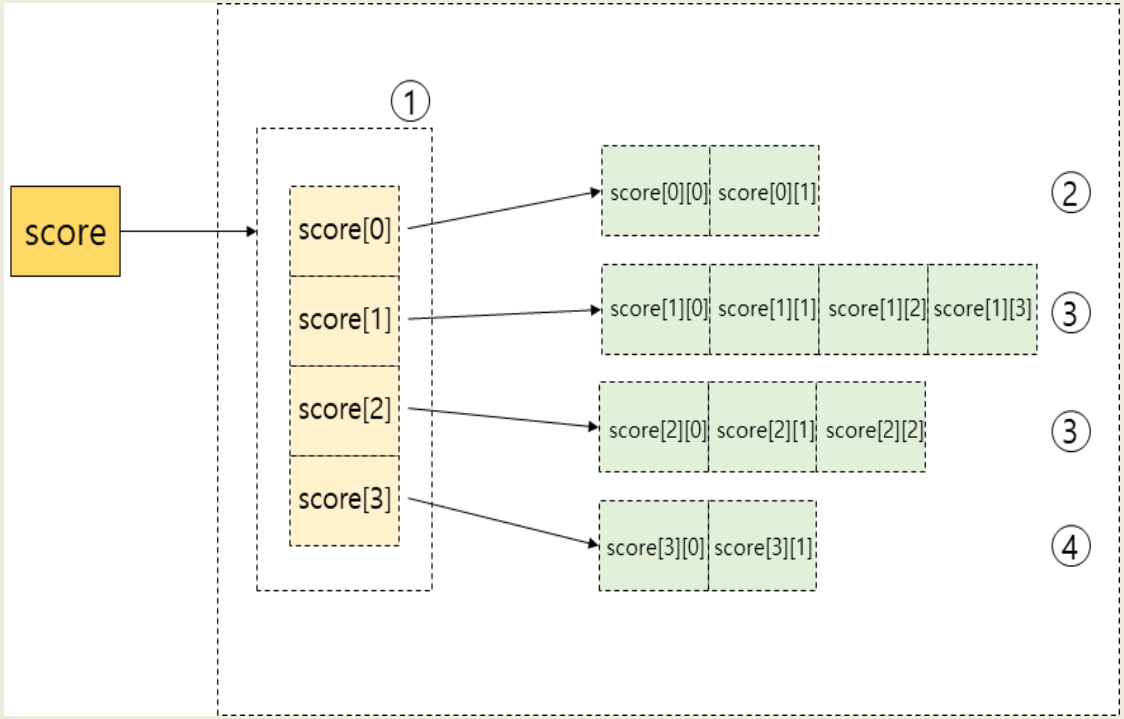
제6장
유용한 참조타입과 배열
Array 4부

비정방형 배열(Ragged Array)

- 정방형 배열
  - 각 행의 열의 개수가 같은 배열
- 비정방병 배열
  - 각 행의 열의 개수가 다른 배열
  - 비정방형 배열의 생성

```
int[ ][ ] score;
score = new int[4][ ];
score[0] = new int[2];
score[1] = new int[4];
score[2] = new int[3];
score[3] = new int[2];
```

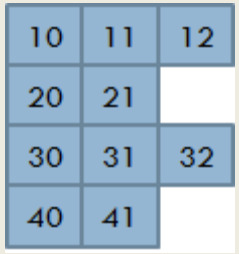


비정방 배열의 Length

- 비정방형 배열의 length
  - score.length -> 2차원 배열의 행의 개수로서 4
  - score[n].length는 n번째 행의 열의 개수
    - score[0].length -> 0번째 행의 열의 개수로서 2
    - score[1].length -> 1번째 행의 열의 개수로서 4
    - score[2].length -> 2번째 행의 열의 개수로서 3
    - score[3].length -> 3번째 행의 열의 개수로서 2

비정방 배열의 생성과 접근 예

다음 그림과 같은 비정방형 배열을 만들어 값을 초기화하고 출력하시오.

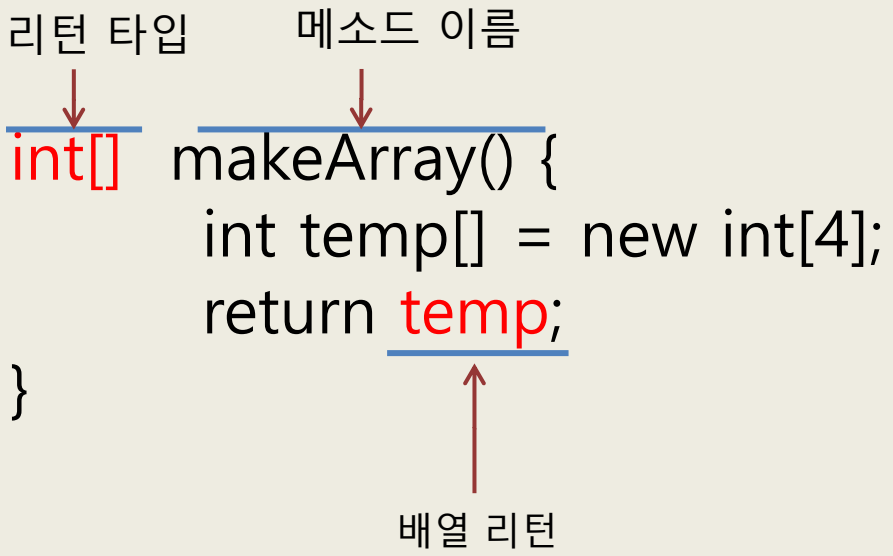



```
public class IrregularArray {
   public static void main (String[] args) {
      int a = 0;
      int intArray[][] = new int[4][];
      intArray[0] = new int[3];
      intArray[1] = new int[2];
      intArray[2] = new int[3];
      intArray[3] = new int[2];
      for (int i = 0; i < intArray.length; i++)
         for (int j = 0; j < intArray[i].length; j++)
            intArray[i][j] = (i+1)*10 + j;
      for (int i = 0; i < intArray.length; i++) {
         for (int j = 0; j < intArray[i].length; j++)
            System.out.print(intArray[i][j]+" ");
         System.out.println();
      }
   }
}
```

```
10 11 12
20 21
30 31 32
40 41
```

배열의 반환

- 메소드가 리턴하는 배열
  - 메소드가 리턴하는 배열의 타입과 차원은 리턴 받는 배열 레퍼런스의 타입 및 차원과 일치해야 함
  - 리턴 타입에 배열의 크기를 지정하지 않음



예제: 배열 반환

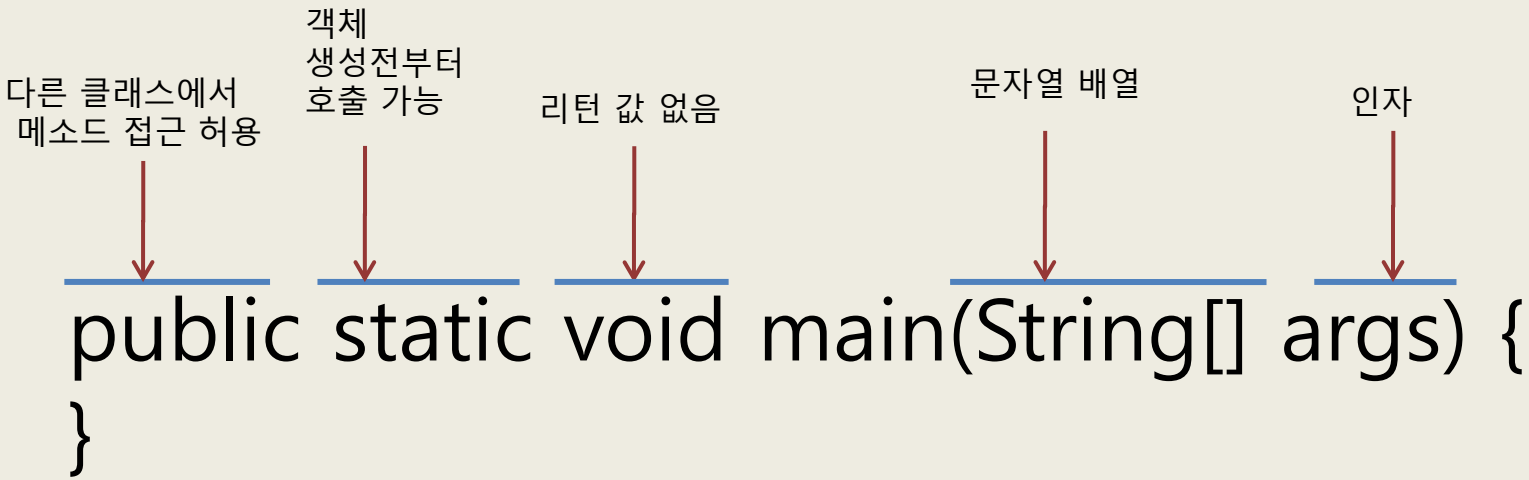
배열을 생성하고 각 원소 값을 출력하는 프로그램을 작성하시오. 배열 생성은 배열을 생성하여 각 원소의 인덱스 값으로 초기화하여 반환하는 메소드를 이용한다.

```
public class ReturnArray {
   static int[] makeArray() {
      int temp[] = new int[4];
      for (int i=0;i<temp.length;i++)
         temp[i] = i;
      return temp;
   }
   public static void main (String[] args) {
      int intArray [];
      intArray = makeArray();
      for (int i = 0; i < intArray.length; i++)
         System.out.println(intArray[i]);
   }
}
```

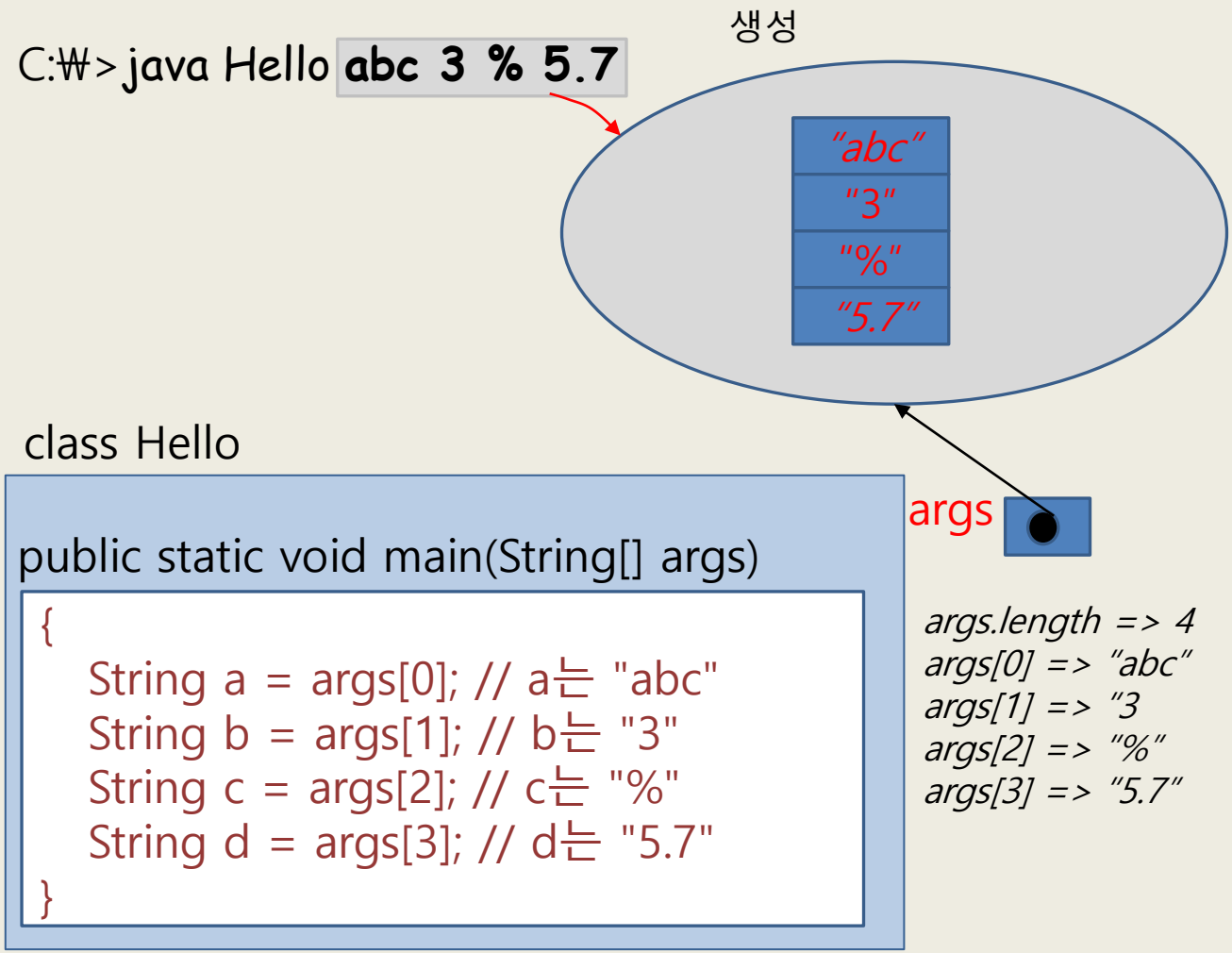
```
0
1
2
3
```

main( ) 메소드

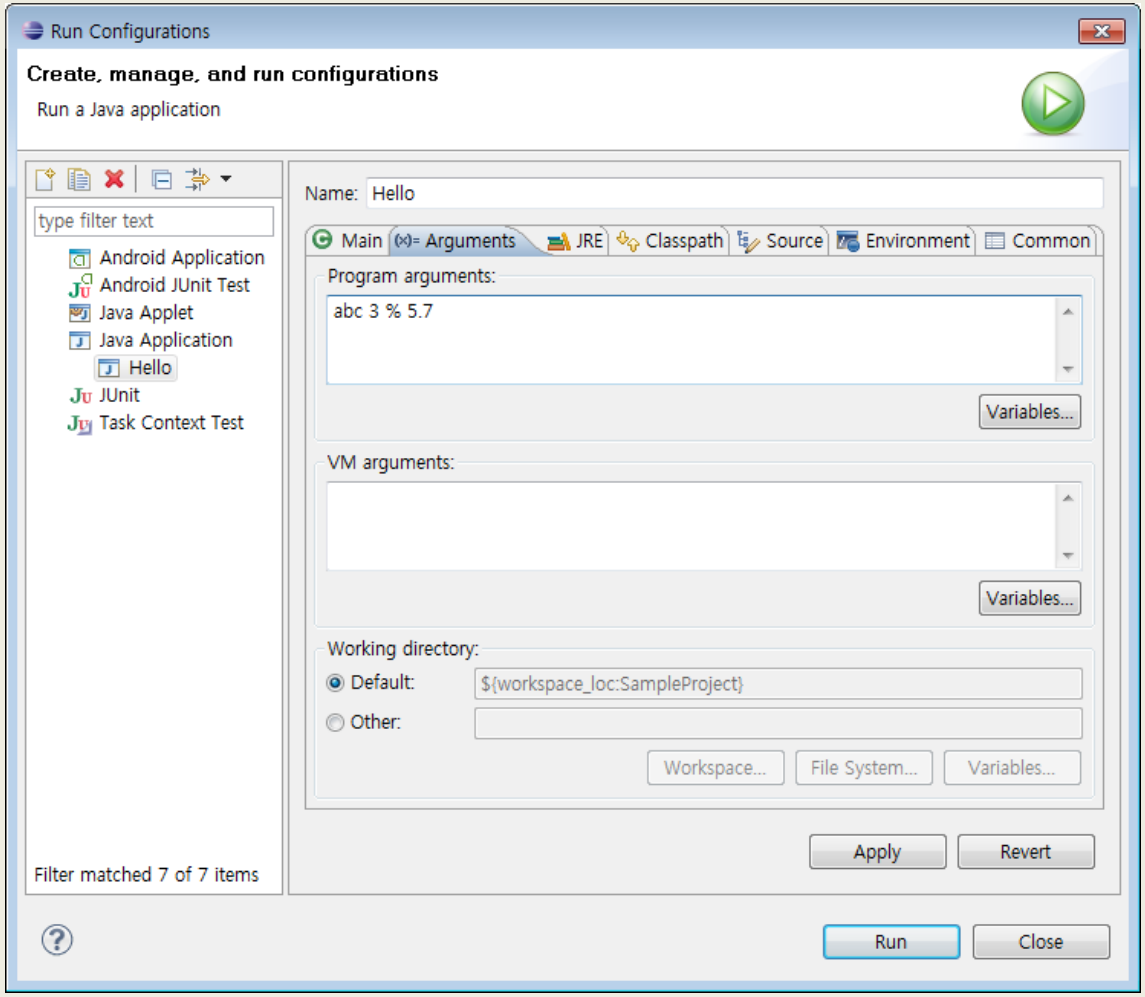
- main()의 메소드 특징
  - 자바 응용프로그램은 main()에서 부터 시작
  - public 속성
  - static 속성
  - 리턴 타입은 void
  - 인수는 문자열 배열(String [])로 전달



main(String[] args) 메소드의 인자 전달

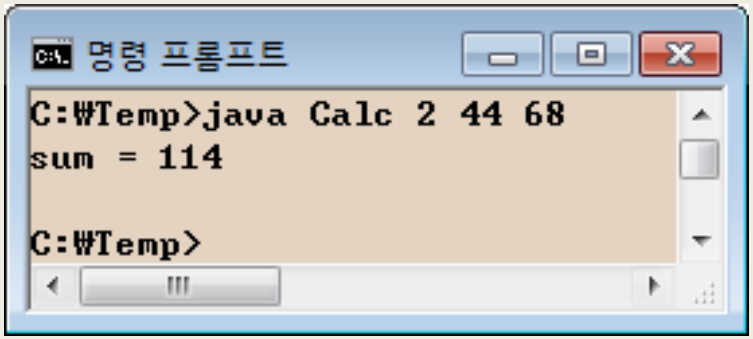


- Run 메뉴의 Run Configurations 항목에서 main() 메소드의 인자를 나열



main()의 인수 이용 예

```
public class Calc {
public static void main(String[] args) {
    int sum = 0;
    for(int i=0; i<args.length; i++) { // 명령행 인자의 개수만큼 반복
        int n = Integer.parseInt(args[i]); // 명령행 인자인 문자열을 정수로 변환
        sum += n; // 숫자를 합한다.
    }
    System.out.println("sum = " + sum);
}
}
```

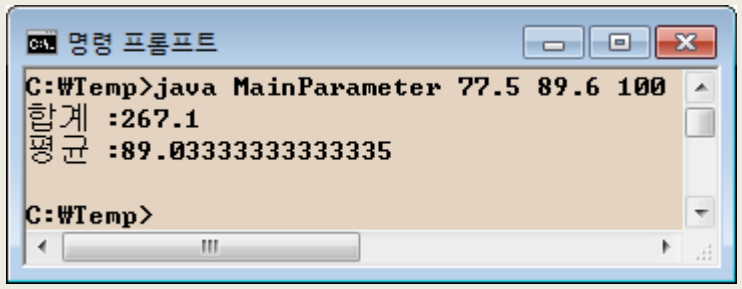


예제

실수를 main() 메소드 인자로 전달받아 평균값을 구하는 프로그램을 작성하시오.

```
public class MainParameter {
   public static void main (String[] args) {
      double sum = 0.0;

      for (int i=0; i<args.length; i++)
         sum += Double.parseDouble(args[i]);
      System.out.println("합계 :" + sum);
      System.out.println("평균 :" + sum/args.length);
   }
}
```



예외처리(Exception Handling)

- 예외(Exception)
  - 실행 중 발생하는 에러는 컴파일러가 알 수 없다.
  - 실행 중 에러 발생 시 예외를 발생시켜 예외 처리함
    - 예외를 처리하지 않으면 예외가 발생한 프로그램은 강제 종료

```
import java.util.Scanner;
public class ExceptionExample1 {
    public static void main (String[] args) {
        Scanner rd = new Scanner(System.in);
        int divisor = 0;
        int dividend = 0;

        System.out.print("나눔수를 입력하시오:");
        dividend = rd.nextInt();
        System.out.print("나눗수를 입력하시오:");
        divisor = rd.nextInt();
        System.out.println(dividend+"를 "+divisor+"로 나누면 몫은 "+dividend/divisor+"입니다.");
    }
}
```

```
나눔수를 입력하시오:100
나눗수를 입력하시오:0
Exception in thread "main" java.lang.ArithmeticException: / by zero
at ExceptionExample1.main(ExceptionExample1.java:12)
```

try-catch-finally 문

- 예외 처리문
  - **try-catch-finally**문 사용
  - **finally**는 생략 가능

```
try {
        예외가 발생할 가능성이 있는 실행문(try 블록)
}
catch (처리할 예외 타입 선언) {
        예외 처리문(catch 블록)
}
finally { // finally는 생략 가능
        예외 발생 여부와 상관없이 무조건 실행되는 문장(finally
        블록)
}
```

생략 가능

try블럭에서 예외가 발생하지 않은 정상적인 경우

```
try {
   ....
   실행문
   ....
}
catch (처리할 예외 타입 선언) {
   예외 처리문
}
finally {
   finally 블록 문
}
```

try블럭에서 예외가 발생한 경우

```
try {
   실행문
   ....
}
catch (처리할 예외 타입 선언) {
   예외 처리문
}
finally {
   finally 블록 문
}
```

예외 발생

자주 발생하는 예외 상황

예외	예외가 발생할 때
ArithmeticException	정수를 0으로 나눌 때 발생
NullPointerException	Null 레퍼런스 참조할 때 발생
ClassCastException	변환할 수 없는 타입으로 객체를 변환할 때 발생
OutOfMemoryException	메모리가 부족한 경우 발생
ArrayIndexOutOfBoundsException	배열의 범위를 벗어난 접근 시 발생
IllegalArgumentException	잘못된 인자 전달 시 발생
IOException	입출력 동작 실패 또는 인터럽트 시 발생
NumberFormatException	문자열이 나타내는 숫자와 일치하지 않는 타입의 숫자로 변환 시 발생

두 정수의 나눗셈에서 ArithmeticException을 처리하도록 수정된 예

try-catch문을 이용하여 정수를 0으로 나누려고 할 때 "0으로 나룰 수 없습니다."라는 경고 메시지를 출력하도록 프로그램을 작성하시오.

```
import java.util.Scanner;
public class ExceptionExample2 {
    public static void main (String[] args) {
        Scanner rd = new Scanner(System.in);
        int divisor = 0;
        int dividend = 0;

        System.out.print("나뉨수를 입력하시오:");
        dividend = rd.nextInt();
        System.out.print("나눗수를 입력하시오:");
        divisor = rd.nextInt();
        try {
            System.out.println(dividend+"를 "+divisor+"로 나누면 몫은
"+
                                    dividend/divisor+"입니다.");
        } catch (ArithmeticException e) {
            System.out.println("0으로 나눌 수 없습니다.");
        }
    }
}
```

```
나뉨수를 입력하시오:100
나눗수를 입력하시오:0
0으로 나눌 수 없습니다.
```

범위를 벗어난 배열의 접근 예

배열의 인덱스가 범위를 벗어날 때 발생하는 ArrayIndexOutOfBoundsException을 처리하는 프로그램을 작성하시오.

```
public class ArrayException {
   public static void main (String[] args) {
      int[] intArray = new int[5];
      intArray[0] = 0;
      try {
         for (int i = 0; i < 5; i++) {
            intArray[i+1] = i+1 + intArray[i];
            System.out.println("intArray["+i+"]"+"="+intArray[i]);
         }
      }
      catch (ArrayIndexOutOfBoundsException e) {
         System.out.println("배열의 인덱스가 범위를 벗어났습니다.");
      }
   }
}
```

```
intArray[0]=0
intArray[1]=1
intArray[2]=3
intArray[3]=6
배열의 인덱스가 범위를
벗어났습니다.
```

정수가 아닌 문자열을 정수로 변환할 때 예외 발생 예

문자열을 정수로 변환할 때 발생하는 NumberFormatException을 처리하는 프로그램을 작성하라.

```
public class NumException {
   public static void main (String[] args) {
      String[] stringNumber = {"23", "12", "998", "3.141592"};
      try {
         for (int i = 0; i < stringNumber.length; i++) {
            int j = Integer.parseInt(stringNumber[i]);
            System.out.println("숫자로 변환된 값은 " + j);
         }
      }
      catch (NumberFormatException e) {
         System.out.println("정수로 변환할 수 없습니다.");
      }
   }
}
```

```
숫자로 변환된 값은 23
숫자로 변환된 값은 12
숫자로 변환된 값은 998
정수로 변환할 수 없습니다.
```