



반복문과 배열 그리고 예외 처리

흐름제어

1. 순차 구조
2. 선택 구조
3. 반복 구조

흐름제어

1. 순차 구조



순차적으로 실행하는 구조
가장 기본적인 제어구조이다.

흐름제어

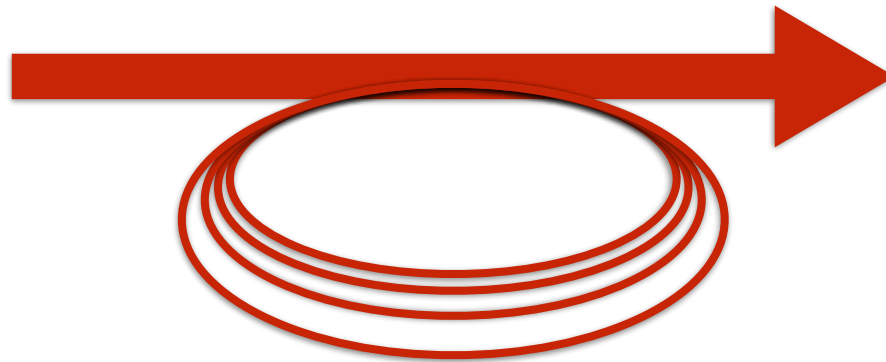
2. 선택 구조



조건을 검사하여 하나를 선택하여 실행하는 구조
논리적인 판단을 할 수 있도록 하는 제어구조

흐름제어

3. 반복 구조



어떤 조건을 만족할때까지 반복하는 구조
반복이 요구되는 제어구조에서 유용

반복의 필요성(C 코드)

```
#include <stdio.h>
```

```
int main(void)  
{
```

```
    int meter;
```

```
    meter = 1 * 1609;  
    printf("1 마일은 %d미터입니다\n", meter);  
    meter = 2 * 1609;  
    printf("2 마일은 %d미터입니다\n", meter);  
    meter = 3 * 1609;  
    printf("3 마일은 %d미터입니다\n", meter);  
    meter = 4 * 1609;  
    printf("4 마일은 %d미터입니다\n", meter);  
    return 0;
```

```
}
```

비슷한 내용
중복

반복문의 필요성(C 코드)

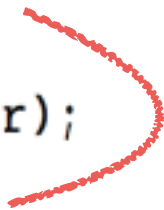
```
#include <stdio.h>
```

```
int main(void)  
{
```

```
    int meter;  
    int i=1;
```

```
    while(i<= 10){  
        meter = 1 * 1609;  
        printf("%d 마일은 %d미터입니다\n",i,meter);  
        i++;  
    }  
    return 0;  
}
```

간결함
코드의 수정이 쉬움



반복문의 특징

8

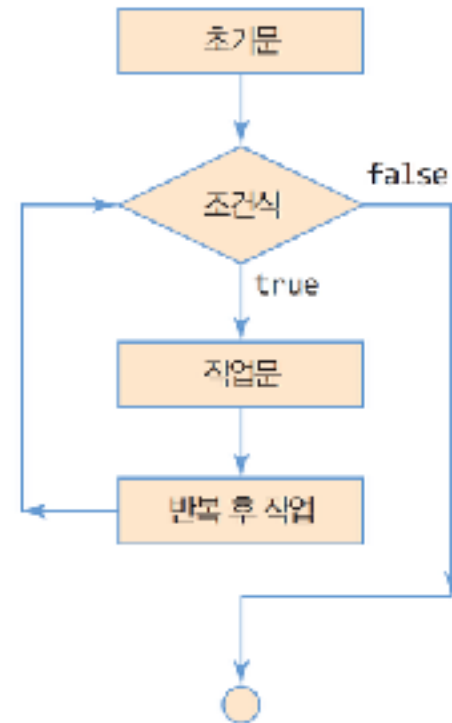
- 자바 반복문의 종류
 - ▣ for 문
 - ▣ while 문
 - ▣ do while 문



for 문의 구성

9

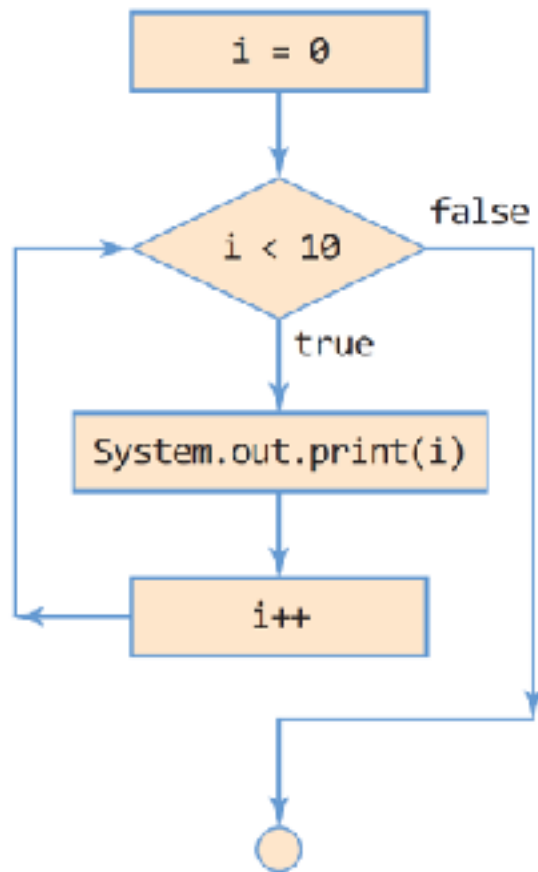
1 2 4
for (초기문; 조건식; 반복 후 작업) {
 ..작업문..
} 3



```
for(i=0; i<10; i++) { // i가 0~9까지 10번 반복
    System.out.print(i); // 0에서 9까지 출력
}
```

for 문의 실행 과정을 나타내는 순서도

10



```
for(i=0; i<10; i++) {  
    System.out.print(i);  
}
```

0123456789

for문의 예시

11

- 0에서 9까지 정수 출력

```
int i;  
for(i = 0; i < 10; i++) {  
    System.out.print(i);  
}
```

```
int i;  
for(i = 0; i < 10; i++)  
    System.out.print(i);
```

- 반복문에 변수 선언 가능

```
for(int i = 0; i < 10; i++) // 변수 i는 for문을 벗어나서 사용할 수 없음  
    System.out.print(i);
```

- 0에서 100까지의 합 구하기

```
int sum = 0;  
for(int i = 0; i <= 100; i++)  
    sum += i;
```

```
int sum;  
for(int i = 0, sum=0; i <= 100; i++)  
    sum += i;
```

```
int sum = 0;  
for(int i = 100; i >= 0; i--)  
    sum += i;
```

for문의 특이한 형태

12

```
for(초기작업; true; 반복후작업) { // 반복 조건이 true이면 무한 반복
.....
}
```

```
for(초기작업; ; 반복후작업) { // 반복조건이 비어 있으면 true로 간주, 무한 반복
.....
}
```

```
// 초기 작업과 반복후작업은 ','로 분리하여 여러 문장 나열 가능
for(i=0; i<10; i++, System.out.println(i)) {
.....
}
```

```
// for문 내에 변수 선언
for(int i=0; i<10; i++) { // 변수 i는 for문 내에서만 사용 가능
.....
}
```

예제 3-1 : for 문을 이용하여 1부터 10까지 합 출력

13

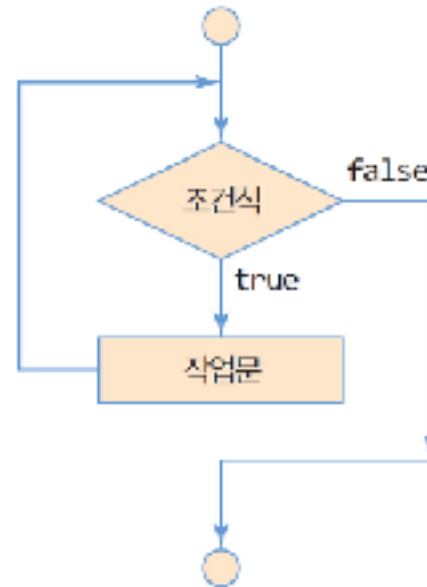
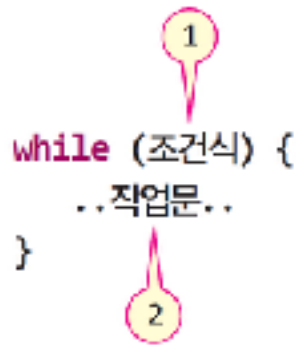
for문을 이용하여 1부터 10까지 덧셈을 표시하고 합을 구하시오.

```
public class ForSample {  
    public static void main(String[] args) {  
        int sum=0;  
  
        for(int i=1; i<=10; i++) { // 1~10까지 반복  
            sum += i;  
            System.out.print(i); // 더하는 수 출력  
  
            if(i<=9) // 1~9까지는 '+' 출력  
                System.out.print("+");  
            else { // i가 10인 경우  
                System.out.print("="); // '=' 출력하고  
                System.out.print(sum); // 덧셈 결과 출력  
            }  
        }  
    }  
}
```

1+2+3+4+5+6+7+8+9+10=55

while 문의 구성

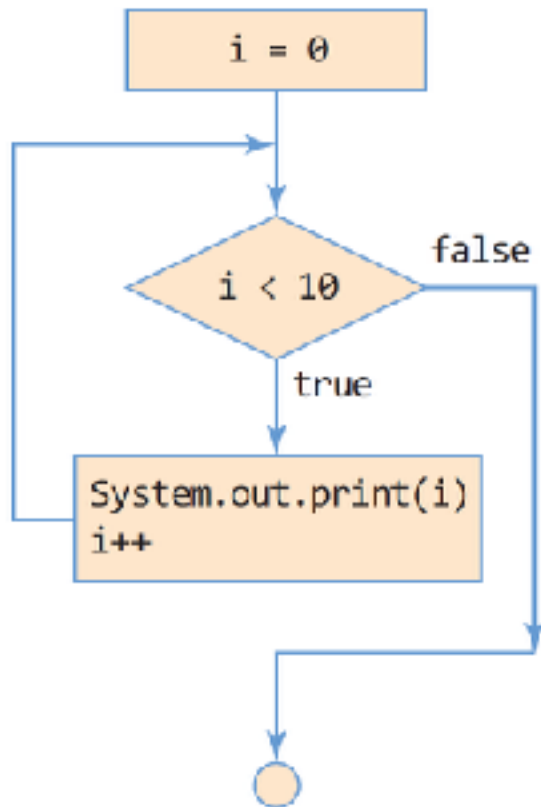
14



- 반복 조건이 true이면 반복, false이면 반복 종료
- 반복 조건이 없으면 컴파일 오류
- 처음부터 반복조건을 통과한 후 작업문 수행

while문의 실행 과정을 나타내는 순서도

15



```
i = 0;  
while(i<10) {  
    System.out.print(i);  
    i++;  
}
```

0123456789

예제 3-2 : -1이 입력될 때까지 입력된 수의 평균 구하기

while문을 이용하여 정수를 여러 개 입력 받고 평균을 출력하라. -1이 입력되면 입력을 종료한다.

```
import java.util.Scanner;
public class WhileSample {
    public static void main(String[] args) {
        int count=0; // count는 입력된 정수의 개수
        int sum=0; // sum은 합
        Scanner scanner = new Scanner(System.in);
        System.out.println("정수를 입력하고 마지막에 -1을 입력하세요.");

        int n = scanner.nextInt(); // 정수 입력
        while(n != -1) { // -1이 입력되면 while 문 벗어남
            sum += n;
            count++;
            n = scanner.nextInt(); // 정수 입력
        }
        if(count == 0) System.out.println("입력된 수가 없습니다.");
        else {
            System.out.print("정수의 개수는 " + count + "개이며 ");
            System.out.println("평균은 " + (double)sum/count + "입니다.");
        }
        scanner.close();
    }
}
```

정수를 입력하고 마지막에 -1을 입력하세요.

10 30 -20 40 -1

정수의 개수는 4개이며 평균은 15.0입니다.

-1은 마지막 입력을 뜻함

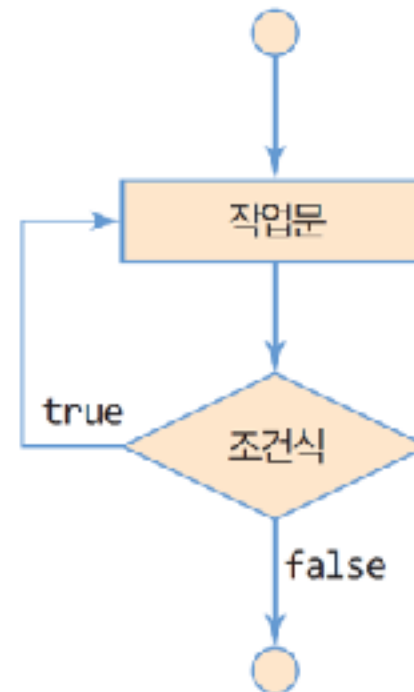
do-while 문의 구성

17

```
do {  
    ..작업문..  
} while(조건식);
```

1

2

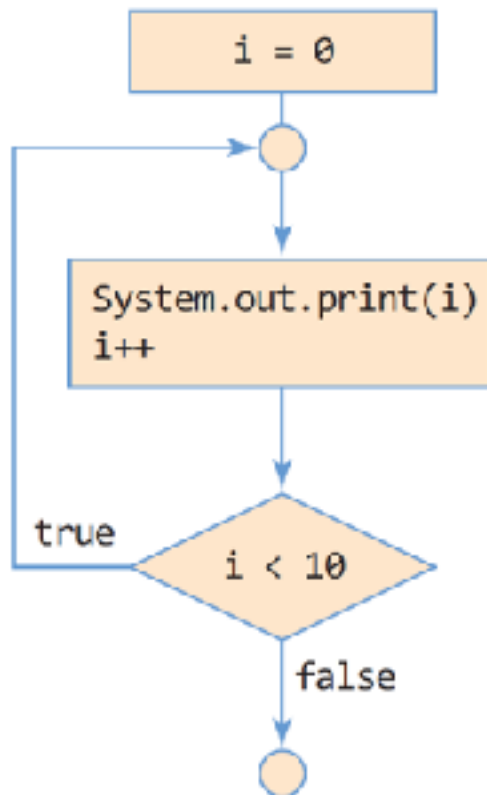


• 무조건 최소 한번 작업문 실행

- 반복 조건이 true이면 반복, false이면 반복 종료
- 반복 조건이 없으며 컴파일 오류

do-while문의 실행 과정을 나타내는 순서도

18



```
i = 0;  
do {  
    System.out.print(i);  
    i++;  
} while(i < 10);
```

0123456789

예제 3-3 : a-z까지 출력

19

do-while 문을 이용하여 'a'부터 'z'까지 출력하는 프로그램을 작성하시오.

```
public class DoWhileSample {  
    public static void main (String[] args) {  
        char c = 'a';  
  
        do {  
            System.out.print(c);  
            c = (char) (c + 1);  
        } while (c <= 'z');  
    }  
}
```

abcdefghijklmnopqrstuvwxyz

중첩 반복

20

- 중첩 반복
 - ▣ 반복문이 다른 반복문을 내포하는 구조
 - ▣ 이론적으로는 몇 번이고 중첩 반복 가능
 - ▣ 너무 많은 중첩 반복은 프로그램 구조를 복잡하게 하므로 2중 또는 3중 반복이 적당

```
for(int i=0; i<100; i++) { // 100개의 학교 성적을 모두 더한다.
```

```
.....
```

```
    for(int j=0; j<10000; j++) { // 10000명의 학생 성적을 모두 더한다.
```

```
        .....
```

```
        .....
```

```
    }
```

```
.....
```

```
}
```

10000명의 학생이 있는 100개 대학의 모든 학생 성적의 합을 구할 때,
for 문을 이용한 이중 중첩 구조

예제 3-4 : 2중 중첩을 이용한 구구단

21

2중 중첩 for문을 사용하여 구구단을 출력하는 프로그램을 작성하시오. 한 줄에 한 단씩 출력한다.

```
public class NestedLoop {  
    public static void main(String[] args) {  
        for(int i=1; i<10; i++) { // 1단에서 9단  
            for(int j=1; j<10; j++) { // 각 단의 구구셈 출력  
                System.out.print(i + "*" + j + "=" + i*j); // 구구셈 출력  
                System.out.print('\t'); // 하나씩 탭으로 띄기  
            }  
            System.out.println(); // 한 단이 끝나면 다음 줄로 커서 이동  
        }  
    }  
}
```

| | | | | | | | | |
|-------|--------|--------|--------|--------|--------|--------|--------|--------|
| 1*1=1 | 1*2=2 | 1*3=3 | 1*4=4 | 1*5=5 | 1*6=6 | 1*7=7 | 1*8=8 | 1*9=9 |
| 2*1=2 | 2*2=4 | 2*3=6 | 2*4=8 | 2*5=10 | 2*6=12 | 2*7=14 | 2*8=16 | 2*9=18 |
| 3*1=3 | 3*2=6 | 3*3=9 | 3*4=12 | 3*5=15 | 3*6=18 | 3*7=21 | 3*8=24 | 3*9=27 |
| 4*1=4 | 4*2=8 | 4*3=12 | 4*4=16 | 4*5=20 | 4*6=24 | 4*7=28 | 4*8=32 | 4*9=36 |
| 5*1=5 | 5*2=10 | 5*3=15 | 5*4=20 | 5*5=25 | 5*6=30 | 5*7=35 | 5*8=40 | 5*9=45 |
| 6*1=6 | 6*2=12 | 6*3=18 | 6*4=24 | 6*5=30 | 6*6=36 | 6*7=42 | 6*8=48 | 6*9=54 |
| 7*1=7 | 7*2=14 | 7*3=21 | 7*4=28 | 7*5=35 | 7*6=42 | 7*7=49 | 7*8=56 | 7*9=63 |
| 8*1=8 | 8*2=16 | 8*3=24 | 8*4=32 | 8*5=40 | 8*6=48 | 8*7=56 | 8*8=64 | 8*9=72 |
| 9*1=9 | 9*2=18 | 9*3=27 | 9*4=36 | 9*5=45 | 9*6=54 | 9*7=63 | 9*8=72 | 9*9=81 |

continue문

22

□ continue 문

- ▣ 반복문을 빠져 나가지 않으면서 다음 반복으로 진행

```
for(초기문; 조건식; 반복 후 작업) {  
    .....  
    continue;  
    .....  
}
```

분기

```
while(조건식) {  
    .....  
    continue;  
    .....  
}
```

조건식으로분
기

```
do {  
    .....  
    continue;  
    .....  
} while(조건식);
```

조건식으로분
기

예제 3-5 : continue 문을 이용하여 양수 합 구하기

23

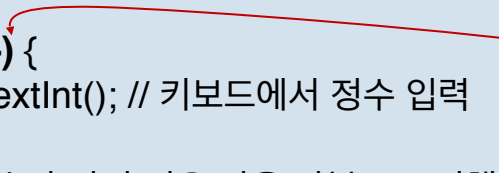
5개의 정수를 입력 받고 그 중 양수들만 합하여 출력하는 프로그램을 작성하라.

```
import java.util.Scanner;

public class ContinueExample {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.println("정수를 5개 입력하세요.");
        int sum=0;
        for(int i=0; i<5; i++) {
            int n = scanner.nextInt(); // 키보드에서 정수 입력
            if(n<=0)
                continue; // 양수가 아닌 경우 다음 반복으로 진행
            else
                sum += n; // 양수인 경우 덧셈
        }
        System.out.println("양수의 합은 " + sum);

        scanner.close();
    }
}
```



정수를 5개 입력하세요.

5

-2

6

10

-4

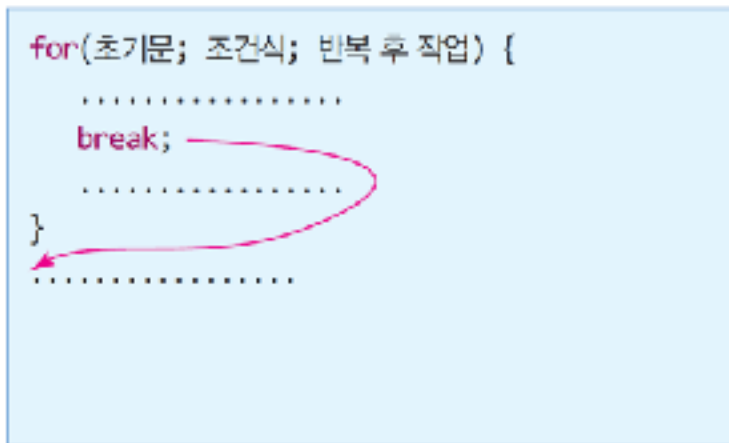
양수의 합은 21

break문

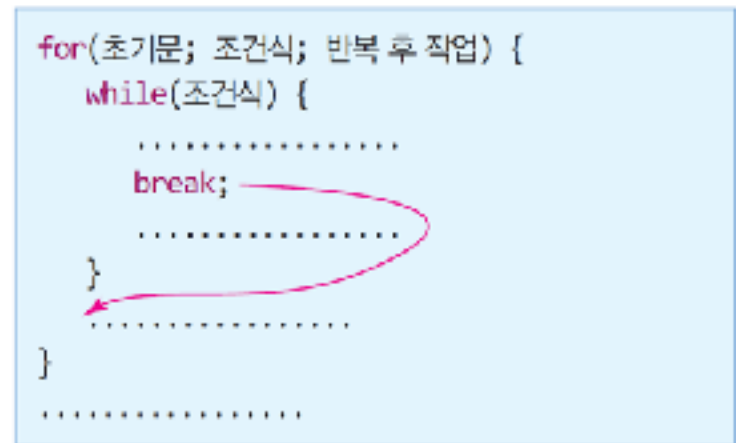
24

□ break 문

- ▣ 반복문 하나를 완전히 빠져 나갈 때 사용
 - 하나의 반복문만 벗어남
 - 중첩 반복의 경우 안쪽 반복문의 break 문이 실행되면 안쪽 반복문만 벗어남



(a) 현재 반복문 벗어나기



(b) 중첩 반복에서 안쪽 반복문만 벗어나는 경우

예제 3-6 : break 문을 이용하여 while 문 벗어나기

"exit"이 입력되면 while 문을 벗어나도록 break 문을 활용하는 프로그램을 작성하라.

```
import java.util.Scanner;

public class BreakExample {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.println("exit을 입력하면 종료합니다.");
        while(true) {
            System.out.print(">>");
            String text = scanner.nextLine();
            if(text.equals("exit")) // "exit"이 입력되면 반복 종료
                break; // while 문을 벗어남
        }
        System.out.println("종료합니다...");

        scanner.close();
    }
}
```

문자열 비교 시 equals()사용

exit을 입력하면 종료합니다.

>>edit

>>exit

종료합니다...

배열이란?

26

□ 배열(array)

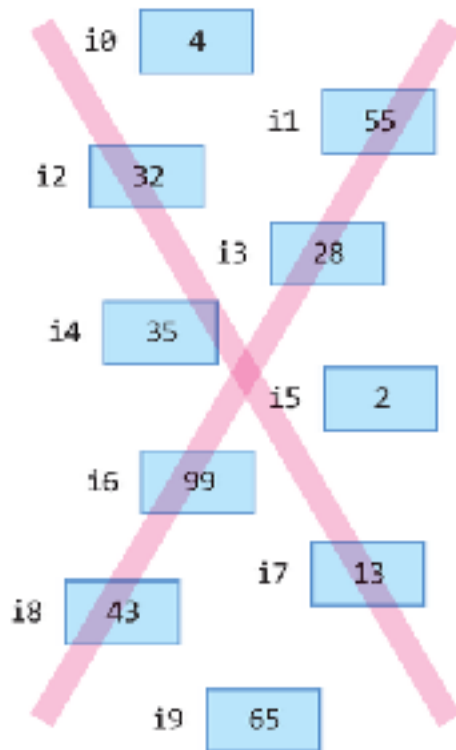
- ▣ 인덱스와 인덱스에 대응하는 데이터들로 이루어진 자료 구조
 - 배열을 이용하면 한 번에 많은 메모리 공간 할당 가능
- ▣ 같은 타입의 데이터들이 순차적으로 저장
 - 인덱스를 이용하여 원소 데이터 접근
 - 반복문을 이용하여 처리하기에 적합
- ▣ 배열 인덱스
 - 0부터 시작
 - 인덱스는 배열의 시작 위치에서부터 데이터가 있는 상대 위치

자바 배열의 필요성과 모양

27

(1) 10개의 정수형 변수를 사용하는 경우

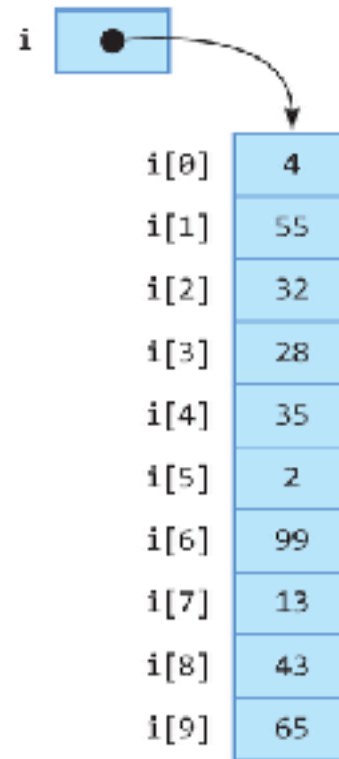
```
int i0, i1, i2, i3, i4, i5, i6, i7, i8, i9;
```



```
sum = i0+i1+i2+i3+i4+i5+i6+i7+i8+i9;
```

(2) 10개의 정수로 구성된 배열을 사용하는 경우

```
int i[] = new int[10];
```



```
for(sum=0, n=0; n<10; n++)  
    sum += i[n];
```

일차원 배열 만들기

28

배열 선언과 배열 생성의 두 단계 필요

배열 선언

```
int    intArray[];  
char   charArray[];
```

또는

```
int[]   intArray;  
char[]  charArray;
```

배열 생성

```
intArray = new int[10];  
charArray = new char[20];
```

또는

```
int intArray[] = new int[10];  
char charArray[] = new char[20];
```

선언과 함께 초기화

배열 선언 시 값 초기화

```
int intArray[] = {0,1,2,3,4,5,6,7,8,9}; // 초기화된 값의 개수(10)만큼의 배열 생성
```

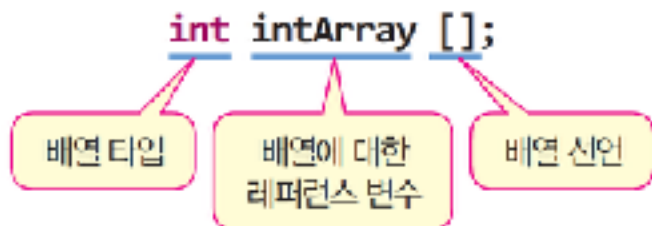
잘못된 배열 선언

```
int intArray[10]; // 컴파일 오류. 배열의 크기를 지정하면 안됨
```

레퍼런스 변수와 배열

29

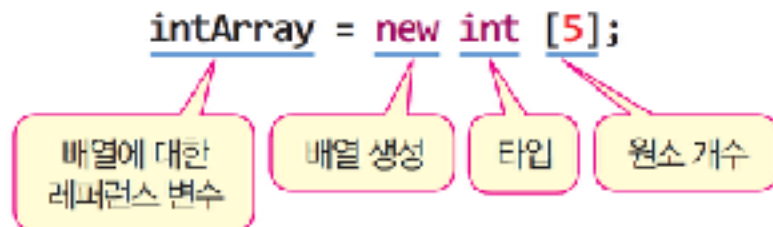
(1) 배열에 대한 레퍼런스 변수 `intArray` 선언



intArray



(2) 배열 생성



intArray



intArray[0]

intArray[1]

intArray[2]

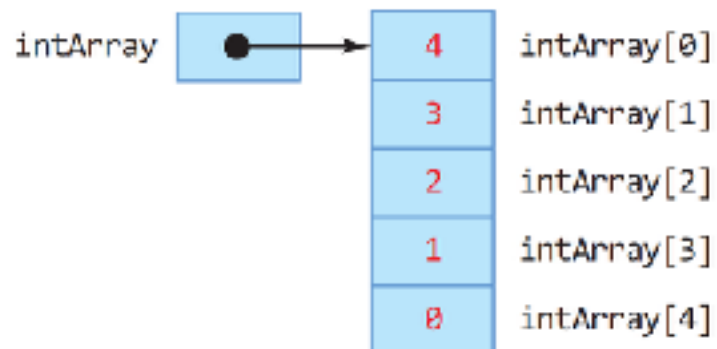
intArray[3]

intArray[4]

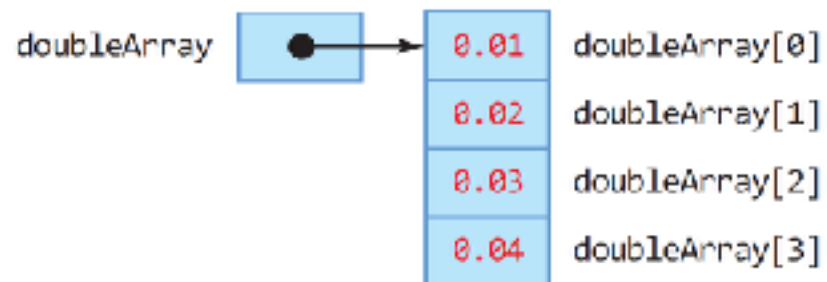
배열을 초기화하면서 생성한 결과

30

```
int intArray[] = {4, 3, 2, 1, 0};
```



```
double doubleArray[] = {0.01, 0.02, 0.03, 0.04};
```



배열 인덱스와 원소 접근

31

□ 배열 원소 접근

▣ 배열 변수명과 [] 사이에 원소의 인덱스를 적어 접근

- 배열의 인덱스는 0부터 시작
- 배열의 마지막 항목의 인덱스는 (배열 크기 - 1)

```
int intArray = new int[5]; // 원소가 5개인 배열 생성. 인덱스는 0~4까지 가능
intArray[0] = 5; // 원소 0에 5 저장
intArray[3] = 6; // 원소 3에 6 저장
int n = intArray[3]; // 원소 3의 값을 읽어 n에 저장. n은 6이 됨
```

▣ 인덱스의 범위



```
n = intArray[-2]; // 실행 오류. 인덱스로 음수 사용 불가
n = intArray[5]; // 실행 오류. 5는 인덱스의 범위(0~4)를 넘었음
```

▣ 반드시 배열 생성 후 접근

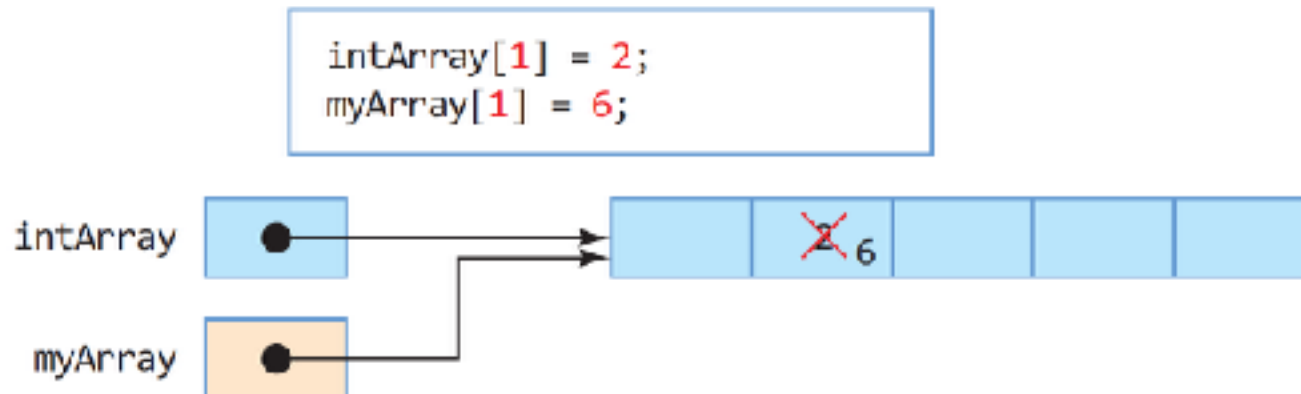
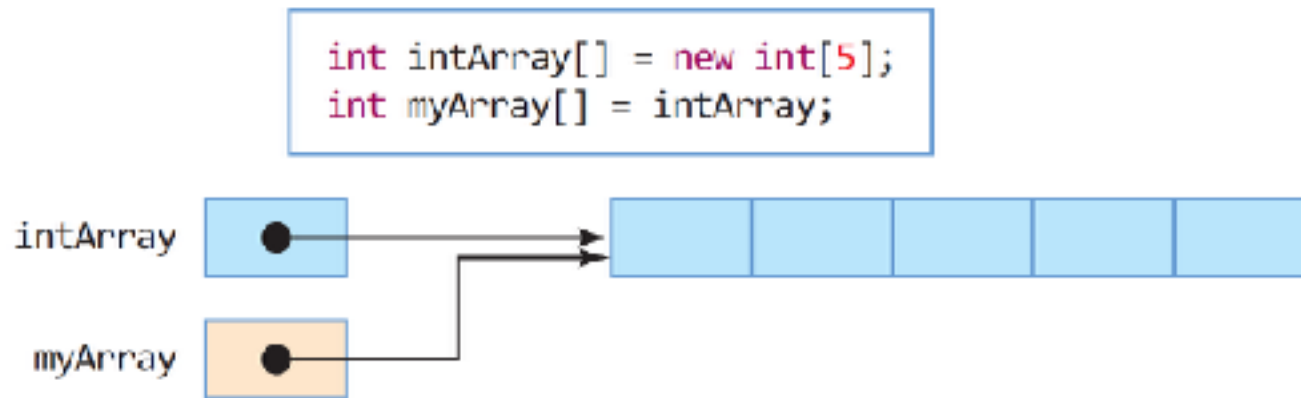


```
int intArray [];
intArray[1] = 8; // 오류, 생성 되지 않은 배열 사용
```

레퍼런스 치환과 배열 공유

32

- 하나의 배열을 다수의 레퍼런스가 참조 가능



예제 3-7 : 배열에 입력받은 수 중 제일큰수 찾기

33

양수 5개를 입력 받아 배열에 저장하고, 제일 큰 수를 출력하는 프로그램을 작성하라.

```
import java.util.Scanner;

public class ArrayAccess {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        int intArray[] = new int[5]; // 배열 생성

        int max=0;    // 현재 가장 큰 수
        System.out.println("양수 5개를 입력하세요.");
        for(int i=0; i<5; i++) {
            intArray[i] = scanner.nextInt(); // 입력받은 정수를 배열에 저장
            if(intArray[i] > max) // intArray[i]가 현재 가장 큰 수보다 크면
                max = intArray[i]; // intArray[i]를 max로 변경
        }
        System.out.print("가장 큰 수는 " + max + "입니다.");

        scanner.close();
    }
}
```

양수 5개를 입력하세요.

1

39

78

100

99

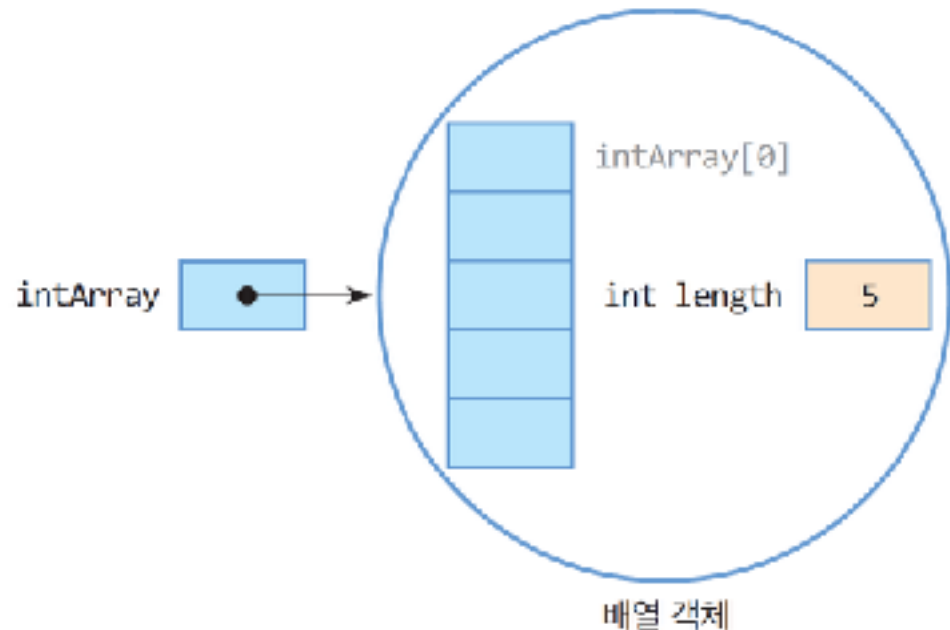
가장 큰 수는 100입니다.

배열의 크기, length 필드

34

- 배열은 자바에서 객체로 관리
 - 배열 객체 내에 length 필드는 배열의 크기를 나타냄

```
int intArray[];  
intArray = new int[5];  
  
int size = intArray.length;  
// size는 5
```



예제 3-8 : 배열 원소의 평균 구하기

35

배열의 length 필드를 이용하여 배열 크기만큼 정수를 입력 받고 평균을 구하는 프로그램을 작성하라.

```
import java.util.Scanner;

public class ArrayLength {
    public static void main(String[] args) {
        int intArray[] = new int[5]; // 배열의 선언과 생성
        int sum=0;

        Scanner scanner = new Scanner(System.in);
        System.out.print(intArray.length + "개의 정수를 입력하세요>>");
        for(int i=0; i<intArray.length; i++)
            intArray[i] = scanner.nextInt(); // 키보드에서 입력받은 정수 저장

        for(int i=0; i<intArray.length; i++)
            sum += intArray[i]; // 배열에 저장된 정수 값을 더하기

        System.out.print("평균은 " + (double)sum/intArray.length);
        scanner.close();
    }
}
```

5개의 정수를 입력하세요>> 2 3 4 5 9
평균은 4.6

배열과 for-each 문

36

▣ for-each 문

- 배열이나 나열(enumeration)의 각 원소를 순차적으로 접근하는데 유용한 for 문

```
int[] num = { 1,2,3,4,5 };  
int sum = 0;  
for (int k : num) // 반복될 때마다 k는 num[0], num[1], ..., num[4] 값으로 설정  
    sum += k;  
System.out.println("합은 " + sum);
```

합은 15

```
String names[] = { "사과", "배", "바나나", "체리", "딸기", "포도" };  
for (String s : names) // 반복할 때마다 s는 names[0], names[1], ..., names[5] 로 설정  
    System.out.print(s + " ");
```

사과 배 바나나 체리 딸기 포도

```
enum Week { 월, 화, 수, 목, 금, 토, 일 }  
for (Week day : Week.values()) // 반복될 때마다 day는 월, 화, 수, 목, 금, 토, 일로 설정  
    System.out.print(day + "요일 ");
```

월요일 화요일 수요일 목요일 금요일 토요일 일요일

예제 3-9 : for-each 문 활용

37

for-each 문을
활용하는
사례를 보자.

```
public class foreachEx {  
    enum Week { 월, 화, 수, 목, 금, 토, 일 }  
  
    public static void main(String[] args) {  
        int [] n = { 1,2,3,4,5 };  
        String names[] = { "사과", "배", "바나나", "체리", "딸기", "포도" };  
  
        int sum = 0;  
        // 아래 for-each에서 k는 n[0], n[1], ..., n[4]로 반복  
        for (int k : n) {  
            System.out.print(k + " "); // 반복되는 k 값 출력  
            sum += k;  
        }  
        System.out.println("합은" + sum);  
  
        // 아래 for-each에서 s는 names[0], names[1], ..., names[5]로 반복  
        for (String s : names)  
            System.out.print(s + " ");  
        System.out.println();  
  
        // 아래 for-each에서 day는 월, 화, 수, 목, 금, 토, 일 값으로 반복  
        for (Week day : Week.values())  
            System.out.print(day + "요일 ");  
        System.out.println();  
    }  
}
```

1 2 3 4 5 합은 15

사과 배 바나나 체리 딸기 포도

월요일 화요일 수요일 목요일 금요일 토요일 일요일

2차원 배열

38

□ 2차원 배열 선언

```
int    intArray[][];  
char   charArray[][];  
double doubleArray[][];
```

또는

```
int[][] intArray;  
char[][] charArray;  
double[][] doubleArray;
```

□ 2차원 배열 생성

```
intArray = new int[2][5];  
charArray = new char[5][5];  
doubleArray = new double[5][2];
```

또는

```
int    intArray[] = new int[2][5];  
char   charArray[] = new char[5][5];  
double doubleArray[] = new double[5][2];
```

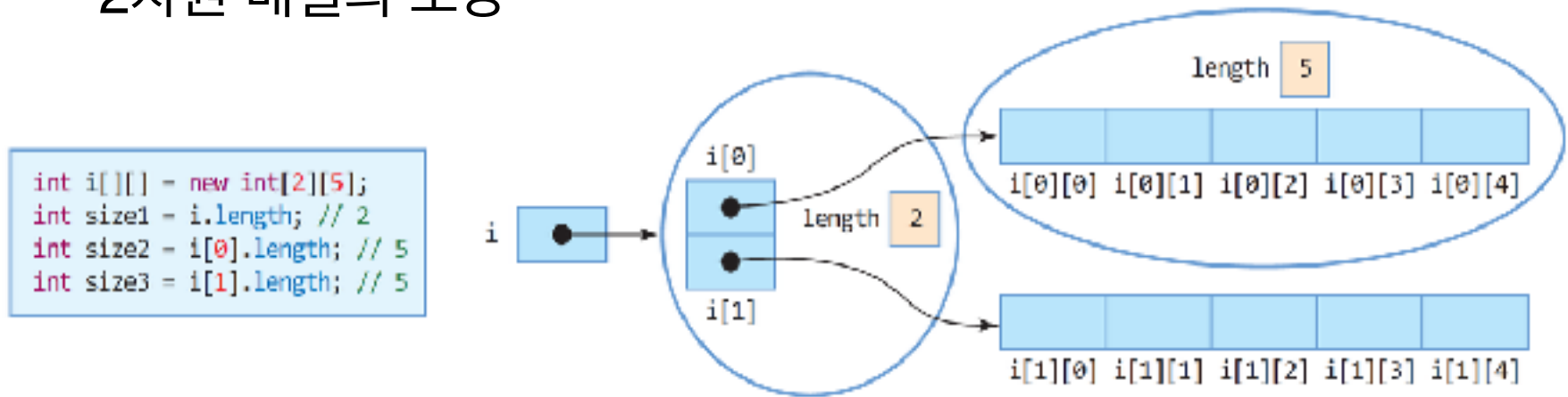
□ 2차원 배열 선언, 생성, 초기화

```
int intArray[][] = {{0,1,2},{3,4,5},{6,7,8}};  
char charArray[][] = {{'a', 'b', 'c'},{'d', 'e', 'f'}};  
double doubleArray[][] = {{0.01, 0.02}, {0.03, 0.04}};
```

2차원 배열의 모양과 length 필드

39

□ 2차원 배열의 모양



□ 2차원 배열의 length

- `i.length` -> 2차원 배열의 행의 개수로서 2
- `i[n].length`는 `n`번째 행의 열의 개수
 - `i[0].length` -> 0번째 행의 열의 개수로서 5
 - `i[1].length` -> 1번째 행의 열의 개수로서 5

예제 3-10 : 2차원 배열로 4년 평점 구하기

40

2차원 배열에 학년별로 1,2학기 성적으로 저장하고, 4년간 전체 평점 평균을 출력하라.

```
public class ScoreAverage {
    public static void main(String[] args) {
        double score[][] = {{3.3, 3.4},    // 1학년 1, 2학기 평점
                           {3.5, 3.6},    // 2학년 1, 2학기 평점
                           {3.7, 4.0},    // 3학년 1, 2학기 평점
                           {4.1, 4.2}}; // 4학년 1, 2학기 평점

        double sum=0;
        for(int year=0; year<score.length; year++) // 각 학년별로 반복
            for(int term=0; term<score[year].length; term++) // 각 학년의 학기별로 반복
                sum += score[year][term]; // 전체 평점 합

        int n=score.length; // 배열의 행 개수, 4
        int m=score[0].length; // 배열의 열 개수, 2
        System.out.println("4년 전체 평점 평균은 " + sum/(n*m));
    }
}
```

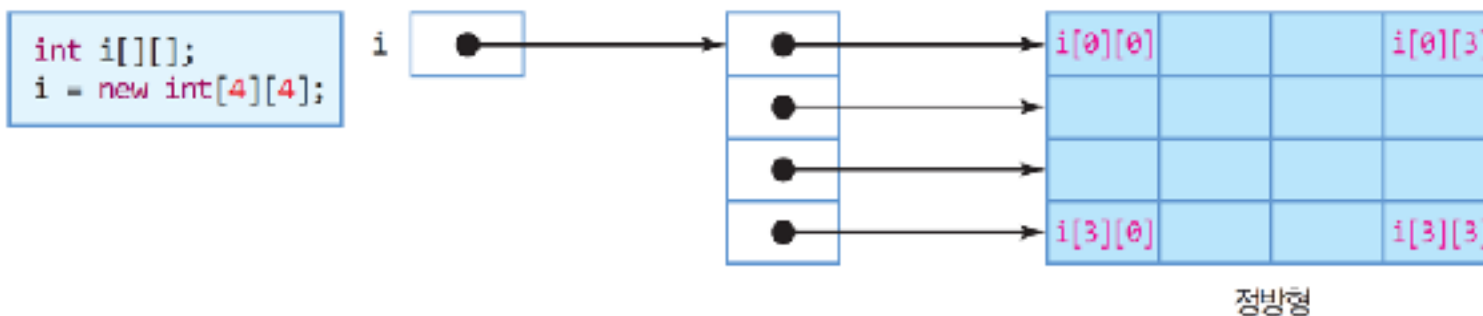
4년 전체 평점 평균은 3.725

비정방형 배열

41

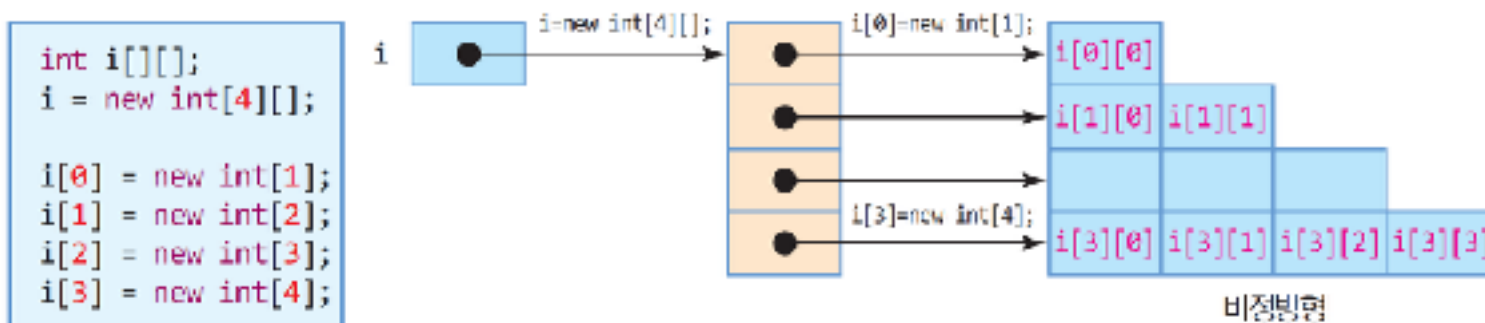
정방형 배열

- 각 행의 열의 개수가 같은 배열



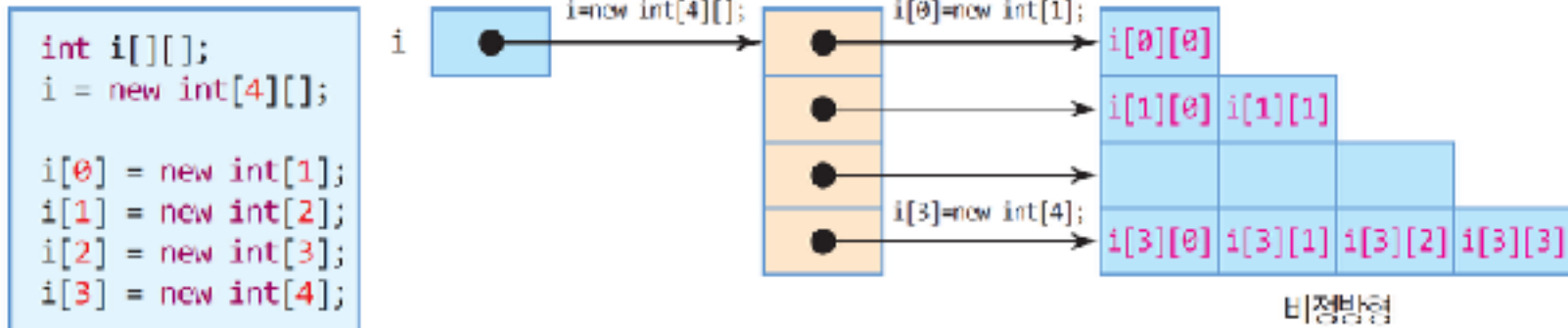
비정방형 배열

- 각 행의 열의 개수가 다른 배열
- 비정방형 배열의 생성



비정방형 배열의 length

42



□ 비정방형 배열의 length

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

예제 3-11 : 비정방형 배열의 생성과 접근

43

다음 그림과 같은 비정방형 배열을 만들어 값을 초기화하고 출력하시오.

| | | |
|----|----|----|
| 10 | 11 | 12 |
| 20 | 21 | |
| 30 | 31 | 32 |
| 40 | 41 | |

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

메소드에서 배열 리턴

44

- 메소드의 배열 리턴
 - ▣ 배열의 레퍼런스 리턴
 - ▣ 메소드의 리턴 타입
 - 메소드의 리턴 타입과 리턴 받는 배열 타입과 일치
 - 리턴 타입에 배열의 크기를 지정하지 않음

리턴 타입 메소드 이름

```
int[] makeArray() {  
    int temp[] = new int[4];  
    return temp;  
}
```

배열 리턴

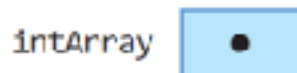
```
int [] intArray;  
intArray = makeArray();
```

배열 리턴 과정

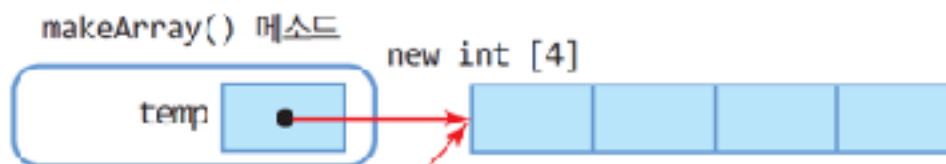
45

```
int[] makeArray() {  
    int temp[] = new int[4];  
    return temp;  
}
```

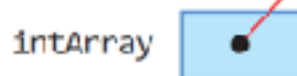
(1) `int[] intArray;`



(2) `makeArray();` // 메소드 실행



(3) `intArray`에 `temp` 값 치환



(4) `intArray[0] = 5;`
...
`intArray[3] = 8;`



예제 3-12 : 배열 리턴

46

정수 4개를 가지는 일차원 배열을 생성하고 1,2,3,4로 초기화한 다음, 배열을 리턴하는 `makeArray()`를 작성하고, 이 메소드로부터 배열을 전달받아 값을 출력하는 프로그램을 작성하라.

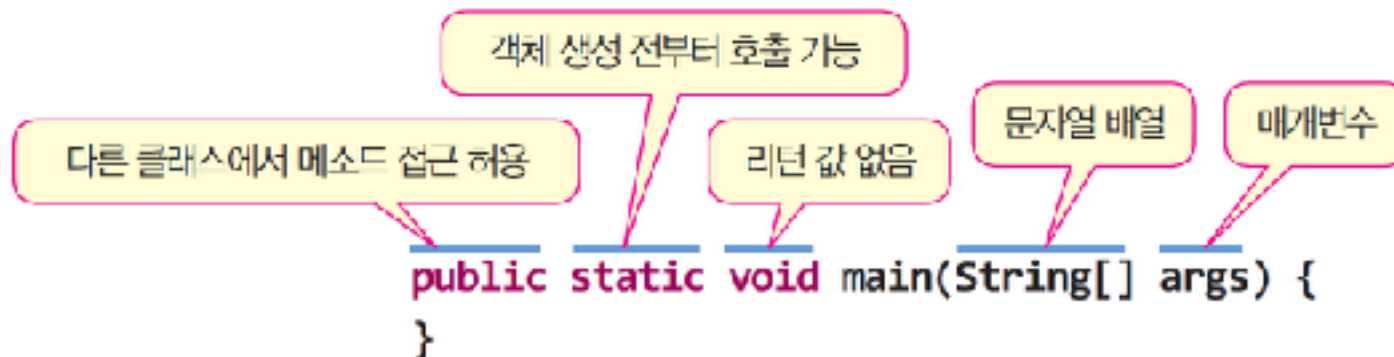
```
public class ReturnArray {  
  
    static int[] makeArray() { // 정수형 배열을 리턴하는 메소드  
        int temp[] = new int[4]; // 배열 생성  
        for (int i=0; i<temp.length; i++)  
            temp[i] = i; // 배열의 원소를 0, 1, 2, 3으로 초기화  
        return temp; // 배열 리턴  
    }  
  
    public static void main (String[] args) {  
        int intArray[]; // 배열 레퍼런스 변수 선언  
        intArray = makeArray(); // 메소드로부터 배열 전달받음  
        for (int i=0; i<intArray.length; i++)  
            System.out.print(intArray[i] + " "); // 배열 모든 원소 출력  
    }  
}
```

0 1 2 3

main() 메소드

47

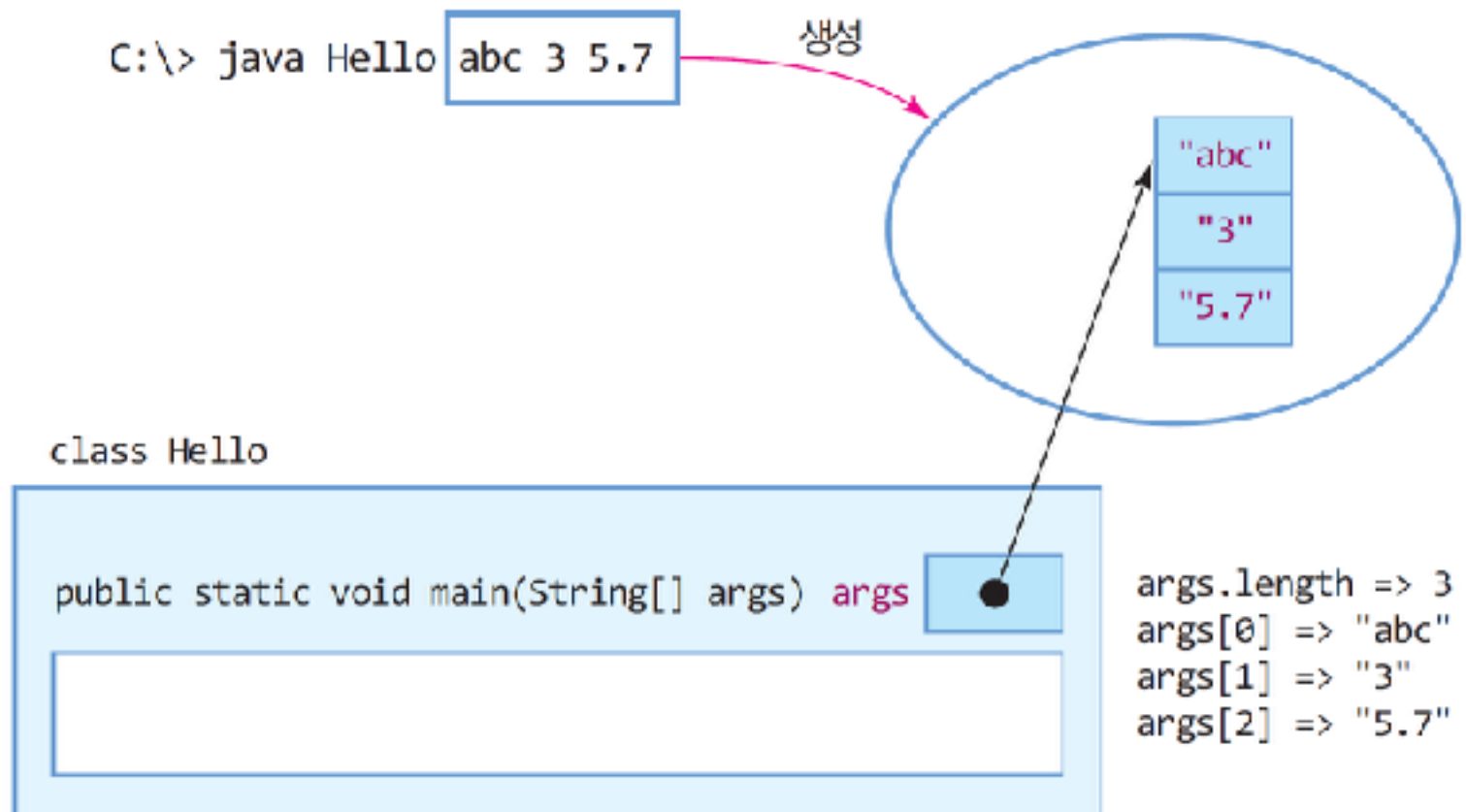
- ▣ main()은 자바 응용프로그램의 실행 시작 메소드
- ▣ main()의 원형
 - 반드시 static
 - 반드시 public
 - 반드시 void
 - 반드시 매개 변수 타입은 문자열 배열



main(string [] args) 메소드의 인자 전달

48

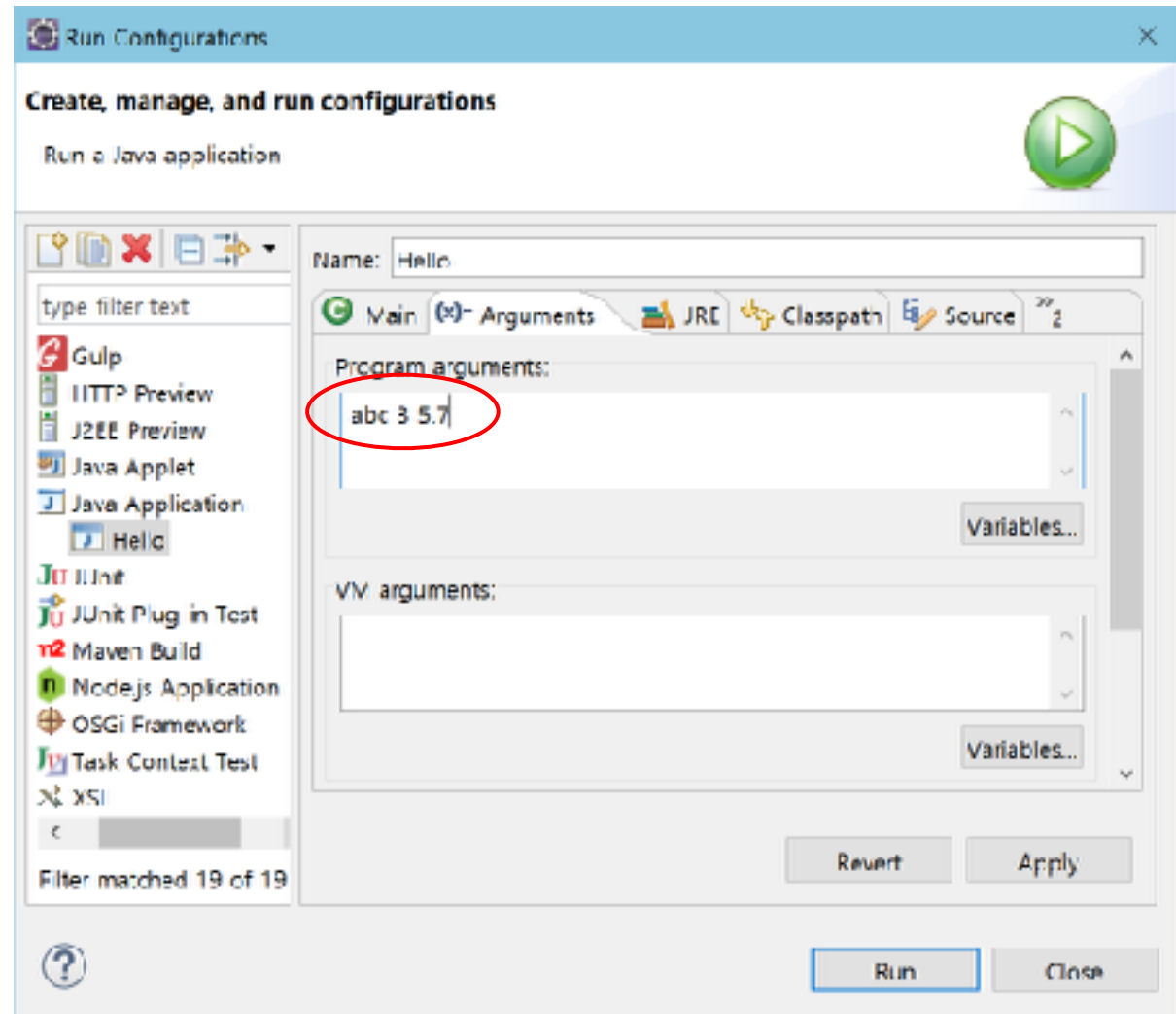
- main() 메소드의 매개변수로 명령행 인자의 전달



이클립스에서 main() 메소드의 인자전달

49

Run 메뉴의
Run Configurations
항목에서
main() 메소드의
인자 나열



예제 3-13 : main()에서 명령행 인자의 합 계산

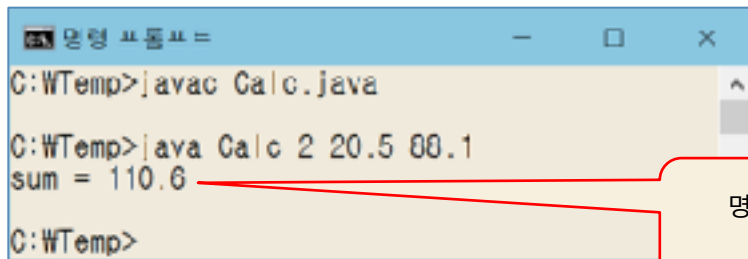
사용자가 명령행에 입력한 여러 개의 실수를 main() 메소드에서 전달받아 합을 구하는 프로그램을 작성하라.

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

        for (int i=0; i<args.length; i++) // 인자 개수만큼 반복
            sum += Double.parseDouble(args[i]); // 문자열을 실수(double 타입)로 변환하여 합산

        System.out.println("합계 :" + sum);
    }
}
```

Double.parseDouble()는 매개변수로
주어진 문자열을 실수로 변환.
Double.parseDouble("20.5") 은 실수
20.5 리턴



```
C:\WTemp>javac Calc.java

C:\WTemp>java Calc 2 20.5 88.1
sum = 110.6

C:\WTemp>
```

명령행 인자 2 20.5 88.1을
모두 합하여 110.6 출력

자바의 예외 처리

51

- 컴파일 오류
 - ▣ 문법에 맞지 않게 작성된 코드
 - ▣ 컴파일할 때 발견
- 예외(Exception)
 - ▣ 오동작이나 결과에 악영향을 미칠 수 있는 실행 중 발생한 오류
 - 정수를 0으로 나누는 경우
 - 배열보다 큰 인덱스로 배열의 원소를 접근하는 경우
 - 존재하지 않는 파일을 읽으려고 하는 경우
 - 정수 입력을 기다리는 코드가 실행되고 있을 때, 문자가 입력된 경우
 - ▣ 자바에서 예외 처리 가능
 - 예외 발생 -> 자바 플랫폼 인지 -> 응용프로그램에서 전달
 - 응용프로그램이 예외를 처리하지 않으면, 응용프로그램 강제 종료

예제 3-14 : 0으로 나누기 예외 발생으로 프로그램이 강제 종료되는 경우

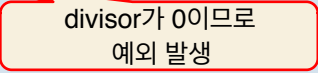
52

두 정수를 입력 받아 나눗셈을 하고 몫을 구하는 프로그램 코드이다. 사용자가 나누는 수에 0을 입력하면 `ArithmeticException` 예외가 발생하여 프로그램이 강제 종료된다.

```
import java.util.Scanner;

public class DivideByZero {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int dividend; // 나뉘는수
        int divisor; // 나눗수

        System.out.print("나뉘는수를 입력하시오:");
        dividend = scanner.nextInt(); // 나뉘는수 입력
        System.out.print("나눗수를 입력하시오:");
        divisor = scanner.nextInt(); // 나눗수 입력
        System.out.println(dividend+"를 "+ divisor + "로 나누면 몫은 " +
            dividend/divisor + "입니다.");
        scanner.close();
    }
}
```



divisor가 0이므로
예외 발생

나뉘는수를 입력하시오:100

나눗수를 입력하시오:0

Exception in thread "main" `java.lang.ArithmeticException: / by zero`
at DivideByZero.main(`ExceptionExample1.java:14`)

예외 처리, try-catch-finally 문

53

- 예외 처리
 - ▣ 예외가 발생할 때 대응하는 응용프로그램 코드
 - ▣ try-catch-finally 문 사용
 - finally 블록은 생략 가능

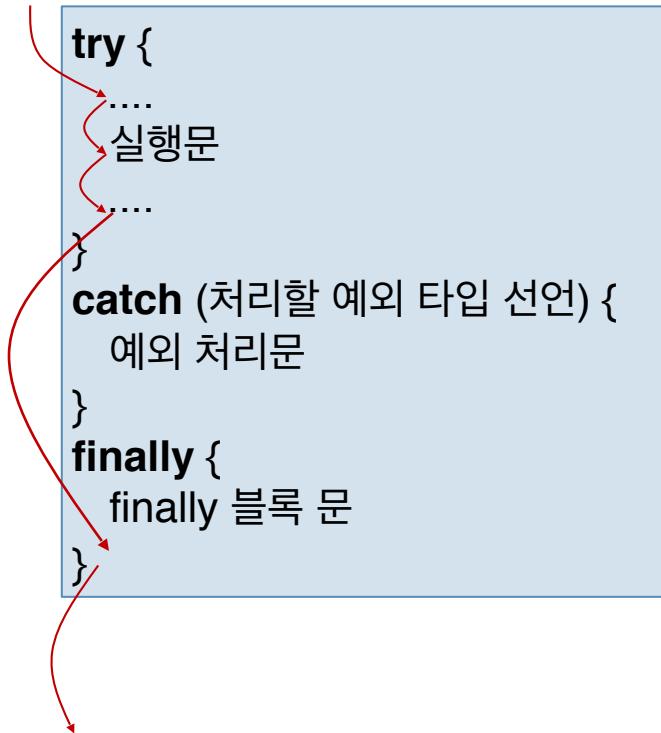
생략
가능

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

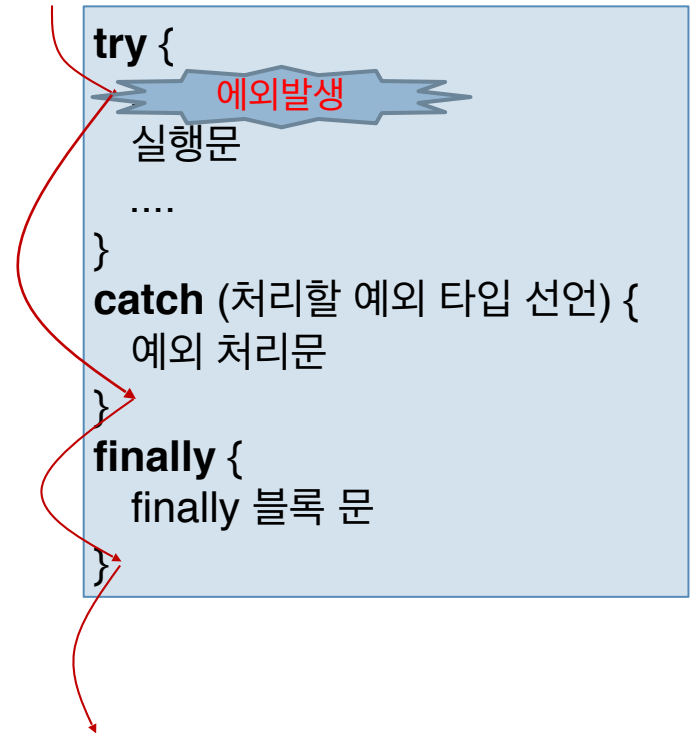
예외에 따른 제어의 흐름

54

try블록에서 예외가 발생하지 않은 정상적인 경우



try블록에서 예외가 발생한 경우



자바의 예외 클래스

55

□ 자주 발생하는 예외

| 예외 타입(예외 클래스) | 예외 발생 경우 | 패키지 |
|--------------------------------|--|-----------|
| ArithmeticException | 정수를 0으로 나눌 때 발생 | java.lang |
| NullPointerException | null 레퍼런스를 참조할 때 발생 | java.lang |
| ClassCastException | 변환할 수 없는 타입으로 객체를 변환할 때 발생 | java.lang |
| OutOfMemoryError | 메모리가 부족한 경우 발생 | java.lang |
| ArrayIndexOutOfBoundsException | 배열의 범위를 벗어난 접근 시 발생 | java.lang |
| IllegalArgumentException | 잘못된 인자 전달 시 발생 | java.lang |
| IOException | 입출력 동작 실패 또는 인터럽트 시 발생 | java.io |
| NumberFormatException | 문자열이 나타내는 숫자와 일치하지 않는 타입의 숫자로 변환 시 발생 | java.lang |
| InputMismatchException | Scanner 클래스의 nextInt()를 호출하여 정수로 입력받고자 하였지만, 사용자가 'a' 등과 같이 문자를 입력한 경우 | java.util |

예제 3-15 : 0으로 나눌 때 발생하는 ArithmeticException 예외 처리

56

try-catch 블록을 이용하여 예제 3-14를 수정하여, 정수를 0으로 나누는 경우에 "0으로 나눌 수 없습니다!"를 출력하고 다시 입력 받는 프로그램을 작성하라.

```
import java.util.Scanner;

public class DevideByZeroHandling {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        while(true) {
            System.out.print("나뉘수를 입력하시오:");
            int dividend = scanner.nextInt(); // 나뉘수 입력
            System.out.print("나눗수를 입력하시오:");
            int divisor = scanner.nextInt(); // 나눗수 입력
            try {
                System.out.println(dividend + "를 " + divisor + "로 나누면 몫은 " + dividend/divisor + "입니다.");
                break; // 정상적인 나눗기 완료 후 while 벗어나기
            }
            catch(ArithmeticException e) { // ArithmeticException 예외 처리 코드
                System.out.println("0으로 나눌 수 없습니다! 다시 입력하세요");
            }
        }
        scanner.close();
    }
}
```

ArithmeticException
예외 발생

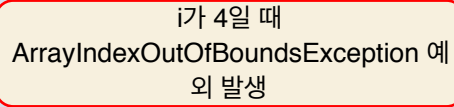
```
나뉘수를 입력하시오:100
나눗수를 입력하시오:0
0으로 나눌 수 없습니다! 다시 입력하세요
나뉘수를 입력하시오:100
나눗수를 입력하시오:5
100를 5로 나누면 몫은 20입니다.
```


예제 3-16 : 범위를 벗어난 배열의 접근

57

배열의 인덱스가 범위를 벗어날 때 발생하는 `ArrayIndexOutOfBoundsException`을 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.
```

예제 3-17 : 입력오류시발생하는 예외(InputMismatchException)

58

3개의 정수를 입력받아 합을 구하는 프로그램을 작성하라. 사용자가 정수가 아닌 문자를 입력할 때 발생하는 `InputMismatchException` 예외를 처리하여 다시 입력받도록 하라.

```
import java.util.Scanner;
import java.util.InputMismatchException;

public class InputException {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("정수 3개를 입력하세요");
        int sum=0, n=0;
        for(int i=0; i<3; i++) {
            System.out.print(i+">>");
            try {
                n = scanner.nextInt(); // 정수 입력
            }
            catch(InputMismatchException e) {
                System.out.println("정수가 아닙니다. 다시 입력하세요!");
                scanner.next(); // 입력 스트림에 있는 정수가 아닌 토큰을 버린다.
                i--; // 인덱스가 증가하지 않도록 미리 감소
                continue; // 다음 루프
            }
            sum += n; // 합하기
        }
        System.out.println("합은 " + sum);
        scanner.close();
    }
}
```

사용자가 문자를 입력하면
`InputMismatchException` 예외 발생

정수 3개를 입력하세요

0>>5

1>>R

정수가 아닙니다. 다시 입력하세요!

1>>4

2>>6

합은 15

예제 3-18 : 정수가 아닌 문자열을 정수로 변환할 때 예외 발생 (NumberFormatException)

59

문자열을 정수로 변환할 때 발생하는 `NumberFormatException`을 처리하는 프로그램을 작성하라.

```
public class NumException {  
    public static void main (String[] args) {  
        String[] stringNumber = {"23", "12", "3.141592", "998"};  
  
        int i=0;  
        try {  
            for (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(stringNumber[i] + "는 정수로 변환할 수 없습니다.");  
        }  
    }  
}
```

“3.141592”를 정수로 변환할 때
`NumberFormatException`
예외 발생

숫자로 변환된 값은 23
숫자로 변환된 값은 12
3.141592는 정수로 변환할 수 없습니다.

Questions??

