1. 총합과 평균: 배열의 모든 요소를 더해서 총합과 평균을 구한다.

```
package ch5;

class ArrayEx5 {
    public static void main(String[] args) {
        int sum =0;             // 총점을 저장하기 위한 변수
        float average = 0f;     // 평균을 저장하기 위한 변수

        int[] score = {100, 88, 100, 100, 90};

        for (int i=0; i < score.length ; i++ ) {
            sum += score[i];
        }

        average = sum / (float)score.length ; // 계산결과를 float로 얻기 위함.

        System.out.println("총점 : " + sum);
        System.out.println("평균 : " + average);
    }
}
```

2. 최대값과 최소값: 배열의 요소 중에서 제일 큰 값과 제일 작은 값을 섞는다. 배열의 첫 번째 요소의 값으로 최대값을 의미하는 변수 max와 최소값을 의미하는 변수 min을 초기화하고, 배열의 두번째 요소부터 (i = 1) for문을 돌면서 max와 min과 비교해서 max보다 크면 max에 저장, min보다 작으면 min에 저장하는 방식.

```java
package ch5;

class ArrayEx6 {
    public static void main(String[] args) {
        int[] score = {79, 88, 91, 33, 100, 55, 95};

        int max = score[0]; // 배열의 첫 번째 값으로 최대값을 초기화 한다.
        int min = score[0]; // 배열의 첫 번째 값으로 최소값을 초기화 한다.

        for(int i=1; i < score.length;i++) {
            if(score[i] > max) {
                max = score[i];
            } else if(score[i] < min) {
                min = score[i];
            }
        } // end of for

        System.out.println("최대값 :" + max);
        System.out.println("최소값 :" + min);
    } // end of main
} // end of class
```

---

3. 섞기(shuffle)

(1) 카드 섞기: 배열의 요소의 순서를 반복해서 바꾼다. 길이가 10인 배열을 생성하고 0~9의 숫자로 차례대로 초기화하여 출력한 다음, random()을 이용해서 배열의 임의의 위치에 있는 값과 첫번째 요소의 값을 교환하는 일을 numArr.length-1만큼 반복(또는 100번 반복).

```java
package ch5;

class ArrayEx7 {
    public static void main(String[] args) {
        int[] numArr = new int[10];

        for (int i=0; i < numArr.length ; i++ ) {
             numArr[i] = i;  // 배열을 0~9의 숫자로 초기화한다.
            System.out.print(numArr[i]);
        }
        System.out.println();

        for (int i=0; i < 100; i++ ) {
            int n = (int)(Math.random() * 10);  // 0~9중의 한 값을 임의로 얻는다.

            int tmp = numArr[0];
            numArr[0] = numArr[n];
            numArr[n] = tmp;
        }

        for (int i=0; i < numArr.length ; i++ )
            System.out.print(numArr[i]);
    } // main의 끝
}
```

```java
// 더 효율적인 코드
class ArrayEx7 {
    public static void main(String[] args) {
        int[] numArr = new int[10];

        for (int i=0; i < numArr.length ; i++ ) {
             numArr[i] = i;  // 배열을 0~9의 숫자로 초기화한다.
            System.out.print(numArr[i]);
        }
        System.out.println();

        for (int i=0; i < numArr.length; i++ ) {
            int n = (int)(Math.random() * 10);  // 0~9중의 한 값을 임의로 얻는다.

            int tmp = numArr[i];
            numArr[i] = numArr[n];
            numArr[n] = tmp;
        }

        for (int i=0; i < numArr.length ; i++ )
            System.out.print(numArr[i]);
    } // main의 끝
}
```

(2) 로또 번호 생성: 배열의 인덱스가 i인 값과 random()에 의해서 결정된 임의의 위치에 있는 값과 자리를 바꾸는 것을 6번 반복한다. 45개의 요소 중에서 앞의 6개의 요소만 임의의 위치에 있는 요소와 자리를 바꾸면 된다.

```java
package ch5;

class ArrayEx8 {
    public static void main(String[] args) {
        // 45개의 정수값을 저장하기 위한 배열 생성.
        int[] ball = new int[45];

        // 배열의 각 요소에 1~45의 값을 저장한다.
        for(int i=0; i < ball.length; i++)
            ball[i] = i+1;    // ball[0]에 1이 저장된다.

        int temp = 0;  // 두 값을 바꾸는데 사용할 임시변수
        int j = 0;     // 임의의 값을 얻어서 저장할 변수

        // 배열의 i번째 요소와 임의의 요소에 저장된 값을 서로 바꿔서 값을 섞는다.
        // 0번째 부터 5번째 요소까지 모두 6개만 바꾼다.
        for(int i=0; i < 6; i++) {
            j = (int)(Math.random() * 45); // 0~44범위의 임의의 값을 얻는다.
            temp    = ball[i];
            ball[i] = ball[j];
            ball[j] = temp;
        }

        // 배열 ball의 앞에서 부터 6개의 요소를 출력한다.
        for(int i=0; i < 6; i++)
            System.out.printf("ball[%d]=%d%n", i, ball[i]);
    }
}
```

4. 임의의 값으로 배열 채우기: 연속 또는 불연속적인 값들로 배열을 초기화한다.

1) 연속적인 범위의 임의의 값으로 배열 채우기.

```
for (i = 0; i < arr.length; i++) {
    arr[i] = (int)(Math.random() * 5); // 0~4 범위의 임의의 값을 저장
}
```

2) 불연속적인 범위의 값들로 배열 채우기: 불연속적인 값들을 담을 배열을 하나 더 생성해서, 임의로 선택된 index에 저장된 값으로 다든 배열의 요소들을 하나씩 채우면 된다.

```
package ch5;

import java.util.*;  // Arrays.toString()을 사용하기 위해 추가

class ArrayEx9 {
    public static void main(String[] args) {
        int[] code = { -4, -1, 3, 6, 11 };  // 불연속적인 값들로 구성된 배열
        int[] arr = new int[10];

        for (int i=0; i < arr.length ; i++ ) {
            int tmp = (int)(Math.random() * code.length);
            arr[i] = code[tmp];
        }

        System.out.println(Arrays.toString(arr));
    } // main의 끝
}
```

5. 정렬하기(sort): 오름차순, 내림차순으로 배열을 정렬. **버븐정련 앋고리즘**을 통해서 크기 순으로 정렬한다. 배열의 길이가 n일 때, 배열의 첫번째부터 n-1까지의 요소에 대해, 근접한 값과 크기를 비교하여 자리 바꿈을 반복한다.

```java
package ch5;

class ArrayEx10 {
    public static void main(String[] args) {
        int[] numArr = new int[10];

        for (int i=0; i < numArr.length ; i++ ) {
            System.out.print(numArr[i] = (int)(Math.random() * 10));
        }
        System.out.println();

        for (int i=0; i < numArr.length-1 ; i++ ) {
            boolean changed = false;    // 자리바꿈이 발생했는지를 체크한다.

            for (int j=0; j < numArr.length-1-i ;j++) {
                if(numArr[j] > numArr[j+1]) { // 옆의 값이 작으면 서로 바꾼다.
                    int temp = numArr[j];
                    numArr[j] = numArr[j+1];
                    numArr[j+1] = temp;
                    changed = true; // 자리바꿈이 발생했으니 changed를 true로.
                }
            } // end for j

            if (!changed) break;    // 자리바꿈이 없으면 반복문을 벗어난다.

            for(int k=0; k<numArr.length;k++)
                System.out.print(numArr[k]); // 정렬된 결과를 출력한다.
            System.out.println();
        } // end for i
    } // main의 끝
}
```

첫번째와 두번째 요소의 값을 비교해서 왼쪽 요소의 값이 크면 두 값의 위치를 바꾸고, 그렇지 않으면 바꾸지 않는다. 이러한 작업을 배열의 끝에 도달할 때까지 반복하면, 배열에서 제일 큰 값이 배열의 마지막 값이 된다. 비교횟수는 배열의 길이보다 1이 작은 값(배열.length-1)이다. 그 다음으로는 배열의 마지막 요소가 최대값이므로 비교할 필요가 없기 때문에 비교해야 할 범위가 하나씩 줄어든다. 비교작업(for문)을 반복할수록 비교해야 하는 범위가 하나씩 줄어드는 것이다. 그래서 원래는 배열의 길이에서 1이 작은 '배열.length-1'번을 비교해야 하는데, 매 반복마다 비교횟수가 1씩 줄어들기 때문에 바깥쪽 for문의 제어변수 i을 빼주는 것이다(배열.length-1-i). 한 번 수행되는 것만으로는 정렬이 되지 않기 때문에 비교작업(위의 for문)을 모두 '배열의 길이-1'만큼 반복해서 비교해야 한다. 그래서 바깥쪽 for문의 조건식이 '배열.length-1'이어야 하는 것이다. 또한 더 효율적인 작업을 위해 boolean형 변수를 두어서 자리바꿈이 없으면 break문을 수행하여 정렬을 마치도록 한다. 자리바꿈이 없다는 것은 정렬이 완료되었음을 뜻하기 때문이다.

---

6. 빈도수 구하기: 배열에 어떤 값이 몇 개 저장되어 있는지 세어서 보여 준다. 길이가 10인 배열을 만들고 0과 9 사이의 임의의 값으로 초기화한다. 그리고 이 배열에 저장된 각 숫자가 몇 번 반복해서 나타내는지를 세어서 배열 counter에 담은 다음 화면에 출력한다. 배열 counter에서, 배열 numArr에 저장된 값과 일치하는 인덱스의 요소에 저장된 값을 1증가 시킨다. 이 과정이 반복되고 나면, 배열 counter의 각 요소에는 해당 인덱스의 값이 몇 번 나타났는지 알 수 있는 값이 저장된다.

```java
package ch5;

class ArrayEx11 {
    public static void main(String[] args) {
        int[] numArr  = new int[10];
        int[] counter = new int[10];

        for (int i=0; i < numArr.length ; i++ ) {
            numArr[i] = (int)(Math.random() * 10); // 0~9의 임의의 수를 배열에 저장
            System.out.print(numArr[i]);
        }
        System.out.println();

        for (int i=0; i < numArr.length ; i++ ) {
            counter[numArr[i]]++;
        }

        for (int i=0; i < numArr.length ; i++ ) {
            System.out.println( i +"의 개수 :"+ counter[i]);
        }
    } // main의 끝
}
```

Resource: 자바의 정석