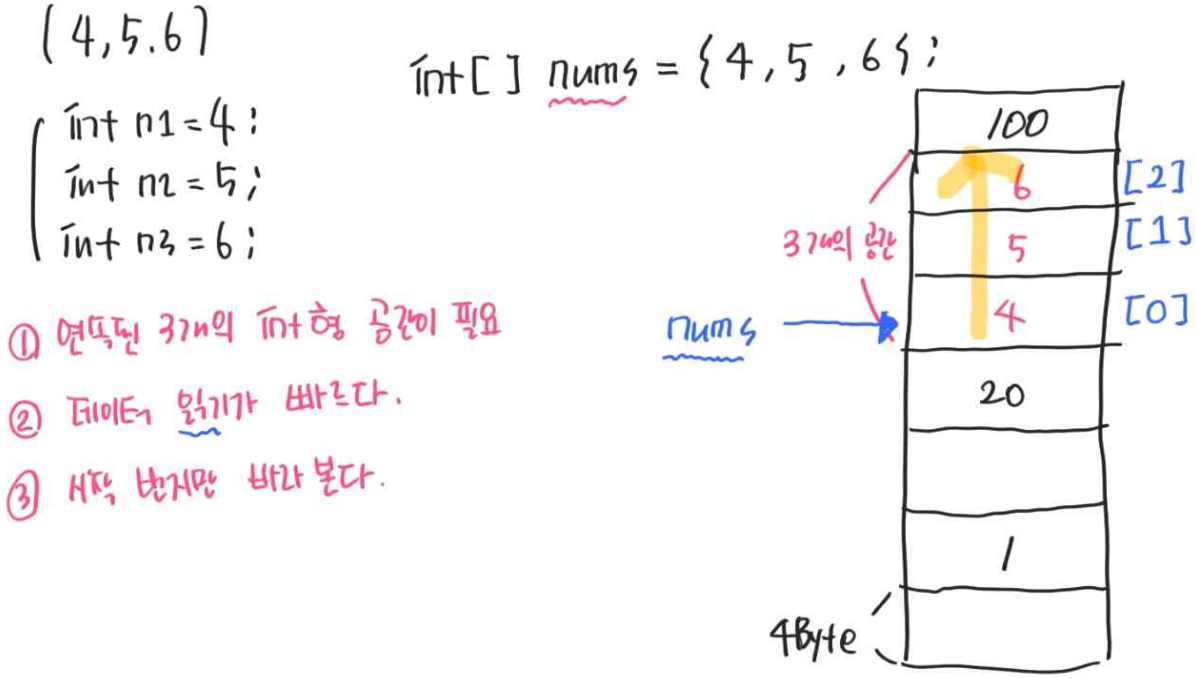
Chapter 6. 배열/조건문/반복문/컬렉션/제네릭

48장. 배열

▷ 배열 : 연관된 데이터를 저장하기 위한 자료구조

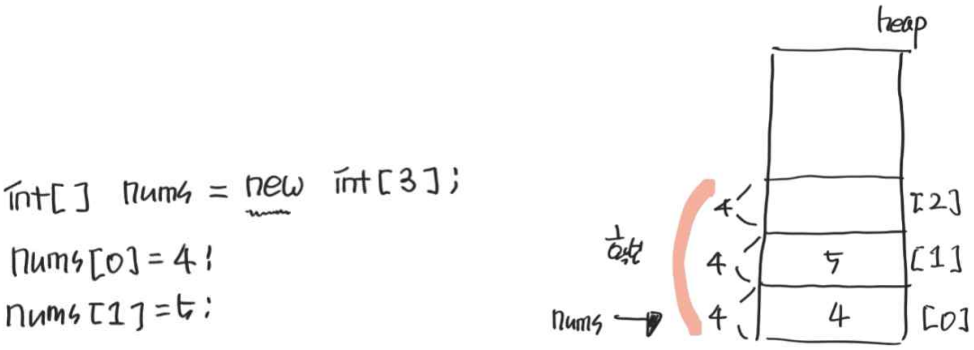
※ 배열은 항상 0 번지 부터 시작한다.




```
package ch06;
public class ArraryEx01{
   public static void main(String[] args){
      int[] nums = {4,5,6};
      System.out.println(nums[0]);
      System.out.println(nums[1]);
      System.out.println(nums[2]);
      System.out.println(nums[3]);    // Out of Bound 오류 발생!! → 범위를 벗어났다는 오류!

      //for 구문을 이용한 출력!
      for(int i=0;i<num.length;i++){
                System.out.println(num[i]);
      }
   }
}
```

49장. 배열 고급

▷ 배열의 동적 할당




```
package ch06;
public class ArraryEx02{
   public static void main(String[] args){
      int[] nums = new int[3];
      nums[0] = 4;
      nums[1] = 5;
      System.out.println(nums[0]);
      System.out.println(nums[1]);
      System.out.println(nums[2]);   // 생성 후 아무것도 넣지 않으면 초기값으로 0 이 들어간다.

      //향상된 for 구문(advanced for)
      for(int i : nums){
          System.out.println(i);
      }
   }
}
```

50장. 2차원 배열

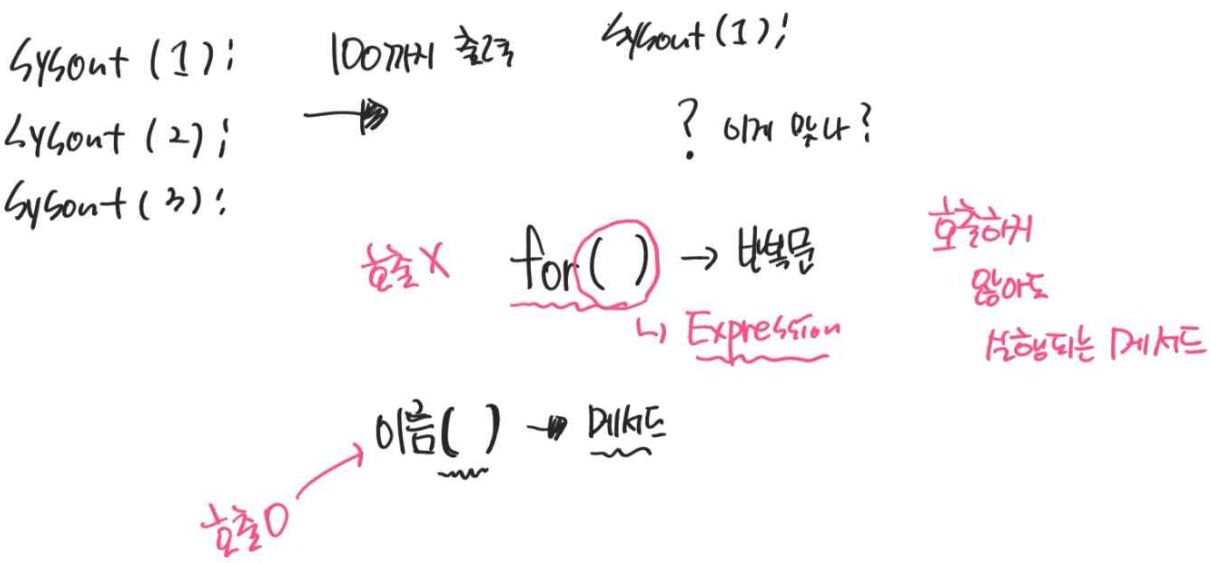
▷ 1차원 배열이 두개 생성

▷ 힙 메모리에는 행 저장, 그리고 열 저장

```
package ch06;
public class ArraryEx03{
   public static void main(String[] args){
      int[][] nums = {{1,2,3}, {4,5,6}};   // 2행 3열의 2차원 배열 생성
      System.out.println(nums[0][0]);
      System.out.println(nums[0][2]);
      System.out.println(nums[1][1]);

      int[][] nums2 = new int[2][2];
      nums2[0][0] = 1;
      nums2[0][1] = 2;
      nums2[1][0] = 3;
      nums2[1][1] = 4;
      System.out.println(nums2[1][0]);
   }
}
```

51장. 반복문(for 문)

▷ for (변수 = 시작값; 변수 <= 끝값; 변수++) {}




```
package ch06;
public class ForEx01{
   public static void main(String[] args){
      // 표현식 : Expression
      // int i = 1 → 초기화(for문이 시작될 때 단 한번만 실행된다.)
      // i = i + 1 → 증감식(증가 or 감소) : for 문이 한번 실행된 후 조건이 만족되면 계속 실행
      // i < 3 → 조건문(처음부터 끝까지 계속 실행)
      for(int i=0; i<3; i=i+1){   // i 는 지역변수
         System.out.println(i);
      }
   }
}
```

52장. 반복문(구구단)

▷ 원시적으로 구구단을 출력해 보자.

```
package ch06;
public class ForEx02{
   public static void main(String[] args){
      // 구구단 중 2단 출력
      System.out.println("2 * 1 = 2");
      System.out.println("2 * 2 = 4");
      System.out.println("2 * 3 = 6");
      System.out.println("2 * 4 = 8");
      System.out.println("2 * 5 = 10");
      System.out.println("2 * 6 = 12");
      System.out.println("2 * 7 = 14");
      System.out.println("2 * 8 = 16");
      System.out.println("2 * 9 = 18");  //  →  얘네들을 자동화하는 내용으로 변환
   }
}
```

53장. 반복문(구구단 연습문제 및 풀이)

▷ for 문을 이용한 구구단

```
package ch06;
public class ForEx03{
   public static void main(String[] args){
      for( i=2, i<=9, i=i+1){
         System.out.println( i + " * 1  = " + i * 1);
         System.out.println( i + " * 2  = " + i * 2);
         System.out.println( i + " * 3  = " + i * 3);
         System.out.println( i + " * 4  = " + i * 4);
         System.out.println( i + " * 5  = " + i * 5);
         System.out.println( i + " * 6  = " + i * 6);
         System.out.println( i + " * 7  = " + i * 7);
         System.out.println( i + " * 8  = " + i * 8);
         System.out.println( i + " * 9  = " + i * 9);
      }
      for( i=2; i<=9; i=i+1){
         System.out.println(i +"단");
         System.out.println("============================");
         for( j=1; j<=9; j=j+1){
```

```
            System.out.println( i + " * " + j + " = " + i*j);
        }
        System.out.println("============================");
        System.out.println();
    }
  }
}
```

54장. 반복문(while 문)

▷ While 구문은 끝이 없는 반복문이다.

▷ While(조건) { }

```
package ch06;
public class WhileEx02{
  public static void main(String[] args){
    int n = 1;
    while(n<10){    // 지금 상태는 무한루프에 걸린다.
      System.out.println(n);
      n = n + 1;
    }
  }
}
```

55장. 조건문(if 문) & switch 구문

▷ if (조건) {
    참인문장;
  } else {
    거짓인문장;
  }

```
package ch06;
public class ConEx01{
  public static void main(String[] args){
    int point = 70;
    // 성적이 60점이 넘으면 용돈을 주겠다.
    if(point >= 60){
      System.out.println("용돈을 주겠다.");
    }
  }
}
```

```
package ch06;
public class ConEx02{
  public static void main(String[] args){
    int point = 80;
    // 성적이 90점이 넘으면 차를 사주겠다.
    // 만약에 90점이 안되면 벌을 주겠다.
    if(point >= 90){
      System.out.println("차를 사 주겠다.");
    } else {
      System.out.println("벌을 주겠다.");
    }
  }
}
```

```
package ch06;
public class ConEx03{
  public static void main(String[] args){
    int point = 80;
    // 성적이 90점이 넘으면 차를 사주겠다.
    // 만약에 80점이 넘으면 자전거를 주겠다.
    // 그게 아니면 벌을 주겠다.
    if(point >= 90){  // 1. 만약 90점이 넘는다면?
      System.out.println("차를 사 주겠다.");
```

```
    } else if(point >= 80) {  // 2. 90점이 넘지 못하고 80점을 넘는다면?
       System.out.println("자전거를 주겠다.");
    } else {  // 3. 위 두 조건을 만족하지 못하면?
       System.out.println("벌을 주겠다.");
    }
  }
}

---  if 구문 내에서의 continue, break, return
        for (int i = 1; i <= 10 ; i++) {
            if (i % 5 == 0) {
                //continue, break, return test
                return;
            } else {
                System.out.println(i);
            }
        }
        System.out.println("메서드 종료");
```

▷ switch 구문

```
switch (변수) {
case 값: // 변수와 값이 일치하면 해당 case 실행문을 작동시킨다.
   실행문;
   break; // break는 조건에 해당하는 실행문을 작동시키고 switch문을 종료하기
          위해 사용된다.
default: // 변수와 값이 불일치하면 default 실행문을 작동시킨다.
   실행문;
   break;
}
```

```
package ch06;
public class SwitchEx {
        public static void main(String[] args) {
                int n = 2;
                switch (n) {        // 조건
                case 1:             // 값 불일치(미실행)
                        System.out.println("1");
                        break;
                case 2:             // 값 일치
```

```
                    System.out.println("2"); // 실행
                    break; // 종료
                case 3:
                    System.out.println("3");
                    break;
                default:
                    System.out.println("4이상");
                }
        }
}
```

56장. 조건문 + 반복문(연습문제)

① 1부터 10까지의 수를 출력하세요. (단, 숫자가 6일 때는 출력하지 마세요)

```
package ch06;
public class ConEx04{
   public static void main(String[] args){
      for(int i=1; i<=10; i++){
         if(i != 6){
            System.out.println(i);
         }
      }
   }
}
```

② 10부터 1까지의 수를 출력하세요. (단, 숫자가 3일 때는 출력하지 마세요)

```
package ch06;
public class ConEx05{
   public static void main(String[] args){
      for(int i=10; i>=1; i--){
         if(i != 3){
            System.out.println(i);
         }
      }
   }
}
```

③ 1부터 20까지의 수를 출력하세요. (단, 2의 배수일 때만 출력하세요)

```
package ch06;
public class ConEx06{
   public static void main(String[] args){
      // 증분을 이용한 방법
      for(int i=2; i<=20; i=i+2){
            System.out.println(i);
        }
      }
      // 나머지를 이용한 방법
      for(int i=1; i<=20; i++){
        if((i%2)==0){
           System.out.println(i);
        }
      }
   }
}
```

57장. 배열, 반복문, 메서드 연습문제

▷ 동전 바꾸기

```
package ch06;
public class ArrayForEx07{
   public static void main(String[] args){
      // 최소 동전 구하기 500, 100, 50, 10
      int[] coin = {500, 100, 50, 10}
      int money = 3680;
      // Step 1
      System.out.println(coin[0] + "원 : " + (restMoney/coin[0]));
      money = money % coin[0]; // 나머지 : 180
      System.out.println("남은금액 : " + money);

      // Step 2
      System.out.println(coin[1] + "원 : " + (money/coin[1]));
      money = money % coin[1];
      System.out.println("남은금액 : " + money);

      // Step 3
      System.out.println(coin[2] + "원 : " + (money/coin[2]));
      money = money % coin[2];
```

```
    System.out.println("남은금액 : " + money);

    // Step 4
    System.out.println(coin[3] + "원 : " + (money/coin[3]));
    money = money % coin[3];
    System.out.println("남은금액 : " + money);
  }
}
```

▷ 위의 코드 중 스탭 4개를 배열과 for 문을 이용해서 수정하시오!! → 리팩토링(재생산)

```
package ch06;
public class ArrayForEx08{
  public static void main(String[] args){
    // 최소 동전 구하기 500, 100, 50, 10
    int[] coin = {500, 100, 50, 10}
    int money = 3680;
    for(int i=0; i<4; i++) {
      System.out.println(coin[i] + "원 : " + (money/coin[i]));
      money = money % coin[i];
      System.out.println("남은금액 : " + money);
    }
  }
}
```

▷ 위의 코드 중 for 문 안의 코드를 Method로 변환하시오.

```
package ch06;
public class ArrayForEx09{
  static void 남은금액계산(int[] coin, int money){
    for(int i=0; i<4; i++) {
      System.out.println(coin[i] + "원 : " + (money/coin[i]));
      int money = money % coin[i];
      System.out.println("남은금액 : " + money);
    } }
  public static void main(String[] args){
    // 최소 동전 구하기 500, 100, 50, 10
    int[] coin = {500, 100, 50, 10}
    int money = 3680;
    남은금액계산(coin, money);
  }
}
```

package name : simpleAlgorithm

> 1번 문제

▶ class name : BetweenNumberSum

▶ 내용> 임의의 두 정수 사이의 합을 구하여 출력합니다.

예를 들어, 3과 5가 주어지면 3+4+5를 계산해서 리턴하면 됩니다. 두 수는 Scanner를 통해 정수로 입력 받습니다.

단, 작은 수와 큰 수가 바뀌어 들어 올 경우 두 수를 바꾸어서 처리합니다.

출력 예>

- 첫 번째 수를 입력하세요 : 5
- 두 번째 수를 입력하세요 : 3
- 두 수 3 과 5 사이의 수의 합은 : 13입니다.
- 계속하시겠습니까?(Y/N)

> 2번 문제

▶ class name : LottoNum

▶ 내용> Random 클래스 함수를 이용하여 중복되지 않는 로또 번호 6개를 생성하여 출력하시오.

단, 로또번호는 1번~45번 사이입니다. 결과는 배열에 넣어 저장하고 출력하시오.

- Random 클래스를 이용한 1~45사이의 랜덤값은

```
Random r = new Random();
lottoNum = r.nextInt(45)+1;을 사용합니다.
```

출력 예> 34, 9, 12, 8, 10, 42

> 3번 문제

▶ class name : DiceProblem

▶ 주사위 두 개를 36,000번 던져서 나오는 모든 경우의 수를 계산하는 프로그램을 작성하세요.
주사위 각각은 1부터 6까지의 정수값을 표시할 수 있으므로 합계는 2부터 12까지입니다.

- 배열을 사용하여 주사위 2개를 던지는 과정을 처리합니다.
- String.format("%.6f", 나온수/36000); 형식을 이용하면 소수이하 6자리를 표현할 수 있습니다.

[아래의 출력 값과 유사한 값이 출력되어야 합니다.]

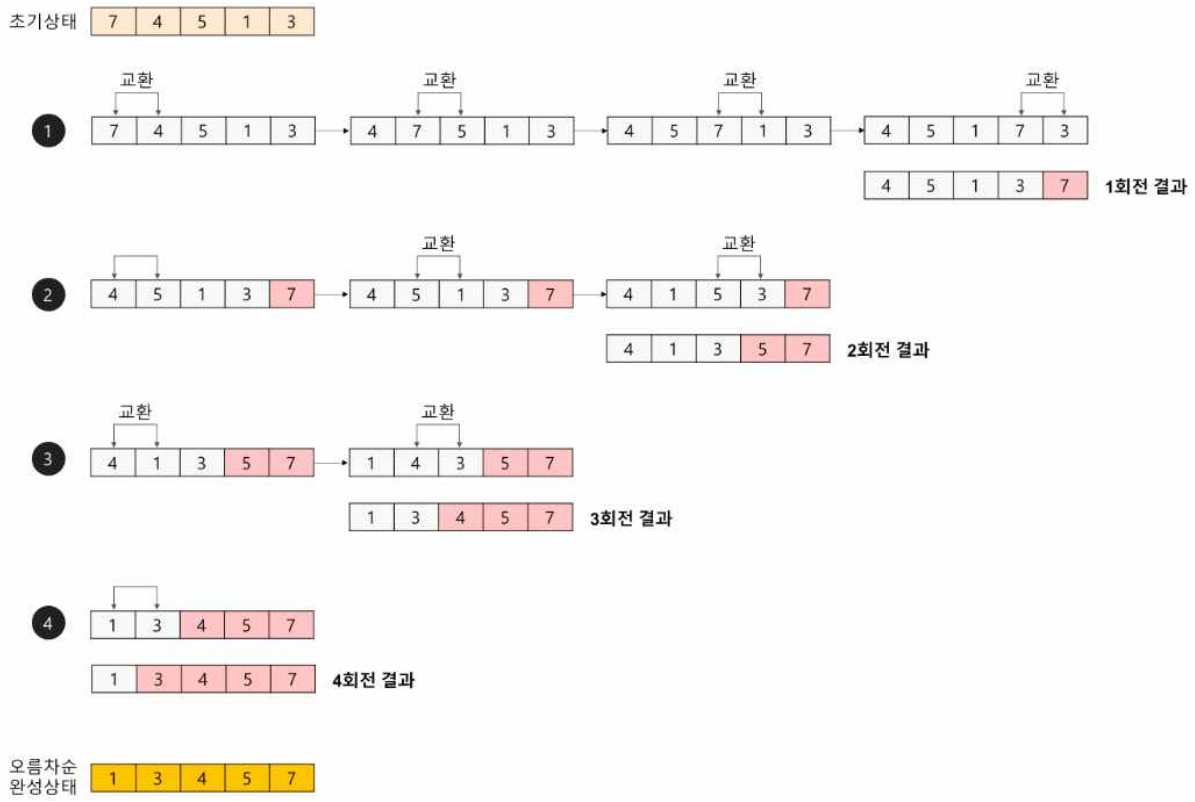
```
                        Output :

                    2 : 1026 (0.028500)
                    3 : 2023 (0.056194)
                    4 : 2988 (0.083000)
                    5 : 4086 (0.113500)
                    6 : 5018 (0.139389)
                    7 : 5978 (0.166056)
                    8 : 4928 (0.136889)
                    9 : 3992 (0.110889)
                   10 : 3096 (0.086000)
                   11 : 1907 (0.052972)
                   12 :  958 (0.026611)
```

> 4번 문제

▶ 아래는 버블정렬 알고리즘입니다. 구현하세요.

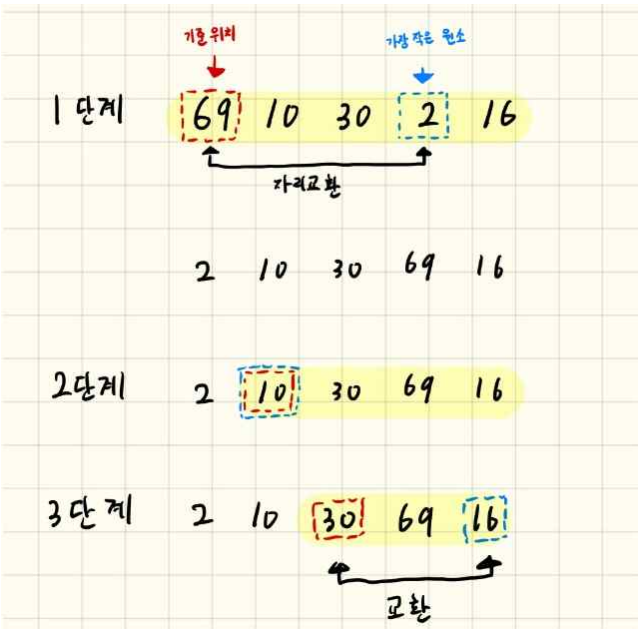
버블 정렬(bubble sort) 알고리즘의 예제

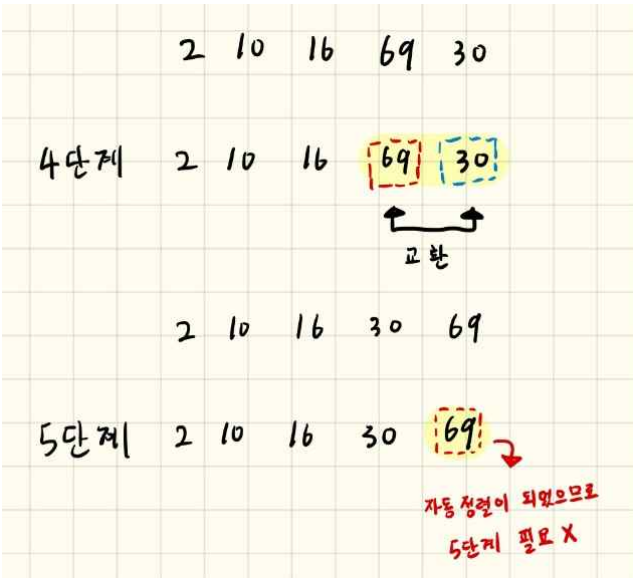
- 배열에 7, 4, 5, 1, 3이 저장되어 있다고 가정하고 자료를 오름차순으로 정렬해 보자.



> 5번 문제

▶ 아래는 선택(Selection)정렬 알고리즘입니다. 구현하세요.





1단계 : 첫 번째 자리를 기준 위치로 정하고, 전체 원소 중 가장 작은 원소인 2를 선택하여 자리를 교환한다.

2단계 : 두 번째 자리를 기준 위치로 정하고, 1번 위치를 제외한 나머지 원소 중 가장 작은 원소인 10과 교환한다.

3단계 : 세 번째 자리를 기준 위치로 정하고, 1번과 2번 위치를 제외한 나머지 원소 중 가장 작은 16과 교환한다.

4단계 : 네 번째 자리를 기준 위치로 정하고, 1번과 2번과 3번을 제외한 나머지 원소 중 가장 작은 30과 교환한다.

5단계 : 마지막 원소이므로 자동 정렬이 되었다. (4단계까지만 하면 된다.)

58장. Object 클래스

▷ 모든 클래스의 부모 클래스이다.

```
package ch06;
class Dog extends Object{  // Ctrl + Click : Object 클래스를 확인할 수 있다.
  String name = "토토";
}
class Cat{
  String name = "야옹이";
}

public class ObjectEx01{
  public static void main(String[] args){
    Object o1 = new Dog();  // 다형성이기 때문에 가능
    Object o2 = new Cat();  // 모든 클래스의 부모는 Object이기 때문에 가능

    Dog d1 = (Dog)o1; // 다운캐스팅 해야 사용 가능
    Cat c1 = (Cat)o2; // 다운캐스팅 해야 사용 가능
    System.out.println(d1.name);
    System.out.println(c1.name);
  }
}
```

▶ Cat과 Dog는 Object 를 상속받고 있으므로, 힙 메모리에
Object o1 = new Dog(); 선언하면 Object와 Dog 객체가 같이 뜬다.
그러나 가리키고 있는 곳이 Object이므로 Dog의 name에 접근이 불가능하게 된다.
※ 접근하기 위해서는 Dog d1 = (Dog)o1; 처럼 Dog 클래스를 가리키도록 다운캐스팅 해야 함.

```
package ch06;
class 궁수{
    String name = "궁수";
  }
class 전사{
    String name = "전사";
  }

public class ObjectEx02{
  public static void main(String[] args){
    궁수[] s1 = new 궁수[2];  // 궁수만 저장하는 배열
    s1[0] = new 궁수();
    s1[1] = new 궁수();
    System.out.println(s1[0].name);
    System.out.println(s1[1].name);
```

```
    // 그런데 나는 배열에 궁수와 전사를 같이 저장하고 싶다.
    Object[] s2 = new Object[2];
    s2[0] = new 궁수();
    s2[1] = new 전사();

    궁수 g1 = (궁수)s2[0];
    전사 g2 = (전사)s2[1];

//      System.out.println(((궁수)s2[0]).name);   // 명시적으로 다운캐스팅
//      System.out.println(((전사)s2[1]).name);   // 명시적으로 다운캐스팅
    System.out.println(g1.name);
    System.out.println(g2.name);

    System.out.println("======================");

        for (Object o : s2){
            if(o instanceof 궁수){
                System.out.println(((궁수)o).name);
            } else {
                System.out.println(((전사)o).name);
            }
        }
  }
}
```

59장. 제네릭이란

▷ 어떤 클래스(객체)를 만들 때 타입이 정해져 있지 않으면 Object로 만들어야 하는데 Object로 생성 시 다운캐스팅 등 처리를 해야하니까 이때 제네릭을 사용하면 편하다.. 이래서 제공하는 기능 임.

```
package ch06;
// 경우에 따라 문자열을 담기도 하고, 숫자를 담는 클래스
class 바구니{
      Object data;
   }
public class GenericEx01{
   public static void main(String[] args){
      바구니 s1 = new 바구니();
      s1.data = 1;
      System.out.println(s1.data);

      바구니 s2 = new 바구니();
      s2.data = "문자열";
      System.out.println(s2.data);
   }
}
```

▷ 위의 내용을 Generic으로 표현해 보자

※ Generic<> 안의 내용은 일반적으로 다음의 표현을 사용한다.

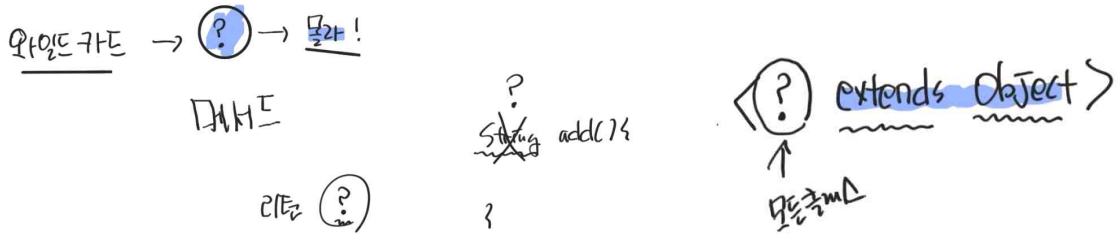
타입	설명
<T>	Type
<E>	Element
<K>	Key
<V>	Value
<N>	Number

```
package ch06;
class 호랑이{
  String name = "호랑이";
}

class 사자{
  String name = "사자";
}
// 경우에 따라 호랑이 클래스를 담기도 하고, 사자 클래스도 담는 클래스
//class 큰바구니{
class 큰바구니<T>{
    //Object data;
    T data;
  }
public class GenericEx02{
  public static void main(String[] args){
=========================================================
    큰바구니 s1 = new 큰바구니();
    s1.data = new 호랑이();
    //System.out.println(s1.data.name);  → 얘가 실행이 안됨.. name이 안보임. 다운캐스팅 필요!
    호랑이 h1 = (호랑이)s1.data;
    System.out.println(s1.data.name);
==================================  // 불편해서 안쓸거임!
    큰바구니<호랑이> s1 = new 큰바구니<>();  // 뒤는 생략가능
    s1.data = new 호랑이();
    System.out.println(s1.data.name);

    큰바구니<사자> s2 = new 큰바구니<>();  // 뒤는 생략가능
    s2.data = new 사자();
    System.out.println(s2.data.name);
  }
}
```

.

60장. 제네릭고급(와일드카드)

▷ ?




```
package ch06;
public class BasketBall {
	private String name = "농구공";

	public String getName() {
		return name;
	}
}

public class SoccerBall {
	private String name = "축구공";

	public String getName() {
		return name;
	}
}

public class Bag<T> {
	T ball;

	public T getBall() {
		return ball;
	}

	public void setBall(T ball) {
		this.ball = ball;
	}
}
```

```
public class GenericEx03{
  // 9시 : 축구, 12시 : 농구
  public static Bag<?> whatTheBall(int time){
        if(time == 9){
            BasketBall basketBall = new BasketBall();
            Bag<BasketBall> bag = new Bag<>();
            bag.setBall(basketBall);
            return bag;
        } else if(time == 12){
            SoccerBall soccerBall = new SoccerBall();
            Bag<SoccerBall> bag = new Bag<>();
            bag.setBall(soccerBall);
            return bag;
        }
        return null;
    }

    public static void main(String[] args) {
        Bag<?> bag1 = whatTheBall(9);
        Bag<?> bag2 = whatTheBall(12);
        System.out.println("가방에 " +
                ((BasketBall)bag1.getBall()).getName() +"이 있어요");
        System.out.println("가방에 " +
                ((SoccerBall)bag2.getBall()).getName() + "이 있어요");
    }
```

▷ 위와 같이 와일드카드 제네릭을 사용할 때도 다운캐스팅이 필요하다.

▷ 다운캐스팅을 하지 않으려면 농구공과 축구공을 품은 abstract class를 선언해서 일치 시킨다.

```
package ch06;
public abstract class Ball {
    public abstract String getName();
}

public class BBall extends Ball{
    private String name = "농구공";
    @Override
    public String getName() {
        return name;
    }
}
```

```
public class SBall extends Ball{
	private String name = "축구공";
	@Override
	public String getName() {
		return name;
	}
}

public class MyBag<T>{
	T ball;

	public T getBall() {
		return ball;
	}

	public void setBall(T ball) {
		this.ball = ball;
	}
}


public class GenericEx04{
  public static MyBag<? extends Ball> whatBall(int time){
		if(time == 9){
			Ball ball = new BBall();
			MyBag<Ball> myBag = new MyBag<>();
			myBag.setBall(ball);
			return myBag;
		} else if(time == 12){
			Ball ball = new SBall();
			MyBag<Ball> myBag = new MyBag<>();
			myBag.setBall(ball);
			return myBag;
		}
		return null;
	}
```

```
    public static void main(String[] args) {
        System.out.println(whatBall(9).getBall().getName());

        MyBag<? extends Ball> okBag = new MyBag<>();
        okBag = whatBall(12);
        System.out.println(okBag.getBall().getName());
    }
}
```