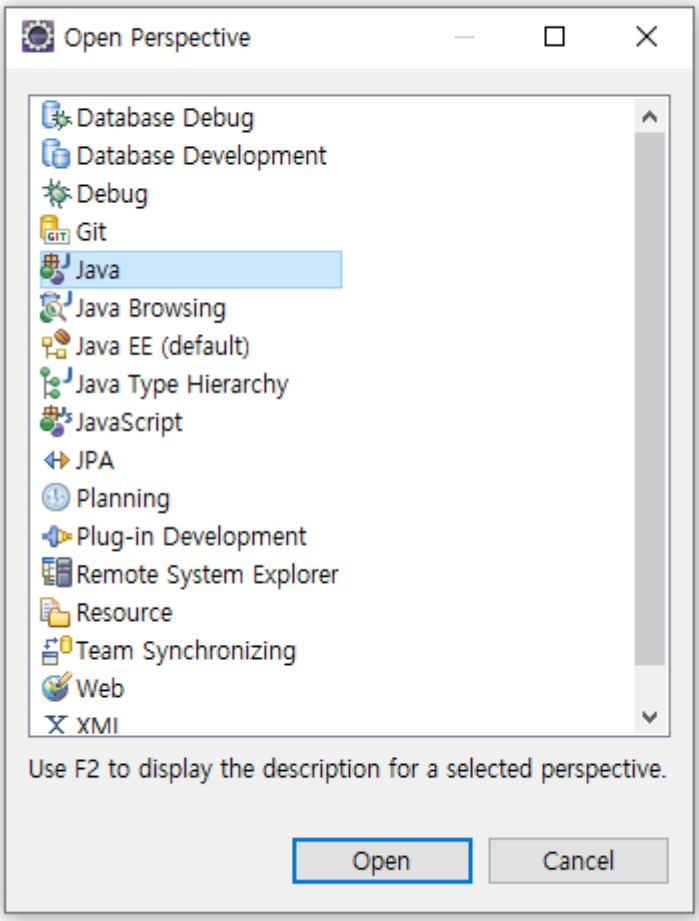
JAVA 기초

기본 세팅

- 버전: 11.0.12
- 이클립스 세팅



- Encoding 잊지 말기~

JAVA 종류

- BE 개발 → Enterprise Edition
- 앱 개발 → Micro Edition

변수

메모리에 언제 생성되는지, 언제 사용할 수 있는지 아는 게 중요

- 클래스 변수 (class variables)
  - 공용 변수, 정적 변수, static 변수
  - class가 로딩될 때 단 한 번 생성되는 변수
- 인스턴스 변수 (instance variables)
  - 객체가 생성될 때 메모리에 올라오는 변수
- 매개 변수 (parameter variables)
  - 메서드를 호출할 때 값을 전달받기 위해 사용하는 변수
  - 지역 변수와 범위가 같음
- 지역 변수 (local variables)
  - 특정 메서드나 범위 안에서만 사용할 수 있는 변수
  - 특정 블록 내에서 선언된 변수

1. 반드시 초기화 후 사용해야 한다.

```
int num;
num += 1;
// The local variable num may not have been initialized
// 초기화 오류
System.out.println(num);

int num = 0; // 초기화 후 사용 필요
```

2. 선언된 변수는 정의된 범위 안에서만 사용할 수 있다.

- 지역 변수는 선언된 위치가 중요하다!
- 예외 처리 시 중요하다

```
// 변수를 try~catch 내 선언하면 밖에서 사용할 수 없다
FileInputStream in = null;
try {
      in = new FileInputStream("aa.txt");
    } catch (Exception e) {
```

```
  }

// 변수를 안쪽에 선언하면 찾을 수 없다는 오류 출력
// a cannot be resolved to a variable
if(num == 1) {
      int a = 0;
      a = 100;
  }
  System.out.println("a: " + a);
```

변수 선언

- 주의사항

1. **변수명은 특수문자를 사용하지 않는다.**
2. **숫자로 시작하지 않는다.**
3. **소문자로 시작한다.**
4. **합성어는 두 번째 단어 첫 글자를 대문자로 쓴다.**

```
int num; // 변수 선언 (메모리 할당)
num = 10; // 변수 초기화
int num2 = 20;
```

cf) javascript → 인터프리터

- 데이터에 대한 엄격한 규정이 없다.

java → 컴파일 언어

- 데이터 타입이 중요하다.

기본 리터럴

- 정수값: int
- 실수값: double
- ‘A’: char
- “문자열”: String
- true: boolean

```
double d = 3.14;
float f = (float) 3.14; // 에러. double형을 float형에 넣으려고 하기 때문
// cannot convert from double to float
```

```
// 명시적 형 변환 하거나, 뒤에 f 붙여 주기
float ff = 3.14f;
```

상수

- 클래스 상수: 객체 생성 안 해도 사용할 수 있도록 하기

```
public static final int age = 20;
클래스 이름.상수 이름
```

자료형

기본형

- 8가지
  - 정수형: byte, short, int, long
  - 실수형: float, double
  - 문자형: char
  - 부울형: boolean

참조형

- String → class에서 나오기 때문에 object 타입
- 주소값이 있다.

Wrapper 클래스

- Byte, Short, Integer, Float, Double
- 기본 자료형을 참조형(Object type)으로 변환(Boxing)
- Java 1.5부터 컴파일러는 Boxing과 Unboxing이 필요한 상황에서 자동으로 처리

```
ArrayList<Integer> list = new ArrayList<Integer>();
list.add(3);

// 컴파일러가 자동으로 변환 (AutoBoxing)
list.add(Integer.valueOf(3));
```

```
int n = list.get(i);
// 컴파일러 자동 변환 (AutoUnboxing)
int n = list.get(i).intValue();
```

형 변환 (casting)

묵시적

- 데이터가 작은 값이 큰 값으로 바뀌는 것
    - 순서: byte < short < int < long < float < double

예) int → long

명시적

- 데이터가 큰 값이 작은 값으로 바뀌는 것
- 강제 형 변환 필요: 데이터 손실 유발

```
int b = (int)3.14;
System.out.println(b); // 3

double num3 = 3.14 + 1; // double로 변환
System.out.println(num3); // 4.140000000000001 컴퓨터는 실수형에서 오차 발생
if(num3 == 4.14) {
      System.out.println("같다.");
} System.out.println("다르다"); // "다르다" 출력

// int와 char 변환
char num4 = 'A';
System.out.println((int)num4); // 65
int num5 = 66;
System.out.println((char)num5); // B

// String
String str = new String("AAA");
String str2 = "BBB"; // 참조형이지만 기본형처럼 쓰게 해 준다

// String과 int 변환

// String => int
String str3 = "1";
int num6 = Integer.parseInt(str3);
// 연산할 경우 다 String으로 바뀌기 때문에, 그전에 int로 변환해 주기
int result = num6 + 1;
System.out.println(result); // 2
// 예외
String str4 = "1안녕";
int num7 = Integer.parseInt(str4); // java.lang.NumberFormatException:
```

```
// int => String
int nnum = 100;
// 1. api 이용
String sstr = String.valueOf(nnum);
// 2. + ""
String ssstr = 100 +"";

// 퀴즈
String strstr = 7 + "7" + 7; // 777
```

문제 풀기

1번

- 임의의 정수값에 대해 전체 자릿수 중 짝수, 홀수의 개수를 구하기
- 예) 12345 ⇒ 짝수: 2개, 홀수: 3개

내 풀이

```
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;

public class Mission01 {
  public static void main(String[] args) throws IOException {
    // 임의의 정수값에 대해 전체 자릿수 중 짝수, 홀수의 개수를 구하자.
    // 다섯 자리 수 예) 12345
    // 짝수: 2개, 홀수: 3개
    BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
    BufferedWriter bw = new BufferedWriter(new OutputStreamWriter(System.out));
    String str = br.readLine();
    int[] arr = new int[str.length()];
    for (int i = 0; i < str.length(); i++) {
      arr[i] = str.charAt(i) - '0';
    }
    int odd = 0;
    int even = 0;

    // if/else 이용
    for (int i = 0; i < str.length(); i++) {
      if (arr[i] / 2 == 0) {
        even++;
      } else {
        odd++;
      }
    }
```

```
  // 삼항연산자 이용
  for(int i = 0; i < str.length(); i++) {
  // 임시 변수 왼쪽에 배치하고, 괄호 잘 사용하기
    int tmp == (0 == (arr[i] % 2)) ? ++even: +odd;
  }

    bw.write("짝수: " + String.valueOf(even) + "개\n");
    bw.write("홀수: " + String.valueOf(odd)+ "개");
    br.close();
    bw.flush();
    bw.close();
  }

}
```

강사님 풀이

```
package kosa.mission;
import java.util.Scanner;

public class Mission01_ans {
  public static void main(String[] args) {
    // 임의의 정수값에 대해 전체 자릿수 중 짝수, 홀수의 개수를 구하자.
    // 다섯 자리 수 예) 12345
    // 짝수: 2개, 홀수: 3개
    Scanner sc = new Scanner(System.in);
    System.out.print("5자리 정수 입력: ");
    // scanner 입력 시 end 눌러서 커서 끝으로 두고 입력

    int num = sc.nextInt();
    // 지역 변수 꼭 초기화
    int even = 0;
    int odd = 0;

    even += (num / 10000 % 2 == 0) ? 1 : 0;
    even += (num / 1000%10 % 2 == 0) ? 1 : 0;
    even += (num / 100%10 % 2 == 0) ? 1 : 0;
    even += (num / 10%10 % 2 == 0) ? 1 : 0;
    even += (num % 2 == 0) ? 1 : 0;

    odd = 5 - even;
    System.out.println("짝수의 개수: " + even);
    System.out.println("홀수의 개수: " + odd);

  }

}
```

조건문

switch문

- 어떤 케이스가 명확하게 나누어질 때 사용

반복문

do~while문

- 어느 조건이 만족될 때까지 실행해야 한다면 사용하는 것 추천
- 한 번은 꼭 실행

```
do {
      반복 실행할 문장
} while (조건식)

// 조건식이 true면 실행, false면 종료
```

for ~ each 문

```
int[] arr = {1, 2, 3};
for(int num: arr)
  System.out.print(num);
```

문제 1

- 1부터 100까지 정수 중 2의 배수와 3의 배수가 아닌 숫자 출력

내 풀이

```
public class LoopExam {
  public static void main(String[] args) {
    // 1~100까지 정수 중 2의 배수와 3의 배수가 아닌 숫자만 출력
    for(int i = 1; i <= 100; i++) {
      if(i % 2 != 0 && i % 3 != 0)
        System.out.print(i + " ");
    }
  }

}
```

강사님 풀이

- 나랑 반대로 풀었군

```
public class LoopExam_ans {
  public static void main(String[] args) {
    for(int i = 1; i <= 100; i++) {
      if(!(i % 2 == 0 || i % 3 == 0))
        System.out.print(i + " ");
    }
  }
}
```

문제 2

- 2개의 정수를 입력받아 b - a 결과가 항상 양의 정수가 나오도록 구현

내 풀이

```
public class LoopExam2 {
  public static void main(String[] args) throws IOException {
    // 2개의 정수를 입력받아 b - a 결과가 항상 양의 정수가 나오도록 구현
      BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
      BufferedWriter bw = new BufferedWriter(new OutputStreamWriter(System.out));
      StringTokenizer st = new StringTokenizer(br.readLine());
      int a = Integer.parseInt(st.nextToken());
      int b = Integer.parseInt(st.nextToken());
      while (b - a > 0) {
        bw.write(String.valueOf(b - a));
        break;
      }
      bw.flush();
      bw.close();
    }
```

- 입력을 계속 받는 게 목적인 건지……? 그거라면 다시 풀어야 한다

강사님 풀이

1. while문 사용

```
public class LoopExam2_ans {
  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    int a = 0;
    int b = 0;
    while (true) {
      System.out.print("a: ");
      a = sc.nextInt();
      System.out.print("b: ");
```

```
      b = sc.nextInt();
      if (a < b)
        break;
    }
    System.out.println("b-a = " + (b-a));
  }
```

- 입력 계속 받는 거 맞았네 ㅎㅎ

2. do~while문 사용

```
import java.util.Scanner;

public class LoopExam2_ans2 {
  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    int a = 0;
    int b = 0;
    do {
      System.out.print("a: ");
      a = sc.nextInt();
      System.out.print("b: ");
      b = sc.nextInt();
    } while (a > b);
    System.out.println("b-a = " + (b - a));
  }
```