

day20 - java

☰ 태그	
📅 날짜	@2022년 10월 27일

ArrayList

자바에서 아주 많이 사용되는 기본적인 클래스 이다. 배열과 동일하게 연속된 메모리 공간을 사용하고 인덱스도 0에서 시작한다.

배열과의 차이는 배열을 그 크기가 한번 정해지면 기본적으로는 변경할 수 없지만 ArrayList는 크기가 가변적으로 변한다.

1. 생성

a. ArrayList를 사용하려면 먼저 import를 해야한다.

- `import java.util.ArrayList;`
`(import java.util. * ;)`

ArrayList의 기본 형태

```
ArrayList arr1 = new ArrayList();
```

```

1 package day20_1027;
2
3 import java.util.ArrayList;
4
5 public class day20_1027 {
6
7     public static void main(String[] args) {
8         ArrayList arr1 = new ArrayList();
9
10        arr1.add("Orange");
11        arr1.add("Cherry");
12        arr1.add(10);
13        System.out.println(arr1);
14
15    }
16
17 }
18

```

ArrayList는 기본적으로 데이터의 타입을 체크하지도 확인하지도 않고 모든 다양한 타입을 하나의 ArrayList에 담을 수 있다. 하지만 규모가 큰 클래스에서는 문제가 발생할 여지가 있다.

따라서 하나의 ArrayList에 하나의 타입 데이터만 저장하는 것을 권장한다.

ArrayList에 하나의 타입의 데이터만 저장하기 위한 방법 중 하나가 바로 **Generics**이다.

```

ArrayList_test.java

package day20_1027;

import java.util.ArrayList;

public class ArrayList_test {

    public static void main(String[] args) {
        // Generics를 String으로 설정하면 숫자는 넣을 수 없다.
        ArrayList<String> arr1 = new ArrayList<>();
        arr1.add("Orange");
        arr1.add("Cherry");
        arr1.add(10);
        System.out.println(arr1);
    }
}

```

```

        // Generics를 Integer로 설정하면 문자는 넣을 수 없다.
        ArrayList<Integer> arr2 = new ArrayList<>();
        arr2.add("Orange");
        arr2.add("Cherry");
        arr2.add(10);
        System.out.println(arr2);
    }
}

```

ArrayList_test1.java - remove

```

package day20_1027;

import java.util.ArrayList;

public class ArrayList_test1 {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        ArrayList<String> arr1 = new ArrayList<>();
        arr1.add("Orange");
        arr1.add("Cherry");
        arr1.add("Apple");
        System.out.println(arr1);

        arr1.remove("Cherry");

        System.out.println(arr1);
    }
}

```

ArrayList_test2.java - 값이 몇번째에 존재하는지 검색

```

package day20_1027;

import java.util.ArrayList;

public class ArrayList_test2 {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        ArrayList<String> arr1 = new ArrayList<>();
        arr1.add("Orange");
    }
}

```

```

arr1.add("Cherry");
arr1.add("Apple");
System.out.println(arr1);

System.out.println(arr1.indexOf("Apple"));
}
}

```

자료형은 `double`이고 요소 수가 5인 배열을 생성하고 이 배열의 모든 요소를 표시하는 프로그램을 작성하자.

실행 결과

```

a[0] = 0.0
a[1] = 0.0
a[2] = 0.0
a[3] = 0.0
a[4] = 0.0

```

```

package day20_1027;
import java.util.ArrayList;
public class ArrayList_test3 {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        ArrayList<Double> arr1 = new ArrayList<>();
        arr1.add(0.0);
        arr1.add(0.0);
        arr1.add(0.0);
        arr1.add(0.0);
        arr1.add(0.0);

        for(int i = 0; i < 5; i++) {

```

```

        System.out.println("a["+i+"]="+arr1.get(i));
    }

}

}

```

```

//인자 위치의 요소 추출
System.out.println(arr1.get(1));

//요소 위치 추출
System.out.println(arr1.indexOf("apple"));

//배열 값의 해시코드 반환
// 문자열이 데이터로 저장
//저장되는 위치를 나타냄
System.out.println(arr1.hashCode());

//배열에 다른 배열의 모든 요소 추가
arr1.addAll(arr2);

//인자 위치 요소 제거
System.out.println(arr1.remove(1)+" deleted.");

arr1.clear();

//배열 요소 모두 제거
arr1.removeAll(arr1);

System.out.println(arr1);

```

쓰레드 Thread

동작하고 있는 프로그램을 프로세스 라고 한다.

보통 한개의 프로세스는 한가지 일을 하지만 필요에 따라 쓰레드를 이용하면 한 프로세스 내에서 두개 또는 그 이상의 일을 **동시**에 할 수 있다.

Thread를 상속 받아 사용하는 방법

```

public class ThreadExam extends Thread {

```

Thread를 상속하면 run 메서드를 구현해야만 한다.

```
public void run() {  
    System.out.println("thread run.");  
}
```

sample 클래스가 Thread클래스를 상속했다. Thread 클래스에 run 메서드를 구현하면 객체명.start() 실행 시 run 메서드가 수행 된다.

```
public static void main(String[] args) {  
  
    ThreadExam sample = new ThreadExam();  
    sample.start(); // start()로 스레드를 실행한다.  
}
```

start() 실행 시 run메서드가 수행되도록 내부적으로 이미 동작이 정의 되어 있다.

```
ThreadrunExam.java  
  
package day20_1027;  
  
public class ThreadrunExam extends Thread {  
  
    int seq;  
  
    public ThreadrunExam(int seq) {  
        this.seq = seq;  
    }  
  
    public void run() {  
  
        // 스레드 시작  
        System.out.println(this.seq + "thread start.");  
        try {  
            Thread.sleep(2000); // 1초  
        } catch (Exception e) {  
  
        }  
        System.out.println(this.seq + "thread end.");  
        // 스레드 끝  
    }  
}
```

```

public static void main(String[] args) {

    for(int i = 0; i < 10; i++) { // 총 10개의 쓰레드를 생성

        Thread t = new ThreadrunExam(i);
        t.start();
    }
    System.out.println("최종 종료");

}
}

```

```

0thread start.
최종 종료
1thread start.
3thread start.
4thread start.
5thread start.
6thread start.
8thread start.
9thread start.
7thread start.
2thread start.
1thread end.
0thread end.
7thread end.
3thread end.
6thread end.
4thread end.
2thread end.
8thread end.
9thread end.
5thread end.

```

0번 쓰레드 부터 9번 쓰레드까지 순서대로 실행되지도 않고 그 순서가 일정하지도 않다.
따라서 쓰레드는 순서에 상관없이 동시에 실행된다는 것을 확인 할 수 있다.

쓰레드가 종료되기도 전에 main메서드가 종료 된다.

main 메서드 종료 시 “최종 종료”라는 문자열이 출력되는데 결과엔 중간에 출력이 된다.

쓰레드가 모두 수행되고 종료되기 전에 main메서드가 먼저 종료 되었다는 것은 모든 쓰레드가 종료된 뒤에 main메서드를 종료시키고 싶다면 어떻게 해야할까.

```
ThreadrunExam.java

package day20_1027;

import java.util.ArrayList;

public class ThreadrunExam extends Thread {

    int seq;

    public ThreadrunExam(int seq) {
        this.seq = seq;
    }

    public void run() {

        // 쓰레드 시작
        System.out.println(this.seq + "thread start.");
        try {
            Thread.sleep(2000); // 1초
        } catch (Exception e) {

        }
        System.out.println(this.seq + "thread end.");
        // 쓰레드 끝
    }

    public static void main(String[] args) {

        ArrayList<Thread> threads = new ArrayList<>();
        for(int i = 0; i < 10; i++) { // 총 10개의 쓰레드를 생성

            Thread t = new ThreadrunExam(i);
            t.start();
            threads.add(t);
        }
        for(int i = 0; i < threads.size(); i++) {
            Thread t = threads.get(i);
            try {
                t.join();
            } catch (Exception e) {

            }
        }
    }
}
```



```

        System.out.println("최종 종료");

    }

}

```

생성된 쓰레드를 담기 위해서 ArrayList의 객체 threads를 만든 후 생성 시에 생성된 객체를 threads 어레이리스트 배열에 저장하였다. 그리고 main 메서드가 종료되기 전에 threads에 담긴 각각의 thread에 join 메서드를 호출하여 쓰레드가 종료될 때까지 대기하도록 하였다. 쓰레드의 join메서드는 쓰레드가 전부 종료될 때까지 기다리게하는 메서드이다.

자바에서 난수 발생 방법

1. Math.random(); ← 비추

```

random_test.java

package day20_1027;

public class random_test {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        System.out.println(Math.random());
        System.out.println((int)(Math.random()*10));
        System.out.println((int)(Math.random()*100));
    }

}

```

2. rand.nextInt();

```

random_test1.java

package day20_1027;

import java.util.Random;

public class random_test1 {

    public static void main(String[] args) {

```

```

    Random rand = new Random();

    System.out.println(rand.nextInt());
    System.out.println(rand.nextInt(10));
    System.out.println(rand.nextInt(100));

    System.out.println(rand.nextBoolean());
    System.out.println(rand.nextFloat());
    System.out.println(rand.nextDouble());
}
}

```

random_test4.java - ArrayList 사용

```

package day20_1027;
import java.util.*;
public class random_test4 {

    public static void main(String[] args) {
        ArrayList<Integer> arr = new ArrayList<>();
        Random rand = new Random();

        for(int i = 0; i < 6; i++) {
            arr.add(rand.nextInt(45));
        }
        System.out.println(arr);
    }

}

//ArrayList를 사용한 방법
// 오름차순은 조금 더 알아볼것.

```

random_test2.java - set 사용 (중복데이터 제거)

```

package day20_1027;

import java.util.ArrayList;
import java.util.Random;
import java.util.HashSet;

public class random_test2 {

    public static void main(String[] args) {
        HashSet<Integer> set = new HashSet<>();

        while(set.size() < 6) {
            set.add((int)(Math.random()*45)+1);
        }
    }
}

```

```

    }

    System.out.println(set);
}
}

```

실행 예

요소 수:7

a[0] = 8

a[1] = 8

a[2] = 2

a[3] = 3

a[4] = 2

a[5] = 7

a[6] = 2

```

package day20_1027;
import java.util.Random;
import java.util.ArrayList;
import java.util.Scanner;
public class random_test3 {

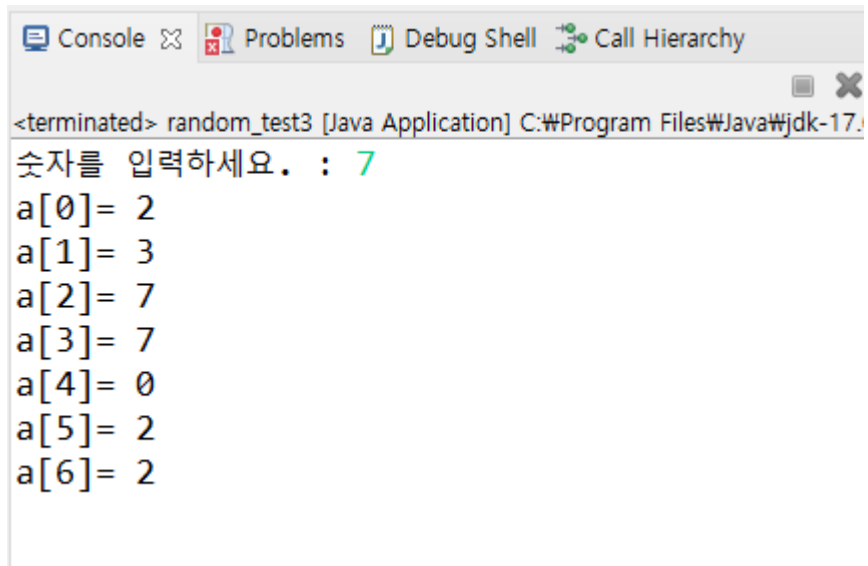
    public static void main(String[] args) {

        Random rand = new Random();
        ArrayList<Integer> arr = new ArrayList<>();
        Scanner sc = new Scanner(System.in);
        System.out.print("숫자를 입력하세요. : ");
        int a = sc.nextInt();
        int i;

        for(i = 0; i < a; i++) {
            int j= rand.nextInt(10);
            arr.add(j);
            System.out.println("a["+ i +"]= " + arr.get(i));
        }
    }
}

```

```
}  
  
}
```



```
Console Problems Debug Shell Call Hierarchy  
<terminated> random_test3 [Java Application] C:\Program Files\Java\jdk-17.  
숫자를 입력하세요. : 7  
a[0]= 2  
a[1]= 3  
a[2]= 7  
a[3]= 7  
a[4]= 0  
a[5]= 2  
a[6]= 2
```