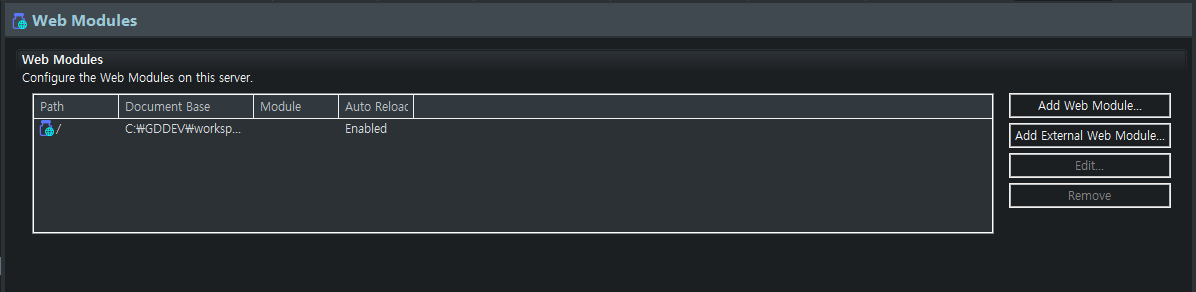
위 코드를 통해 Hash,List,Map 등에 존재하는 데이터를 하나씩 출력해 낼 수 있다 --for 문처럼 사용가능

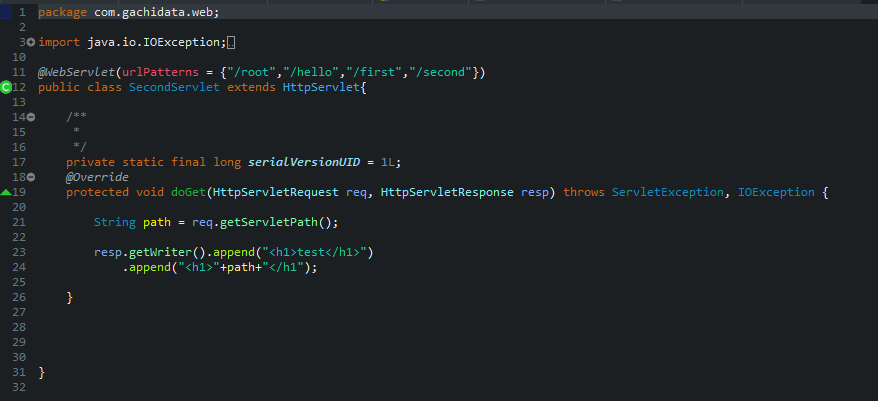
tomcat 을 이용한 server 구축하기

project 생성하여 라이브러리에 apache tomcat 라이브러리를 추가해준다



프로젝트에 webroot/WEB\_INF/lib 를 추가한뒤 index.html 파일을 webroot 에 생성해준뒤 server 탭에서 index.html 을 찾을 수 있게 모듈의 경로를 추가해준다

 src 디렉터리에 HttpsServlet 을 상속한 클래스를 정의해 준다 // 서블렛 추가시 “/”이 빠지게 되면 오류가 발생함

 이후 서버에 지정된 포트로 접속하면 web 서버가 구축된것을 확인할 수 있다

