8_09_2022__java_Review

1. 확장 for문

- 일반 for 문 (for 문 안에서 i 값이 필요할 때 사용한다.)

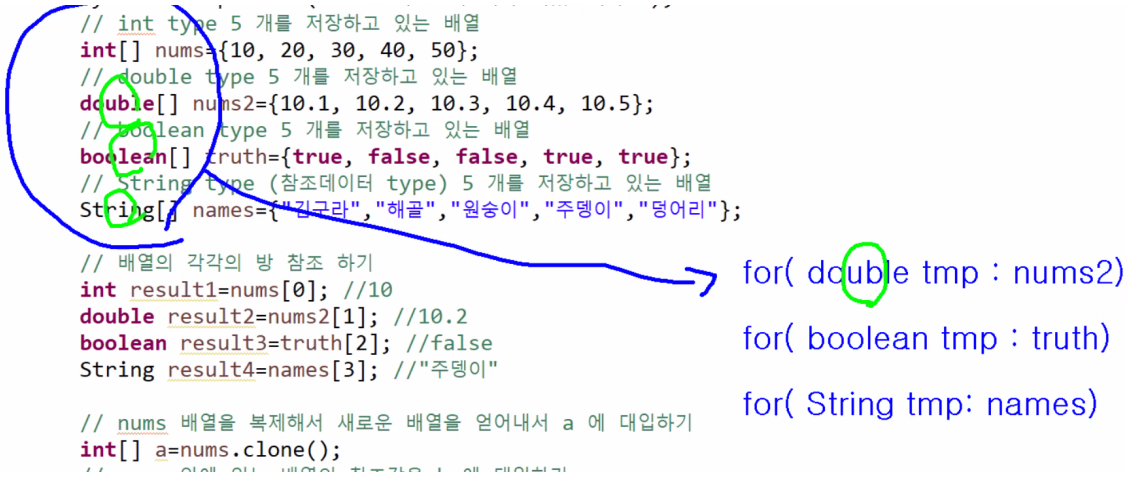
```
int[]nums = {10, 20, 30, 40, 50};

for(int i=0; <nums.length; i++){
  int tmp = nums[i]
}
```

- 확장 for 문 (i값 필요없이 순서대로 빼오고 싶다면 사용한다.)

```
int[]nums = {10, 20, 30, 40, 50};

for(int tmp:nums){ }
```



빼오는 것에는 타입이 중요하다.

2. Static Method

- 메소드를 만들 때 static을 붙여 만든 메소드 들이 있음
- new 하지 않고 class명에 . 찍어서 호출한다.

3. InnerClass

- class 안에 다른 class를 정의해도 된다.

- method 안에 정의한 class는 로컬 클래스라고도 불린다.

4. Map, List, Array

- Map
  - 순서가 중요하지 않은 데이터를 key값과 value값으로 데이터를 관리하고 싶을 때 사용함
  - key값만 기억하고 있으면 데이터를 빼낼 수 있다.
- List
  - 순서가 중요한 데이터를 사용할 때 add 하거나 지울 때 방의 갯수가 유동적이다.
- Array
  - 방의 갯수가 정해지면 방을 추가하거나 제거할 수 없다.

5. TryCatch

```
try{
  여기에서 발생할 가능성이 있는 Exception type을
}catch(여기에 type을 선언하고){
  해당 Exception이 발생했을 때 원하는 작업을 여기서 한다.
}
```

- try안의 발생하는 Exception 여러개면 catch에서 세부적으로
  exception 종류별로 만들어 처리할 수 있다.

6. Generic Class

- 클래스를 만들다보면 그 안에서 사용하는 타입이 있는데 그걸 고정하지 않고 동적으로 적용할 수 있게 하는 문법이다. 사용자가 필요에 따라 결정할 수 있게 만들어 놓은 문법

7. Interface

- 구현되지 않은 추상메소드만 가지고 있다.
- 특정 인터페이스를 구현을 하면 기능을 override 해 사용할 수 있다.
- 자기 마음대로 만드는게 아니고 인터페이스에 정의된 모양대로 만들수 있도록 한다.
- 특별한 용도로 사용될 클래스를 만들어서 납품을 해야할때
  - 특별한 용도 → 이미 어떻게 사용될 지 정해져 있음
  - 어떻게 사용될지 정해져 있음 → 메소드를 마음대로 못 만듦
  - 인터페이스에 정해진대로 만들면 됨

8. Thread

- run 했을 때 시작되는 작업의 흐름

- 눈에 보이지 않는 어떤 실행의 흐름(작업단위)이다.
- 어떤 app이 시작되었다는 것은 누군가가 미리 준비된 java code를 순서대로 실행한 것이다.
- main method에서 시작된 실행의 흐름을 main thread라고 한다.

9. InputOutput

- Step10_InputOutput
  - MainClass16

```
package test.main;

import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;

public class MainClass16 {
  public static void main(String[] args) {
    // 필요한 참조값을 담을 지역 변수를 미리 만든다.
    FileInputStream fis = null;
    FileOutputStream fos = null;
    try {
      // 1. jpg 에서 byte 를 읽어낼 객체
      fis = new FileInputStream("C:\\Users\\HNJN-PC\\Desktop\\school\\java_work\\myFolder\\1.jpg");
      //copied.jpg 에 byte를 출력할 객체
      fos = new FileOutputStream("C:\\Users\\HNJN-PC\\Desktop\\school\\java_work\\myFolder\\copied.jpg");
      //byte 알갱이를 담을 방 1024개 짜리 byte[]객체 생성
      byte[] buffer=new byte[1024];

      while (true) {
        //byte[] 객체를 read() 메소드에 전달해서  byte를 읽어내고 몇 byte를 읽었는지 리턴 받는다
        int readByte=fis.read(buffer);
        System.out.println(readByte);
        //만일 더이상 읽을 byte 가 없다면
        if(readByte==-1)break;
        fos.write(buffer, 0, readByte);
      }
    } catch (IOException e) {
      e.printStackTrace();
    } finally {
      // fos, fis 마무리 하기

        try {
          if(fis!=null)fis.close();
          if(fos!=null)fos.close();
        } catch (IOException e) {
          e.printStackTrace();
        }
    }
  }
}
```

복사