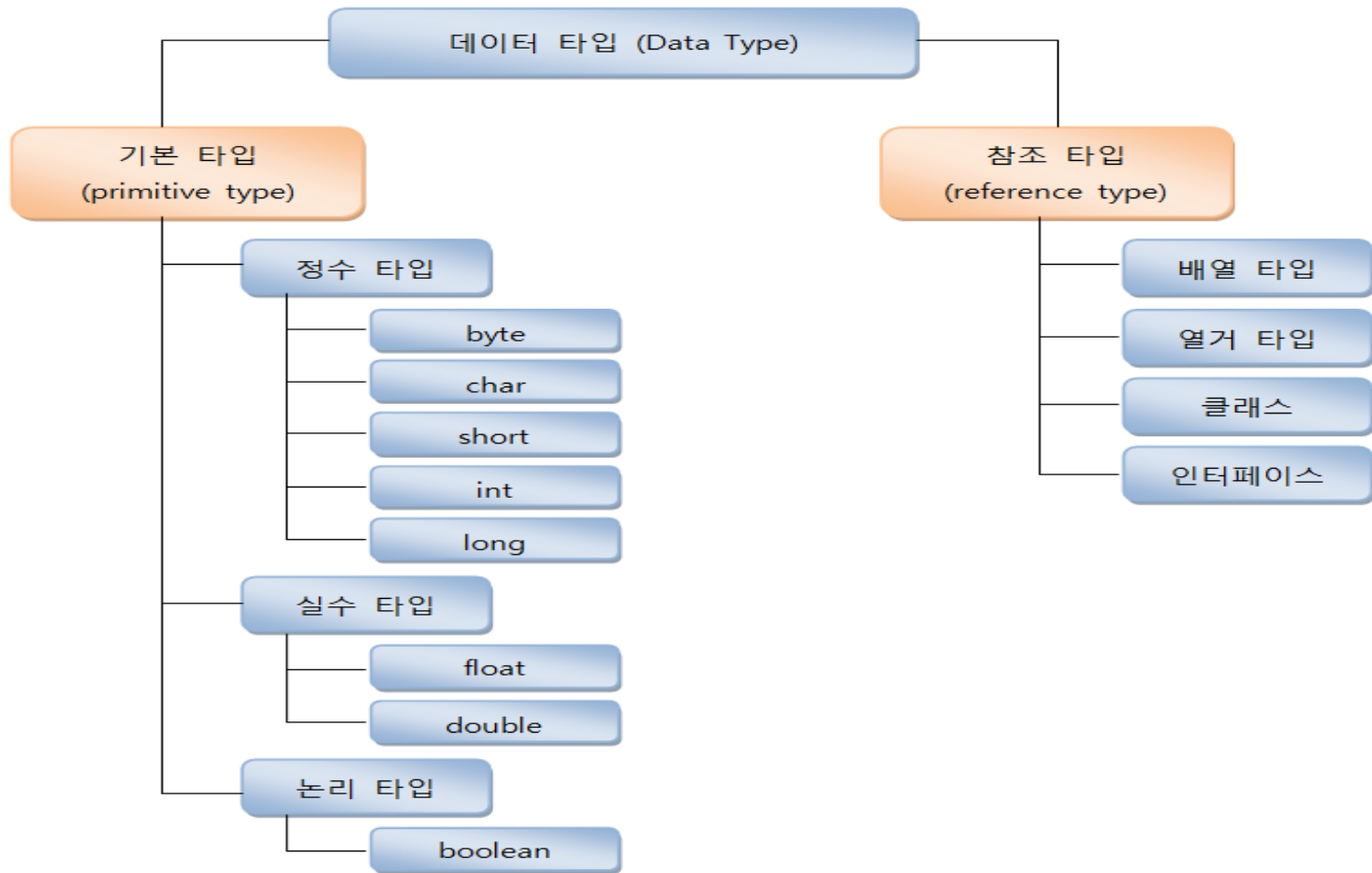


메모리

데이터 타입



메모리 영역

Runtime Data Area

Method Area

클래스-1

런타임 상수풀
필드/메소드 데이터
메소드 코드
생성자 코드

...

클래스-n

런타임 상수풀
필드/메소드 데이터
메소드 코드
생성자 코드

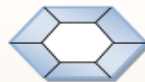
힙(Heap Area)



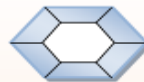
객체-1



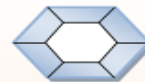
배열-2



객체-3



배열-4



객체-5



객체-6



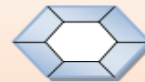
배열-7



객체-8



배열-9



객체-10



객체-11

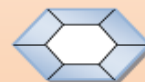


배열-12



객체-13

...



객체-n

스레드-1

JVM 스택(Stack)

프레임-n

변수-n

...

변수-1

...

프레임-1

변수-n

...

변수-1

...

스레드-n

JVM 스택(Stack)

메모리 영역

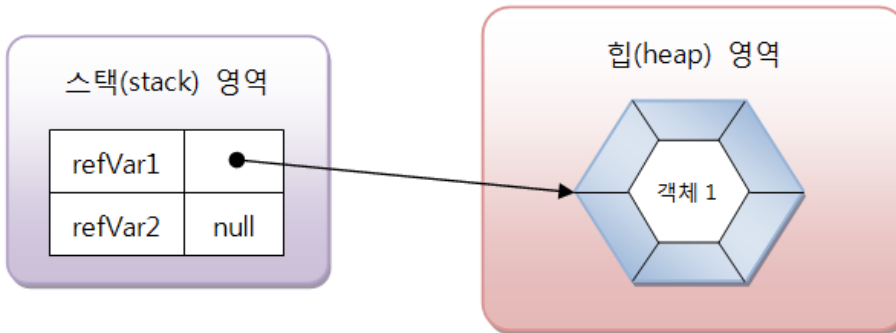
❖ JVM이 사용하는 메모리 영역

- ✓ 메소드 영역 : 클래스 영역이나 static 영역이라고도 함
 - JVM이 시작할 때 생성
 - 로딩된 클래스의 바이트 코드 내용을 분석 후 저장
 - 모든 스레드가 공유
 - 한번 할당 받으면 내용을 수정할 수 없음
- ✓ 힙 영역
 - JVM 시작할 때 생성
 - 객체/배열 저장
 - 사용되지 않는 객체는 Garbage Collector 가 자동 제거
- ✓ JVM 스택
 - 스레드(메소드) 별 생성
 - 메소드 호출할 때마다 스택에 추가(push)
 - 메소드 종료하면 제거(pop)

null

❖ null(널)

- ✓ 참조형 변수가 참조하는 객체가 없을 경우 초기값으로 사용 가능
- ✓ 참조 타입의 변수에만 저장가능
- ✓ null로 초기화된 참조 변수는 스택 영역에 생성



- ✓ ==, != 연산 가능

refVar1 == null	결과: false
refVar1 != null	결과: true

refVar2 는 null 값을 가지므로 연산의 결과는 다음과 같다.

refVar2 == null	결과: true
refVar2 != null	결과: false

배열

1. 배열

- ❖ **동일한 자료형**으로 구성된 **데이터(객체)**의 연속된 **집합(Collection)**
- ❖ 물리적으로 연속된 기억 공간을 사용하며 heap 메모리에 생성
- ❖ 배열도 객체(인스턴스)
- ❖ 하나의 학급에 있는 학생 3명의 평균(실수) 점수를 저장해야 하는 경우
 - ✓ 배열이 없을 때는 `double score1, score2, score3` 이렇게 3개의 변수가 필요
 - ✓ 여러 개의 데이터를 묶을 수 있는 자료형이 있는 경우 `double score[] = new double[3]` 이렇게 선언해서 `score`라는 하나의 이름으로 관리
 - ✓ 배열을 선언하면 하나의 이름으로 관리하고 각 공간에 인덱스를 부여하기 때문에 여러 개의 데이터를 반복문을 이용해서 접근할 수 있어서 많은 양의 데이터를 적은 양의 코드로 처리할 수 있게 됩니다.
 - ✓ 동일한 자료형이라도 데이터 서로 간에 비교가 가능한 경우에 배열로 생성

1. 배열

❖ 구성요소

- ✓ 배열 요소 – 배열 요소의 자료형(컬렉션의 자료형이 아니고 컬렉션을 구성하는 데이터들의 자료형) < 예, int, double 등 >
- ✓ 배열의 크기 - 배열 요소의 개수 < 예, [], [7] 등 >
 - 크기의 개수에 따라 1차원 배열과 2차원 배열 이상의 다차원 배열로 구분
- ✓ 배열명 – 메모리 공간을 사용하므로 재사용을 위해서는 이름이 있어야 재사용 가능

❖ 배열의 선언

- ✓ 자료형 [] 배열명;
- ✓ 자료형 배열명 [];

❖ 배열에 메모리 할당(데이터를 저장할 실제 공간 생성) – 데이터를 저장할 수 있는 공간을 할당

- ✓ 배열명 = new 자료형[개수] => 모두 기본값으로 초기화
 - Heap 메모리는 할당되는 순간 초기값이 없으면 기본값으로 초기화
- ✓ 자료형 배열명[] = {데이터 나열} => 나열된 데이터를 가지고 생성하는 것으로 배열을 처음 생성할 때만 가능한 방식
- ✓ 자료형 배열명[] = new 자료형[] {데이터 나열} => 나열된 데이터를 가지고 생성
- ✓ 배열명 = null => 참조할 데이터가 현재 없다면 null을 대입하는 것도 가능

1. 배열

❖ 변수의 타입에 따른 기본값

- ✓ boolean: false
- ✓ char: 'Wu0000'
- ✓ byte, short, int: 0
- ✓ long: 0L
- ✓ float: 0.0f
- ✓ double: 0.0d 또는 0.0
- ✓ 참조형: null(어떠한 객체도 가리키지 않음)

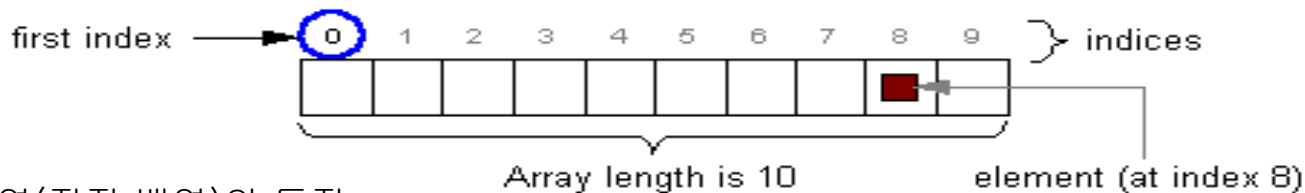
❖ 인덱스(첨자) – 배열요소의 위치를 나타내는 숫자로 0 부터 시작

❖ 배열 선언	메모리 할당	배열 요소의 이용
<code>int[] data;</code>	<code>data = new int[100];</code>	<code>data[0] = 10;</code>

❖ 배열 요소를 사용할 때 잘못된 인덱스를 사용하면 `ArrayIndexOutOfBoundsException`이 발생

1. 배열

❖ 배열의 데이터 개수는 length라는 속성으로 제공 - 배열명.length: 배열에 메모리 할당이 이루어져 있지 않으면 NullPointerException 발생



❖ 배열(정적 배열)의 특징

- ✓ 접근 방법이 쉬움
- ✓ 생성 시 크기를 결정하면 이후에 변경 할 수 없음
- ✓ 연속된 메모리 공간을 사용하므로 연속된 빈 공간이 없으면 생성할 수 없음
- ✓ 데이터를 정렬해두지 않으면 순차 검색을 이용해서 데이터를 검색해야 하기 때문에 검색 속도가 느릴 수 있음
- ✓ 데이터를 중간에 삽입하거나 삭제하려고 하면 새로운 배열을 만들어서 복사한 후 작업해야 하기 때문에 느림

실습(OneArray)

```
public class OneArray {  
    public static void main(String[] args) {  
        int [] ar = {10,20,30};  
        int cnt = ar.length;  
        for(int i=0; i<cnt; i++){  
            //for(int i=0; i<ar.length; i++){  
                System.out.println("ar[" + i + "]=" + ar[i]);  
            }  
        }  
    }  
}
```

ar[0]=10
ar[1]=20
ar[2]=30

1. 배열


- ❖ 배열의 타입으로 참조형도 가능
- ❖ 생성방법은 일반 자료형과 동일
- ❖ 참조형 변수의 기본값은 null이므로 배열을 생성하기만 하면 모두 null을 저장하고 있으므로 메모리 할당을 해서 사용
- ❖ 참조형 배열의 생성 및 초기화

```
String [] names = new String[3];  
names[0] = “데니스 리치히”;  
names[1]=“제임스 고슬링”;  
names[2]=“아네르스 하일스베르”;
```

실습(StringArray)

```
public class StringArray {  
  
    public static void main(String[] args) {  
        //String 객체의 주소 3개를 저장할 수 있는 배열을 생성  
        String [] names = new String[5];  
        //데이터 대입  
        names[0] = "데니스 리치히";  
        names[1] = "제임스 고슬링";  
        names[2] = "귀도 반 로섬";  
        names[3] = "아네르스 하일스베르";  
        names[4] = "애플";  
        //모든 데이터 출력  
        int size = names.length;  
        for(int i=0; i<size; i++){  
            System.out.println(names[i]);  
        }  
    }  
}
```

데니스 리치히
제임스 고슬링
귀도 반 로섬
아네르스 하일스베르



2. 다차원 배열

❖ 2차원 이상의 배열 – 배열의 크기가 2개 이상인 배열

- ✓ 배열의 크기: 행과 열로 구분해서 표현
- ✓ 배열 선언:
 - 자료형 [][] 배열명;
 - 자료형 배열명 [][];
- ✓ 배열의 생성: 배열의 메모리 할당
 - 행과 열의 크기를 동시에 설정
 - 배열명 = new 자료형[행의크기][열의크기];
 - 예> int [][] ar = new int[2][3];
 - 2행 3열의 배열 <2x3 행렬>
 - 행과 열의 분리선언
 - 배열명 = new 자료형[행의크기][];
 - 배열명[인덱스] = new 자료형[열의크기]
 - ar = new int[2][];
 - ar[0] = new int[3]; ar[1] = new int[3];
- ✓ 요소의 접근은 배열명[행번호][열번호]의 형식
- ✓ []를 생략하고 length를 호출하면 생략된 부분의 데이터 개수가 리턴
- ✓ ar.length 의 값은 2이고 ar[0].length의 값은 3

2. 다차원 배열

❖ 2차원 배열을 초기화 한 상태로 생성

- ✓ 형태: 자료형[][] 배열명 = {초기화 리스트};
- ✓ 예> `int[][] a = {{1,2},{3,4},{5,6}};` <3x2 행열>

실습(Matrix)

```
public class Matrix {  
    public static void main(String[] args) {  
        int[][] ar = { { 10, 20, 30 }, { 40, 50, 60 } };  
        int i, j;  
        int rowCount;  
        int columnCount;  
  
        rowCount = ar.length;  
        for (i = 0; i < rowCount; i++) {  
            columnCount = ar[i].length;  
            for (j = 0; j < columnCount; j++)  
                System.out.print(" " + ar[i][j]);  
            System.out.println();  
        }  
    }  
}
```

10	20	30
40	50	60

실습(OneDimentionalArray)

```
public class OneDimentionalArray {  
    public static void main(String[] args) {  
        int[] ar = {10, 20, 30, 40, 50, 60};  
        int i, j;  
        int rowCount = 2;  
        int columnCount = 3;
```

```
10  20  30  
40  50  60
```

```
        for (i = 0; i < rowCount; i++) {  
            for (j = 0; j < columnCount; j++)  
                System.out.print(" " + ar[i*columnCount + j]);  
            System.out.println();  
        }
```

```
    }
```

```
}
```

2. 다차원 배열

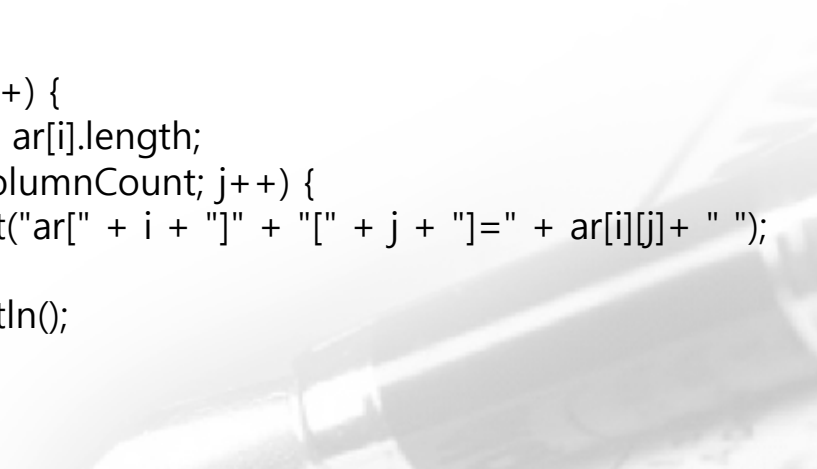
❖ 자바는 배열을 참조형(동적 생성)으로 생성하기 때문에 각 행의 크기가 다른 배열 선언이 가능

```
예> int [][] ar = new int[3][];  
    ar[0] = new int[1]  
    ar[1] = new int[2]  
    ar[2] = new int[3]
```

실습-(VariableArray)

```
public class VariableArray {  
    public static void main(String[] args) {  
        int[][] ar = new int[3][];  
        int i, j = 0;  
        int rowCount, columnCount;  
        ar[0] = new int[1];  
        ar[1] = new int[2];  
        ar[2] = new int[1];  
        ar[0][0] = 10;  
        ar[1][0] = 20;  
        ar[1][1] = 30;  
        ar[2][0] = 40;  
        rowCount = ar.length;  
        for (i = 0; i < rowCount; i++) {  
            columnCount = ar[i].length;  
            for (j = 0; j < columnCount; j++) {  
                System.out.print("ar[" + i + "]" + "[" + j + "]=" + ar[i][j] + " ");  
            }  
            System.out.println();  
        }  
    }  
}
```

ar[0][0]=10
ar[1][0]=20 ar[1][1]=30
ar[2][0]=40



3. 정렬(Sort)

❖정렬은 데이터를 순서대로 나열하는 것

❖정렬방식은 작은 것부터 큰 순서대로 나열하는 오름차순(Ascending) 정렬과 큰 것에서 작은 순서대로 나열하는 내림차순(Descending) 정렬이 있고 정렬을 하는 방법은 여러 가지가 있음

❖선택정렬은 첫 번째 자리부터 마지막에서 두 번째 자리까지 자신보다 뒤에 있는 모든 자리들과 비교해서 다음 자료가 작으면 2개의 요소의 자리를 변경해주면 됩니다.

❖ex)

초기 상태 50 40 10 20 30

1Pass 10 50 40 20 30

2Pass 10 20 50 40 30

3Pass 10 20 30 50 40

4Pass 10 20 30 40 50

❖1번째 자리를 기준으로 2,3,4,5 번째 자리와 비교

❖1번째 자리는 제외하고2번째 자리를 기준으로 3,4,5 번째 자리와 비교

❖3번째 자리를 기준으로 4,5 번째 자리와 비교

❖4번째 자리를 기준으로 5번째 자리와 비교

실습-선택 정렬(SelectionSort)

```
public class SelectionSort {  
    public static void main(String[] args) {  
        int test[] = { 20, 30, 40, 50, 10 };  
        int i, j, temp;  
        int cnt = test.length;  
        System.out.println("정렬 전");  
        for (i = 0; i < cnt; i++) {  
            System.out.println((i + 1) + "번째 데이터" + test[i]);  
        }  
        for (i = 0; i < cnt-1; i++) {  
            for (j = i + 1; j < cnt; j++) {  
                if (test[i] < test[j]) {  
                    temp = test[i];  
                    test[i] = test[j];  
                    test[j] = temp;  
                }  
            }  
        }  
    }  
}
```

정렬 전

1번째 데이터 20

2번째 데이터 30

3번째 데이터 40

4번째 데이터 50

5번째 데이터 10

=====

=====

정렬 후

1번째 데이터 50

2번째 데이터 40

3번째 데이터 30

4번째 데이터 20

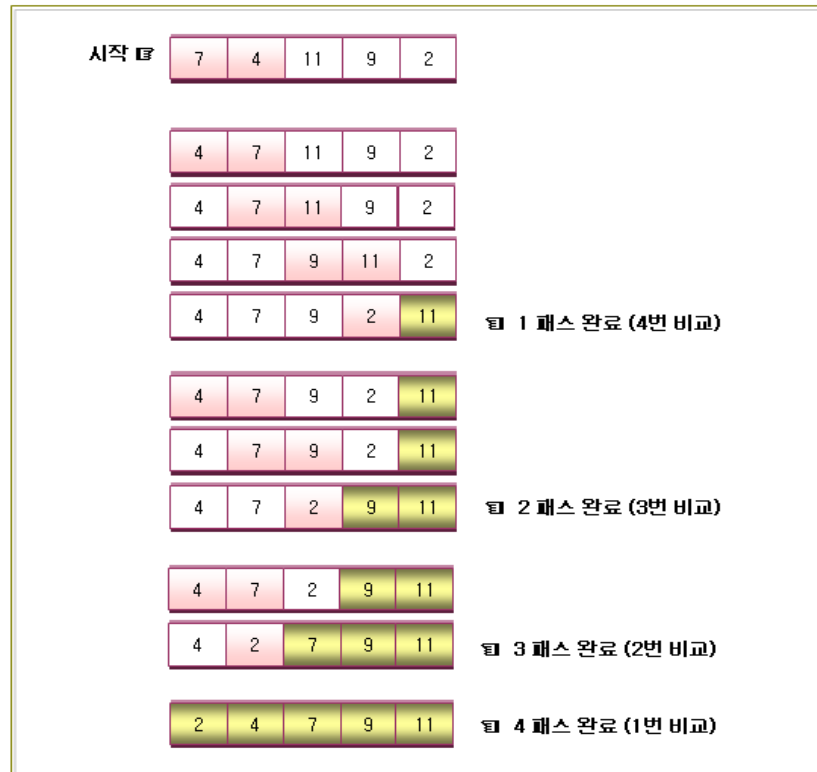
5번째 데이터 10

실습-선택 정렬(SelectionSort)

```
System.out.println("=====");
System.out.println("=====");
System.out.println("정렬 후");
for (i = 0; i < cnt; i++) {
    System.out.println((i + 1) + "번째 데이터" + test[i]);
}
}
}
```

4. 정렬

- ❖ 버블 정렬은 n 개의 데이터가 있을 때 1부터 $n-1$ 번째 자료까지 $n-1$ 번 동안 다음 자료와 비교해가면서 정렬하는 방법
- ❖ 버블 정렬의 효과를 높이기 위해서는 비교 시 횟수만큼 빼면서 정렬하면 성능을 높일 수 있고 flag처리 등을 이용해서 자리 바꿈이 일어나지 않을 때 멈추게 하면 성능을 더욱 향상




실습-버블 정렬(BubbleSort)

```
public class BubbleSort {  
    public static void main(String[] args) {  
        int test[] = { 20, 30, 40, 50, 10 };  
        int i, j, temp, flag;  
        int cnt = test.length;  
        System.out.println("정렬 전");  
        for (i = 0; i < cnt; i++) {  
            System.out.println((i + 1) + "번째 데이터" + test[i]);  
        }  
        for (i = 0; i < cnt-1; i++) {  
            flag = 0;  
            for (j = 0; j < cnt - (i+1); j++) {  
                if (test[j] < test[j + 1]) {  
                    temp = test[j];  
                    test[j] = test[j + 1];  
                    test[j + 1] = temp;  
                    flag = 1;  
                }  
            }  
        }  
    }  
}
```

```
<terminated> bubblesort.java  
정렬 전  
1번째 데이터 20  
2번째 데이터 30  
3번째 데이터 40  
4번째 데이터 50  
5번째 데이터 10  
-----  
정렬 후  
1번째 데이터 50  
2번째 데이터 40  
3번째 데이터 30  
4번째 데이터 20  
5번째 데이터 10
```


실습-버블 정렬(BubbleSort)

```
        if (flag == 0) {  
            break;  
        }  
    }  
    System.out.println("=====");  
    System.out.println("=====");  
    System.out.println("정렬 후");  
    for (i = 0; i < cnt; i++) {  
        System.out.println((i + 1) + "번째 데이터" + test[i]);  
    }  
}  
}
```



5. 검색

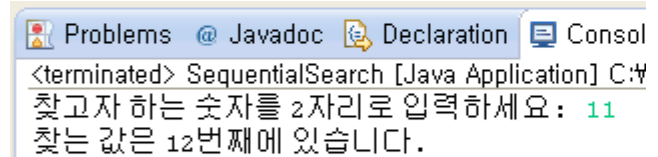
❖ 순차 검색

- ✓ 데이터가 정렬되어 있지 않은 경우 첫 번째 데이터부터 마지막 데이터까지 모두 검색해서 찾는 방법
- ✓ 정렬되지 않은 서랍에서 물건을 찾는 경우와 동일
- ✓ 평균 비교 횟수가 데이터가 있을 확률을 0.5라고 한다면
 $0.5 * n + 0.5 * n / 2$
- ✓ 여러 번 검색해야 하는 경우 데이터가 많거나 찾고자 하는 데이터가 배열에 없는 경우가 많을 경우 비 효율적인 방법

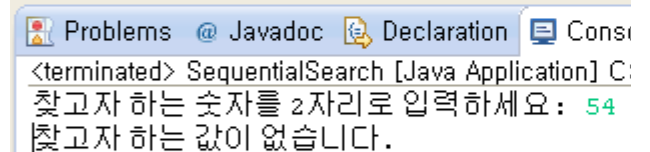
실습-순차검색(SequentialSearch)

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

```
public class SequentialSearch {  
    public static void main(String[] args) {  
        int ar[] = { 23, 45, 19, 63, 57, 26, 75, 73, 82, 89, 47, 11 },  
        int i, num;  
        int key = 0, index = 0;  
        num = ar.length;  
  
        Scanner scanner = new Scanner(System.in);  
        System.out.print("찾고자 하는 숫자를 2자리로 입력하세요: ");  
        key = scanner.nextInt();
```



Problems @ Javadoc Declaration Console
<terminated> SequentialSearch [Java Application] C:\
찾고자 하는 숫자를 2자리로 입력하세요: 11
찾는 값은 12번째에 있습니다.



Problems @ Javadoc Declaration Console
<terminated> SequentialSearch [Java Application] C:\
찾고자 하는 숫자를 2자리로 입력하세요: 54
찾고자 하는 값이 없습니다.

실습-순차검색(SequentialSearch)

```
for (i = 0; i < num; i++) {  
    if (ar[i] == key) {  
        index = i + 1;  
    }  
}  
  
if (index == 0) {  
    System.out.println("찾고자 하는 값이 없습니다.");  
}  
  
else {  
    System.out.println("찾는 값은 " + index + "번째에 있습니다.");  
}  
scanner.close();  
}  
}
```

5. 검색

❖이분 검색

❖데이터가 정렬되어 있는 경우 중앙 값과 비교해서 작으면 왼쪽으로 크면 오른쪽으로 이동해서 검색하는 방법

1) $low = 0$ (데이터의 시작위치), $high = n$ (데이터의 개수)-1로 초기화

2)무한 반복 문에서

$low > high$ 이면 데이터가 없는 것이므로 break;

$low > high$ 가 아니면 $middle = (low + high)/2$ 를 해서 middle값을 찾는다.

검색 값과 $data[middle]$ 과 비교해서 같으면 찾은 것이므로 출력하고 break;

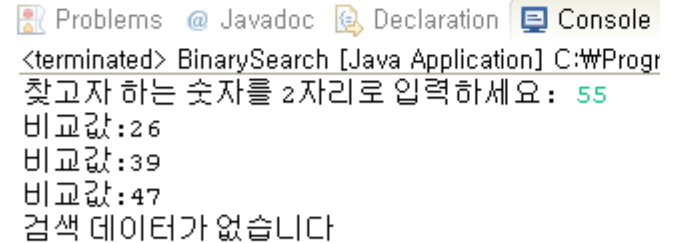
검색 값이 중앙값보다 크다면 $low = middle + 1$

작다면 $high = middle - 1$ 을 해서 반복

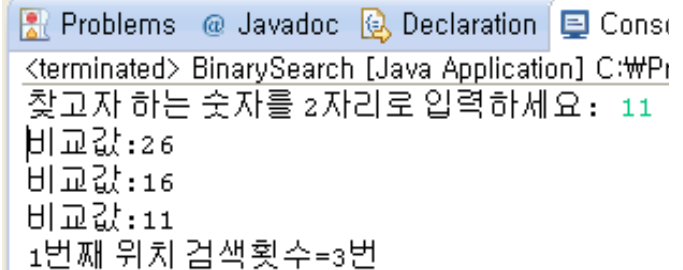
실습-이분검색(BinarySearch)

```
import java.util.*;

public class BinarySearch {
    public static void main(String[] args) {
        int data[] = { 11, 16, 21, 26, 35, 39, 47};
        int k = 0, cnt = 0;
        int low = 0;
        int high = data.length-1;
        int middle;
        Scanner scanner = new Scanner(System.in);
        System.out.print("찾고자 하는 숫자를 2자리로 입력하세요: ");
        k = scanner.nextInt();
```



Problems Javadoc Declaration Console
<terminated> BinarySearch [Java Application] C:\WPPr
찾고자 하는 숫자를 2자리로 입력하세요: 55
비교값:26
비교값:39
비교값:47
검색 데이터가 없습니다



Problems Javadoc Declaration Console
<terminated> BinarySearch [Java Application] C:\WPPr
찾고자 하는 숫자를 2자리로 입력하세요: 11
비교값:26
비교값:16
비교값:11
1번째 위치 검색횟수=3번

실습-이분검색(BinarySearch)

```
while (true) {  
    if (low > high) {  
        System.out.println("검색 데이터가 없습니다");  
        break;  
    }  
    middle = (low + high) / 2;  
    cnt++;  
    System.out.println("비교값:" + data[middle]);  
    if (data[middle] == k) {  
        System.out.println(middle + 1 + "번째 위치 검색횟수=" + cnt + "  
    번");  
        break;  
    }  
    if (k > data[middle]) {  
        low = middle + 1;  
    } else {  
        high = middle - 1;  
    }  
}  
scanner.close();  
}
```

연습문제

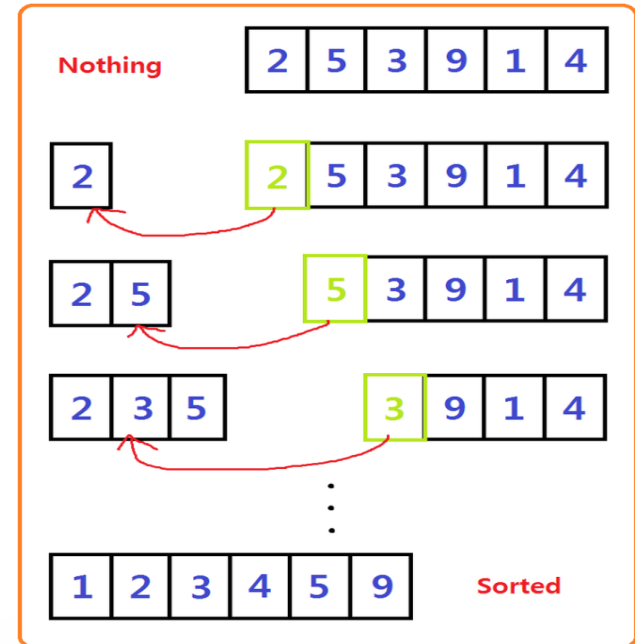
- ❖ 다섯 개의 정수를 입력 받아서 배열에 저장하고 배열에서 가장 큰 수와 그 수가 저장된 위치를 출력하는 프로그램을 작성
- ❖ 다섯 개의 정수를 입력 받아서 배열에 저장하고 다른 배열을 생성해서 순위를 구하는 프로그램을 작성(단 동일한 점수이면 순위는 동일한 것으로 처리)

연습문제

❖ 옆의 그림에서와 같은 숫자를 저장하는 배열을 생성해서 데이터를 저장 한 후 삽입 정렬을 수행해서 다른 배열에 저장 한 후 출력하는 프로그램을 작성

❖ 삽입 정렬

- ✓ 첫번째 데이터만 삽입된 상태로 시작
- ✓ 두번째 데이터부터 삽입하면서 자신의 앞에 있는 데이터와 비교해서 자신의 데이터보다 크다면 뒤로 한칸 이동
- ✓ 첫번째 데이터와 비교하거나 자신보다 작은 데이터를 만나면 비교한 다음 위치에 데이터를 삽입하고 정렬 종료



연습문제

❖ 정수 6개를 저장할 수 있는 배열을 생성해서 1-45까지의 숫자를 중복되지 않게 입력하는 프로그램을 작성

✓ 입력

숫자를 입력하세요:34

숫자를 입력하세요:98

1~45사이의 숫자만 입력하세요

숫자를 입력하세요:32

숫자를 입력하세요:34

중복된 숫자입니다.

숫자를 입력하세요:28

숫자를 입력하세요:29

숫자를 입력하세요:42

숫자를 입력하세요:6

✓ 출력

0:34

1:32

2:28

3:29

4:42

5:6

연습문제

❖년, 월, 일을 입력받아서 입력받은 날짜가 무슨 요일인지 출력하는 프로그램

✓ 입력

년도를 입력하세요:2020

월을 입력하세요:4

일을 입력하세요:13

✓ 출력

2020년 4월 13일은 월요일입니다.

연습문제

❖년과 월을 입력받아서 달력을 출력하는 프로그램을 작성

✓ 입력

년도를 입력하세요:2020

월을 입력하세요:4

✓ 출력

월 화 수 목 금 토 일

1 2 3 4 5

6 7 8 9 10 11 12

13 14 15 16 17 18 19

20 21 22 23 24 25 26

27 28 29 30