

Come to HUFS Meet the World!

객체지향프로그래밍

24.9.25 실습

- 담당 교수 : 전 병 환 교수님
- 조교 : 윤 종 업
- 조교 메일 : juyoon@hufs.ac.kr



9월 13일 실습 정답 코드

- 실습 1: 자바 프로그래밍의 구조

```
import java.io.*;
import java.util.*;

public class Main {
    public static float sum(float n, float m) {
        return n + m;
    }
    public static float sub(float n, float m) {
        return n - m;
    }
    public static float mul(float n, float m) {
        return n * m;
    }
    public static float div(float n, float m) {
        return n / m;
    }
}
```

9월 13일 실습 정답 코드

- 실습 2: 조건 연산자 ?:

```
import java.io.*;
import java.util.*;

public class Main {
    public static int compare(int num1, int num2, int num3) {
        int max = 0;
        max = ((num1 > num2) && (num1 > num3)) ? num1 : ((num2 > num1) && (num2 > num3)) ? num2 : num3;
        return max;
    }
}
```

9월 13일 실습 정답 코드

- 실습 3: Switch 문

```
import java.io.*;
import java.util.*;

public class Main {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Scanner scanner = new Scanner(System.in);

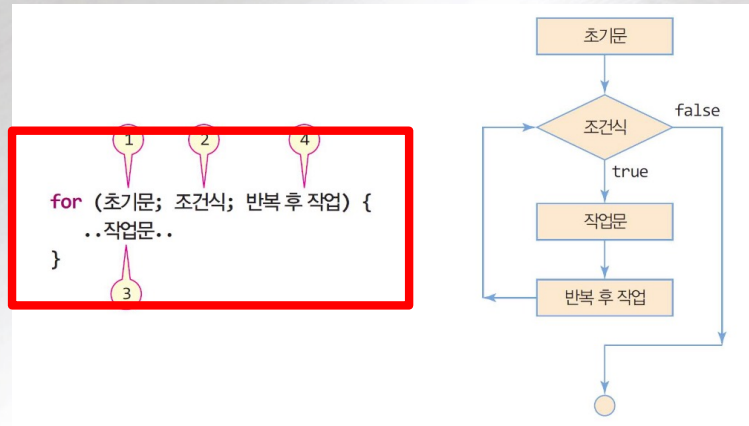
        int month = scanner.nextInt();

        String season = "겨울";
        String a = "현재 계절은 %s입니다.";
    }
}
```

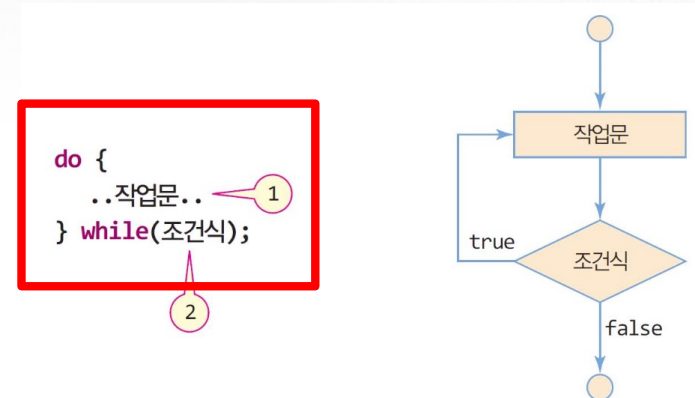
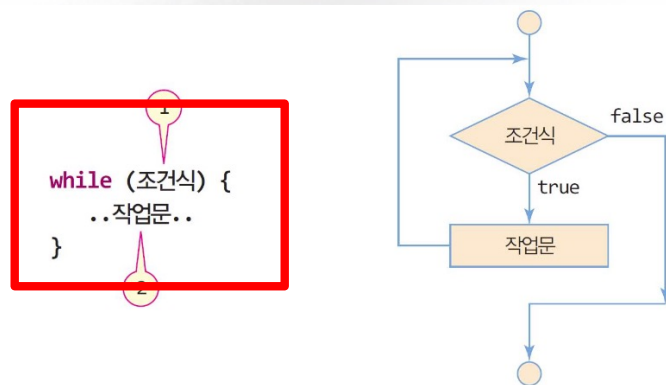
```
if(month>0 && month<13){
    switch(month/3){
        case 1:
            season = "봄";
            break;
        case 2:
            season = "여름";
            break;
        case 3:
            season = "가을";
            break;
    }
}
else{
    a = "입력값이 잘못되었습니다.";
}
System.out.printf(a, season);
}
```


반복문과 배열

- for문



- while 문 / do-while 문



반복문과 배열

• 중첩 반복문

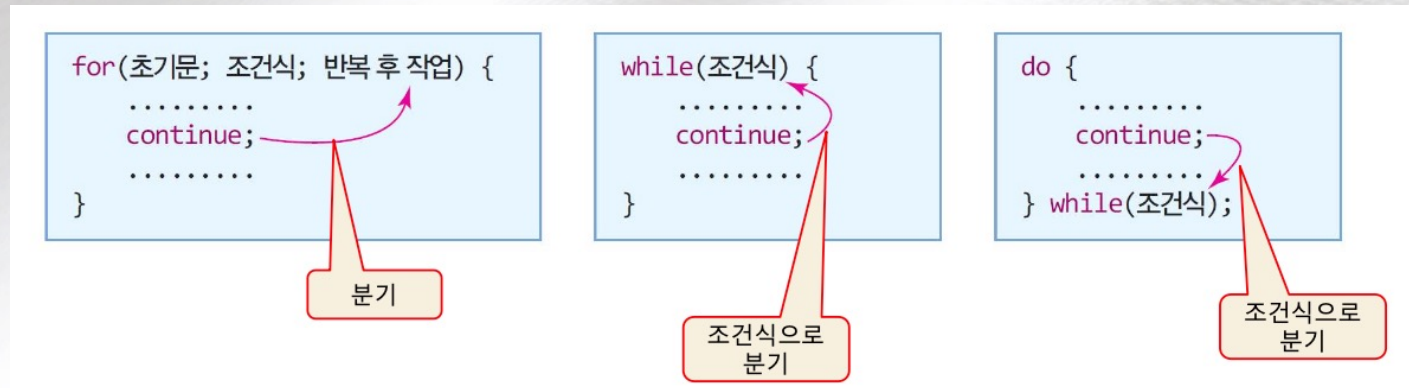
- 반복문이 다른 반복문을 내포하는 구조
- 이론적으로는 몇 번이고 중첩 반복 가능
- 너무 많은 중첩 반복은 프로그램 구조를 복잡하게 하므로 2중 또는 3중 반복이 적당

```
for(int i=0; i<100; i++) { // 100개의 학교 성적을 모두 더한다.  
    ....  
    for(int j=0; j<10000; j++) { // 10000명의 학생 성적을 모두 더한다.  
        ....  
        ....  
    }  
    ....  
}
```

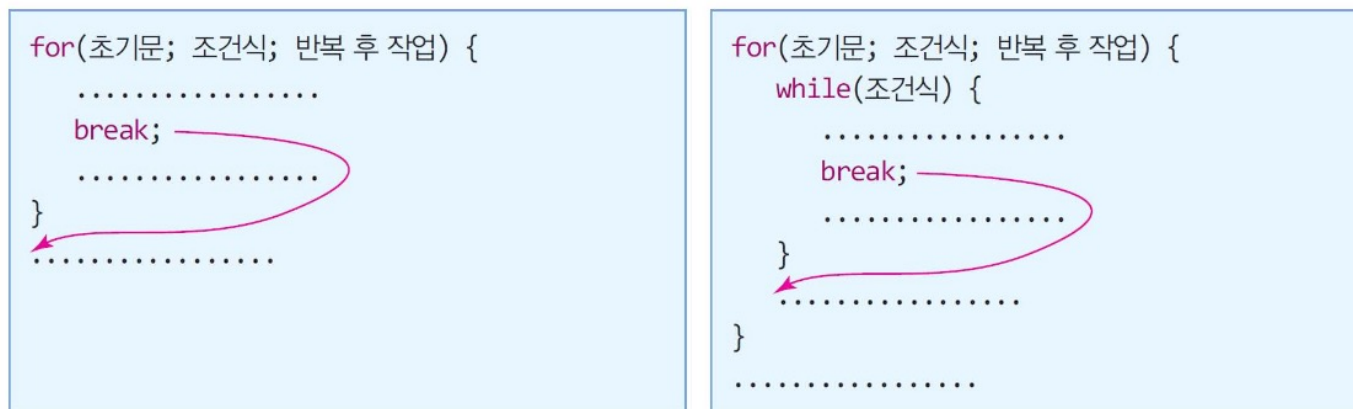
10000명의 학생이 있는 100개 대학의 모든 학생 성적의 합을 구할 때,
for 문을 이용한 이중 중첩 구조

반복문과 배열

• continue 문



• break 문



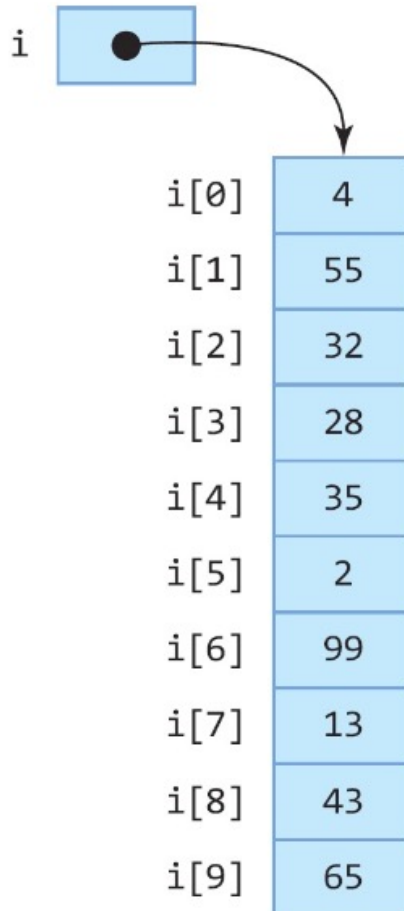
(a) 현재 반복문 벗어나기

(b) 중첩 반복에서 안쪽 반복문만 벗어나는 경우

반복문과 배열

• 배열

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



▣ 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
```


• 배열 인덱스와 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

실습 1: 반복문과 배열

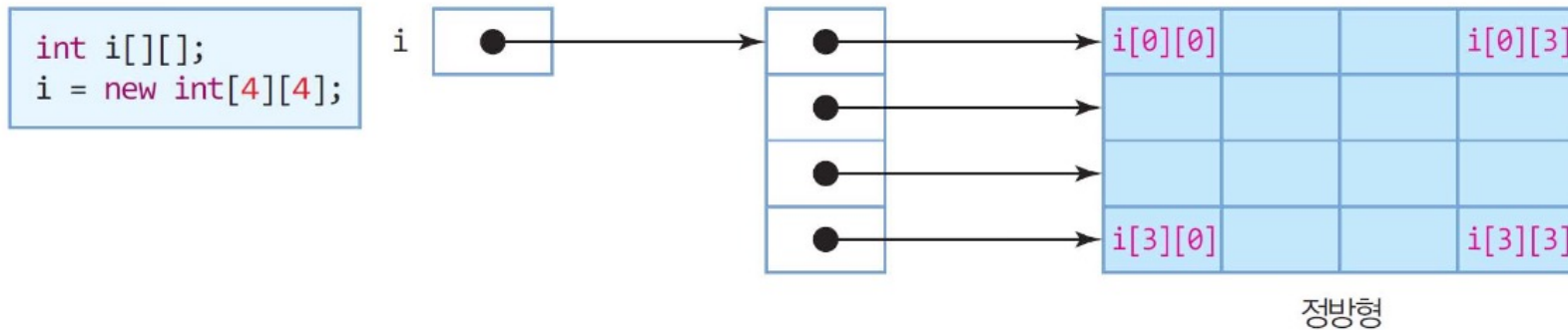
- 정수로 된 돈의 액수를 입력받아 오만 원권, 만 원권, 천 원권, 100원짜리 동전, 10원짜리 동전, 1원짜리 동전이 각 몇 개로 반환되는지 출력하는 코드를 작성하라.
 - 배열과 반복문을 사용한다.
 - 반드시 다음 배열을 이용한다
 - `int [] unit = {50000, 10000, 1000, 100, 10, 1};`

```
Java
1  import java.io.*;
2  import java.util.*;
3
4  public class Main {
5      public static void main(String[] args) {
6          // TODO Auto-generated method stub
7          int [] unit = {50000, 10000, 1000, 100, 10, 1};
8
9      }
10 }
```

비정방형 배열

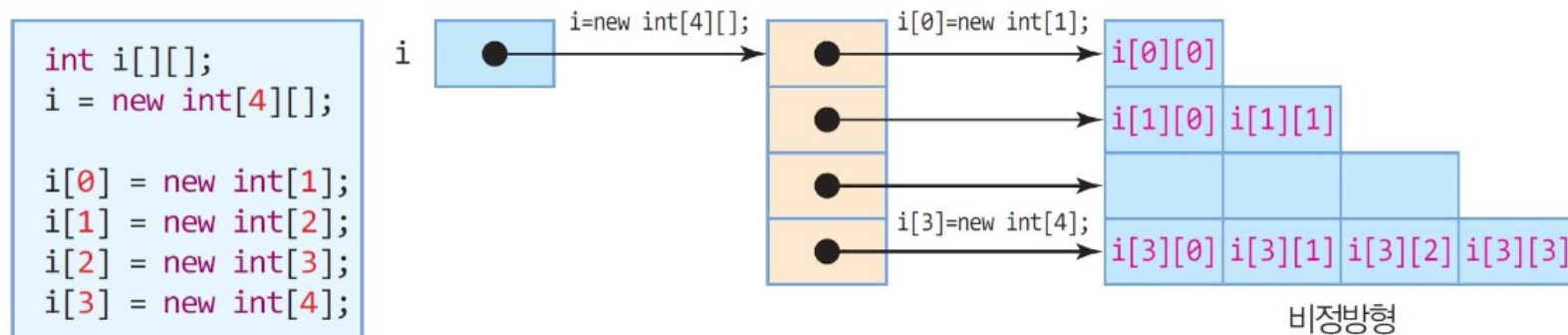
정방형 배열

- 각 행의 열의 개수가 같은 배열



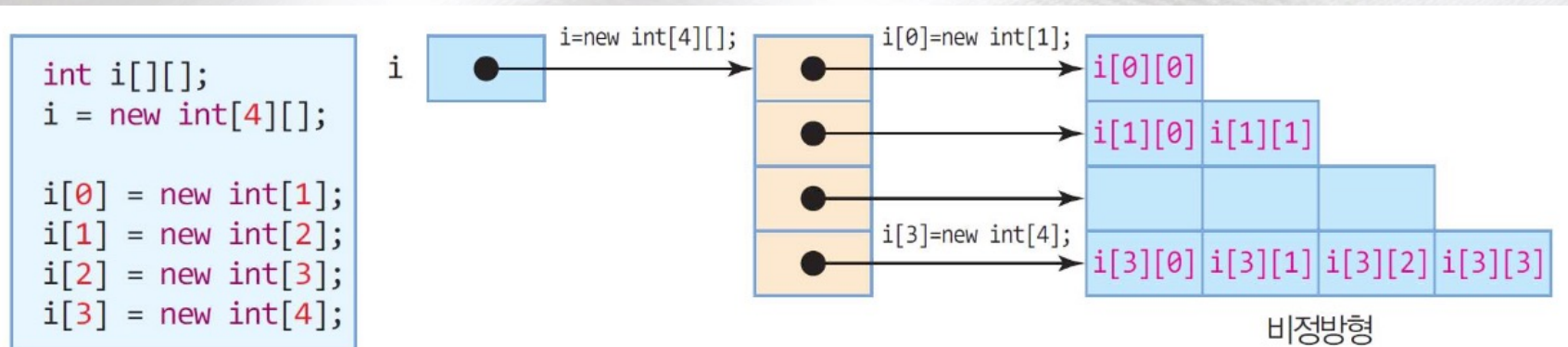
비정방형 배열

- 각 행의 열의 개수가 다른 배열
- 비정방형 배열의 생성



비정방형 배열

- 비정방형 배열의 length



- 비정방형 배열의 length

- `i.length` -> 2차원 배열의 행의 개수로서 4
- `i[n].length`는 `n`번째 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

실습 2: 비정방형 배열

- 다음 그림과 같은 비정방형 배열을 만들어 값을 초기화하고 출력하라.

10	11	12	13	14
20	21	22		
30	31	32	33	
40				
50	51			

```
Java
1  import java.io.*;
2  import java.util.*;
3
4  class Main {
5      public static void main (String[] args) {
6          |
7      }
8  }
```

예외 처리 (try-catch-finally)

```
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)  
}
```

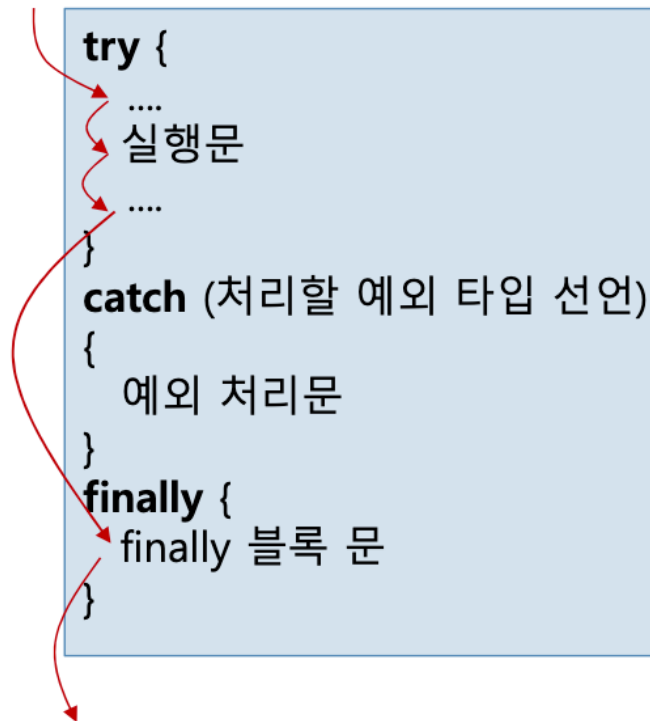
생략
가능

- 예외가 발생할 때 대응하는 응용프로그램 코드
 - try-catch-finally문 사용
 - finally 문은 생략 가능

예외 처리 (try-catch-finally)

- 예외에 따른 제어의 흐름

try블록에서 예외가 발생하지 않은 정상적인 경우



try블록에서 예외가 발생한 경우



예외 처리 (try-catch-finally)

- 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 입력받으려 하였지만, 사용자가 'a' 등과 같이 문자를 입력한 경우	java.util

실습 3: 예외 처리 (try-catch-finally)

- try-catch문을 사용하여 저번 주 실습 1번 코드를 수정하라.
 - 수정 조건
 - try-catch 사용 (finally는 사용해도 되고, 안해도 됨)
 - 0이 입력되면 “0으로 나눌 수 없습니다! 다시 입력하세요.”를 출력
 - 입력 받는 숫자와 메소드 모두 int로 정의
 - 입력 값을 정수로 바꾸면 나누기가 어떻게 달라지는지 생각해보기

```
Java
1  import java.io.*;
2  import java.util.*;
3
4  class Main {
5      public static int sum(int n, int m) {
6          return n + m;
7      }
8      public static int sub(int n, int m) {
9          return n - m;
10     }
11     public static int mul(int n, int m) {
12         return n * m;
13     }
14     public static int div(int n, int m) {
15         return n / m;
16     }
17
18     public static void main(String[] args) {
19     }
20 }
```