8_01_2022__java_Interface, GenericClass, UtilClass

Step010_Interface

~Step12_UtilClass

FunctionalInterface

함수 모양으로 사용할 인터페이스에 붙이는 어노테이션

- 추상 메소드가 1개인 인터페이스
- 추상 메소드의 갯수가 1개로 강제됨

```
@FunctionalInterface
public interface Calc {
  //인자로 숫자 2개를 전달 받아서 어떠한 연산을 한 후 결과를 리턴해주는 추상 메소드
  public double exec(double num1, double num2);
}
```

```
package test.main;

import test.mypac.Calc;

public class MainClass05 {
  public static void main(String[] args) {
    Calc plus=(a, b)->{
      return a+b;
    };

    Calc minus=(a, b)->{
      return a-b;
    };

    Calc multi=(a, b) -> a*b;
    Calc divide=(a, b) -> a/b;


    double result1 = plus.exec(10, 20);
    double result2 = minus.exec(10, 20);
    double result3 = multi.exec(10, 20);
    double result4 = divide.exec(10, 20);
  }
}
```

Generic Class

예를들어 Apple Orange Banana 가 있다면

```
package test.mypac;

public class FruitBox {
  //field
  private Apple item;
  //field에 값을 넣는 method
  public void setItem(Apple item) {
    this.item=item;
  }
  //field에 저장된 값을 리턴하는 method
  public Apple getItem() {
    return item;
  }
}
```

만약 우리가 용도에 따라 타입을 지정해서 바꿀 수 있다면 좋지 않을까?

포괄 클래스로(generic class) 정해놓고 사용하는 시점에 원하는 클래스로 불러오기

```
package test.mypac;

public class FruitBox<T> {
  //field
  private T item;
  //field에 값을 넣는 method
  public void setItem(T item) {
    this.item=item;
  }
  //field에 저장된 값을 리턴하는 method
  public T getItem() {
    return item;
  }
}
```

FruitBox 옆에 추가한<T>를 보자 다른 클래스에도 T를 추가해줬다.

T는 Type을 의미한다.

```
package test.main;

import test.mypac.Apple;
import test.mypac.Banana;
import test.mypac.FruitBox;
import test.mypac.Orange;

public class MainClass01 {
  public static void main(String[] args) {
    FruitBox<Apple> box1=new FruitBox<Apple>();
    FruitBox<Banana> box2=new FruitBox<Banana>();
    FruitBox<Orange> box3=new FruitBox<Orange>();
```

```
  }
}
```

원하는 시점에 타입을 정해서 불러올 수 있다.

```
package test.main;

import test.mypac.Apple;
import test.mypac.Banana;
import test.mypac.FruitBox;
import test.mypac.Orange;

public class MainClass01 {
  public static void main(String[] args) {
    //Generic 클래스를 Apple 로 지정해서 FruitBox 객체 생성하기
    FruitBox<Apple> box1=new FruitBox<Apple>();
    //Method의 인자로 Apple type 전달하기
    box1.setItem(new Apple());
    //Method가 리턴해주는 Apple type 받아오기
    Apple item1=box1.getItem();

    //Generic  클래스를 Banana로 지정해서 Fruit 객체를 생성 위와 비슷한 작업을 해 보세요
    FruitBox<Banana> box2=new FruitBox<Banana>();
    box2.setItem(new Banana());
    Banana item2=box2.getItem();


    FruitBox<Orange> box3=new FruitBox<Orange>();

  }

}
```

T자리에 다른 클래스들을 불러올 수 있다.

Util Class

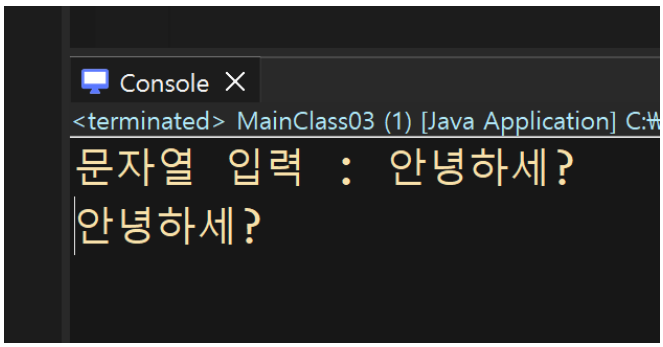
- ArrayList
  - 가변배열 -어떤 아이템을 순서있게 관리할 수 있는 배열

```
package test.mypac;

import java.util.ArrayList;

public class MainClass01 {
  public static void main(String[] args) {
    ArrayList<String> names=new ArrayList<String>();
    // "dubu" , "yuki" , "soju"  3개의 String type을 저장해 보세요
    names.add("dubu");
    names.add("yuki");
    names.add("soju");

    //0번방의 아이템을 불러와서 item이라는 변수에 담아 보세요
    String item = names.get(0);
    //1번방의 아이템을 삭제하려면
```

```
    String item = names.remove(1);
    //0번방에 Beale을 넣고 싶으면?
    names.set(0, "Beale");
    //저장된 아이템의 갯수(size)를 size 라는 지역변수에 담아보세요
    int size=names.size();
    //저장된 모든 아이템 전체 삭제
    names.clear();

  }
}
```

- Scanner
  - 콘솔창으로부터 문자열 입력 받는 객체

```
package test.mypac;

import java.util.Scanner;

public class MainClass03 {
  public static void main(String[] args) {
    Scanner scan = new Scanner(System.in);
    //문자열 1줄 입력 받고
    System.out.print("문자열 입력");
    String line=scan.nextLine();
    //입력 받은 내용 출력하기
    System.out.println(line);
  }
}
```

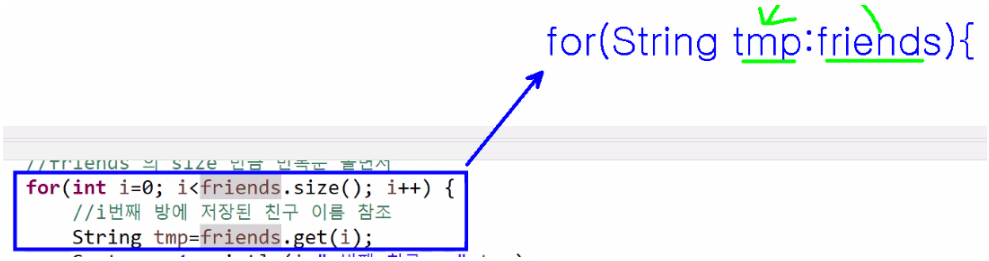


콘솔창에 입력한 문자열을 출력해줌

```
package test.mypac;

import java.util.ArrayList;
import java.util.Scanner;

public class MainClass04 {
  public static void main(String[] args) {
    /*
     * Scanner 객체를 이용해서 반복문 돌면서 친구 이름을 3번 입력 받아서
     * 입력받은 이름을 ArrayList 객체에 순서대로 저장하는 프로그래밍을 해보세요
     */
    ArrayList<String> friends = new ArrayList<>();
    Scanner scan = new Scanner(System.in);
    for (int i = 0; i < 3; i++) {
      System.out.print("친구 등록 : ");
      String line=scan.nextLine();
```

```
      friends.add(line);
    }
    //저장된 내용 출력하기
    for(String tmp:friends) {// 확장for문
      System.out.println(tmp);
    }

  }
}
```

- 확장 for 문



- ArrayList 는 기본 데이터 type 을 저장 할 수 없으므로 기본 데이터 type 을 저장하고 싶으면 Wrapper Class를 활용하면 된다.
  - int -> Integer
  - double -> Double
  - boolean -> Boolean

```
ArrayList<Integer> nums = new ArrayList<Integer>();
nums.add(10);
nums.add(20);
nums.add(30);
```

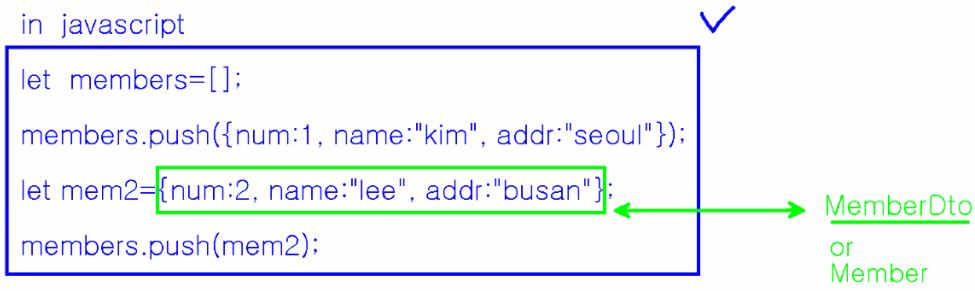
예제

```
package text.mypac;

public class Member {
  public int num;
  public String name;
  public String addr;

  /*
   * 기본 생성자도 필요하다면 정의 할 수 있다.
   * 생선자는 다중 정의가 가능하다
   * 따라서 어떤 객체를 생성하는 방법이 여러 가지가 될 수도 있다는 것이다.
   */
  public Member() {}

  //필드에 저장할 값을 전달하는 생성자
  public Member(int num, String name, String addr) {
```

```
    this.num=num;
    this.name=name;
    this.addr=addr;
  }
}
```

```
package test.main;

import java.util.ArrayList;

import text.mypac.Member;

public class MainClass07 {
  public static void main(String[] args) {
    // 1. Member 객체를 담을 수 있는 ArrayList 객체를 생성해서 참조값을 members라는 지역변수에 담아 보세요
    ArrayList<Member> members = new ArrayList<Member>();
    // 2. 3명의 회원정보를 Member 객체에 각각 담아 보세요. (Member 객체가 3개 생성되어야 함)
    Member memb0 = new Member(1, "두부", "Fairfax");
    Member memb1 = new Member(2, "유키", "High Desert");
    Member memb2 = new Member(3, "소주", "Richmond");

    // 3. 위에서 생성된 Member 객체의 참조값을 members ArrayList 객체에 모두 담아 보세요.
    members.add(memb0);
    members.add(memb1);
    members.add(memb2);

    /*
     * 4. members ArrayList 객체에 담긴 내용을 이용해서 회원목록을 아래와 같은 형식으로 반복문 돌면서 출력해 보세요
     *
     * 번호 : 1, 이름 : 두부, 주소 : 페어펙스 ...
     */
    for (Member tmp : members) {
      System.out.println("번호 : " + tmp.num + ", 이름 : " + tmp.name + ", 주소 : " + tmp.addr);
    }
  }
}
```

자바스크립트의 오브젝트와 비슷한 Member(MemberDto)



그리고 HashMap 이 비슷함

HASH MAP

```
new HashMap<K, V>();
```

- Key 값은 대부분 String 문자열이 일반적
- Value에는 원하는 type을 담으면 됨

```
new HashMap<String, Object>();
```

```
package test.main;

import java.util.HashMap;
import java.util.Scanner;

public class MainClass09 {
  public static void main(String[] args) {
    HashMap<String, String> dic = new HashMap<>();
    dic.put("house", "집");
    dic.put("phone", " 전화기");
    dic.put("car", "차");
    dic.put("pencil", "연필");
    dic.put("eraser", "지우개");


    /*
     * 검색할 단어를 입력하세요 : house
     * house의 뜻은 집 입니다.
     *
     * 검색할 단어를 입력하세요 : dubu
     * dubu는 목록에 없습니다.
     */


    Scanner scan = new Scanner(System.in);
    System.out.print("검색할 단어를 입력하세요 : ");
    String line=scan.nextLine();
    boolean kc = true;
    if (kc == dic.containsKey(line)){
      System.out.println(line+"의 뜻은 "+dic.get(line) +" 입니다.");

    }else {
      System.out.println(line+" 은(는) 목록에 없습니다.");
```

```
    }

//    선생님
//    Scanner scan = new Scanner(System.in);
//    System.out.print("검색할 단어를 입력하세요 : ");
//    String word=scan.nextLine();
//    String mean=dic.get(word);
//
//    if(mean ==null) {
//      System.out.println(word+" 는 목록에 없습니다.");
//    }else {
//      System.out.println(word+" 의 뜻은 " + mean+" 입니다.");
//    }


  }
}
```

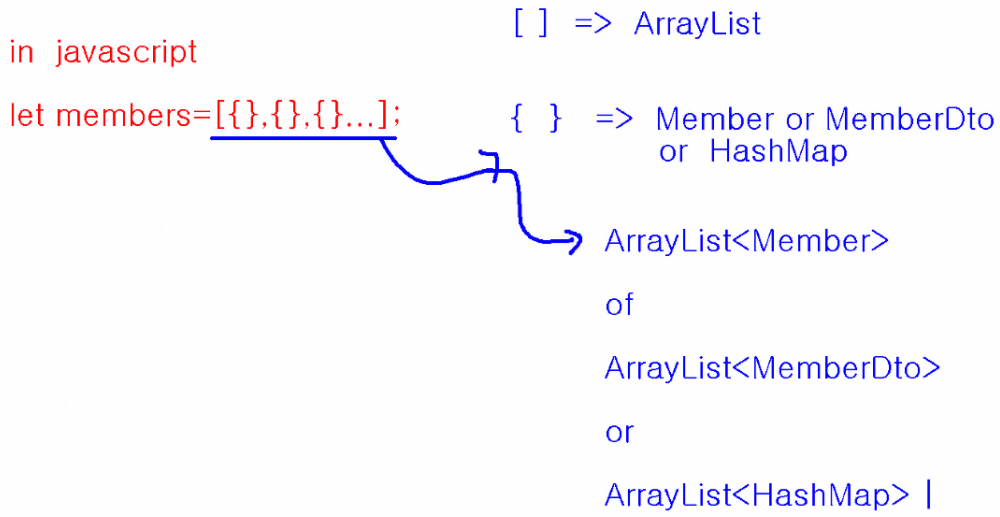
HashMap과 Casting

- 여러 타입을 담고 싶으면 value type을 Object로 사용할 수 있지만 리턴시 casting 을 사용해 원래 타입으로 바꿔 주어야 한다.

```
package test.main;

import java.util.HashMap;

public class MainClass10 {
  public static void main(String[] args) {

    HashMap<String, Object> map = new HashMap<>();
    //value의 generic이 object 이기 때문에 어떤 type 이든지 담을 수 있다.
    map.put("num", 1);
    map.put("name", "Dubu");
    map.put("addr", "Fairfax");

    //Object 타입으로 리턴되기 때문에 원래 타입으로 casting 해 주어야 한다.
    int num=(int)map.get("num");
    String name=(String)map.get("name");
    String addr=(String)map.get("addr");

  }
}
```

HashMap은 js의 Object와 비슷하다.

- 배열에 오브젝트를 담는 방법



ArrayList에 Member 혹은 MemberDto 혹은 HashMap을 넣으면 된다.

```
ArrayList<HashMap<String,Object>>
```

HashSet

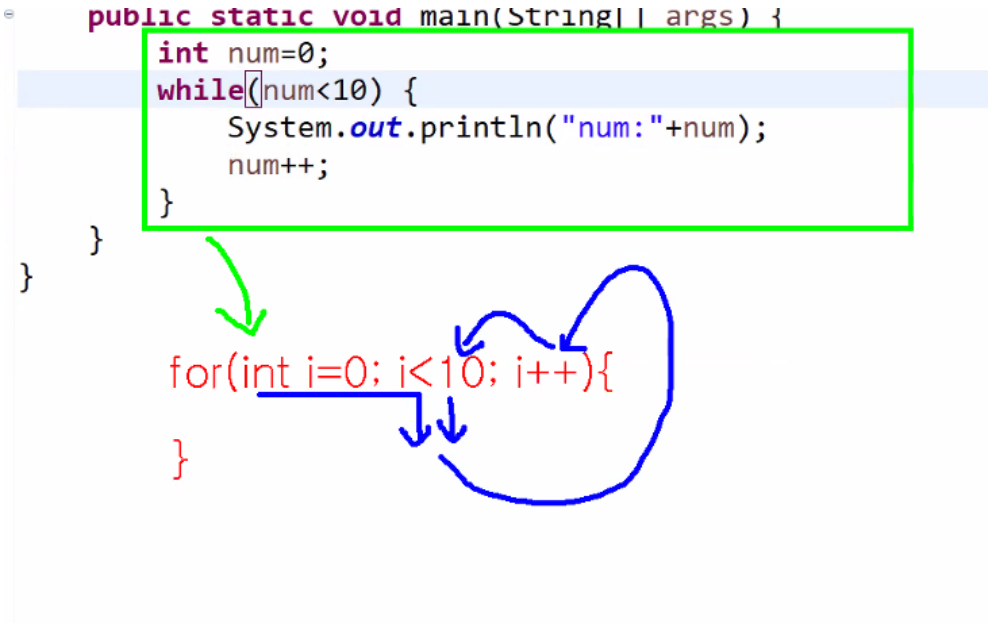
어떤 데이터를 저장하되 중복을 제거하고 저장하고 싶을 때 사용

- HashSet 은 Set 인터페이스를 구현한 클래스이다.
- 순서가 없다.
- key 값도 없다.
- 중복을 허용하지 않는다.
- 어떤 data를 묶을(집합) 으로 관리하고자 할 때 사용한다.

```
HashSet<Integer> set1 = new HashSet<>();
      set1.add(10);
      set1.add(20);
      set1.add(20);
      set1.add(30);
      set1.add(30);
```

WHILE

반복문



for는 반복횟수를 알 수 있을 때 주로 사용한다.

while은 반복횟수를 알 수 없을 때 주로 사용한다.