

능동적 사고 방식의

java

강사 박주병

Park Ju Byeong

Park Ju Byeong



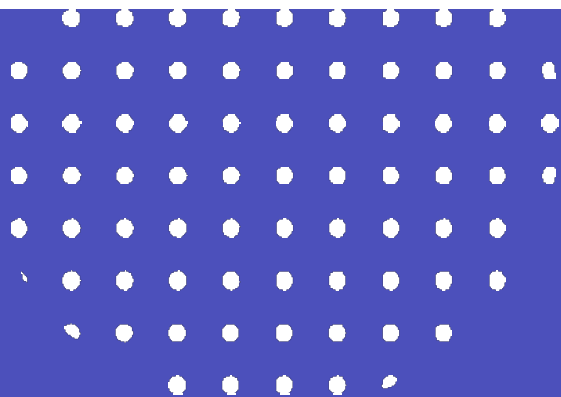
Part14 자료구조와 쓰레드

01 열거형

02 자료구조

03 쓰레드

04 실습 문제



01

열거형



```
public class Student {  
  
    static final int RUNNING = 0;  
    static final int SLEEPING = 1;  
    static final int STUDYING = 2;  
    static final int EATING = 3;  
  
    static int STATE;  
  
    public Student()  
    {  
        STATE = SLEEPING;  
    }  
  
    public void goToSchool()  
    {  
        switch(STATE)  
        {  
            case RUNNING: System.out.println("열심히 가는중이야!"); break;  
            case SLEEPING: System.out.println("..."); break;  
            case STUDYING: System.out.println("이미 등교 했는걸?"); break;  
            case EATING: System.out.println("이것만 먹고 갈게"); break;  
        }  
    }  
}
```

객체의 상태를 상수로서 표현했다.

```
Student st= new Student();  
  
st.STATE = Student.RUNNING;  
  
st.STATE = 13;
```

→ 의도하지 않은 값을 넣는다면?

STATE를 private으로 막고 set로만 값을 넣게 하고 필터링하면?

```
private static int STATE;  
  
public void setState(int state)  
{  
    if(state ==RUNNING  
        ||state ==SLEEPING  
        ||state ==STUDYING  
        ||state ==EATING  
        )  
    {  
        STATE=state;  
    }  
}
```

상태가 늘어날때마다 수정해야 하고 별로 좋아보이지 않는다.
그리고 매개변수가 그냥 int이니 사용자입장에서 뭘넣어야할지 직관적이지 않다.

열거형

```
public class Student {  
    public enum STATE{  
        RUNNING, SLEEPING, STUDYING, EATING  
    }  
    public STATE state;  
    public Student()  
    {  
        state = STATE.SLEEPING;  
    }  
    public void goToSchool()  
    {  
        switch(state)  
        {  
            case RUNNING: System.out.println("물집히 가는중이야!"); break;  
            case SLEEPING: System.out.println("..."); break;  
            case STUDYING: System.out.println("이미 등교 했는걸?"); break;  
            case EATING: System.out.println("이것만 먹고 갈게"); break;  
        }  
    }  
}
```

열거형 이름

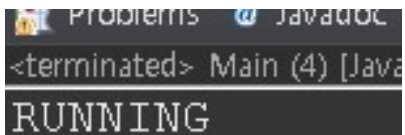
열거형 멤버들

타입이 열거형의 타입이다.
따라서 열거형의 멤버가 아니면
들어갈수 없다.

switch문에서는 열거형이름
없이 멤버를 바로 사용한다.

열거형 요소의 값

```
Student p1 = new Student();  
  
p1.state = Student.STATE.RUNNING;  
System.out.println(p1.state);
```



The screenshot shows an IDE window with a tab labeled 'Problems' and 'Javadoc'. Below the tab, the text '<terminated> Main (4) [Java' is visible. The console output displays 'RUNNING'.

열거형 요소의 순서정보

```
System.out.println(Student.STATE.RUNNING.ordinal());  
System.out.println(Student.STATE.SLEEPING.ordinal());  
System.out.println(Student.STATE.STUDYING.ordinal());  
System.out.println(Student.STATE.EATING.ordinal());
```

<terminated>

0

1

2

3

열거형의 멤버변수들에 내가 특정한 값을 주고 싶으면?


```
public enum STATE{ —————> 그러면 이거는 클래스명???  
    RUNNING(5),SLEEPING(0),STUDYING(-1),EATING(10);  
  
    private int value;  
    STATE(int value) —————> 이거 혹시.. 생성자 아냐??  
    {  
        this.value= value;  
    }  
}
```

```
public enum STATE{
    RUNNING(5),SLEEPING(0),STUDYING(-1),EATING(10);

    private int value;
    STATE(int value)
    {
        this.value= value;
    }
}
```



```
public class STATE{

    //RUNNING(5),SLEEPING(0),STUDYING(-1),EATING(10);

    static final STATE RUNNING = new STATE(5);
    static final STATE SLEEPING = new STATE(0);
    static final STATE STUDYING = new STATE(-1);
    static final STATE EATING = new STATE(10);

    public int value;
    STATE(int value)
    {
        this.value= value;
    }
}
```

static 변수라서 자기자신의 객체를 담을수 없지만 열거형은 이러한 사항을 특별히 처리해놓은 형태인것이다.

```
public enum STATE{  
    RUNNING(5),SLEEPING(0),STUDYING(-1),EATING(10);  
    |  
    public int value;  
    STATE(int value)  
    {  
        this.value= value;  
    }  
}
```

STATE 타입의 참조변수이름

클래스명

```
System.out.println(Student.STATE.RUNNING.value);  
System.out.println(Student.STATE.SLEEPING.value);  
System.out.println(Student.STATE.STUDYING.value);  
System.out.println(Student.STATE.EATING.value);
```

STATE 타입의
static 참조변수

STATE 의 멤버변수

1-1 실습문제 (normal)

People 클래스를 만들고 열거형을 사용하여 취미를 저장하는 멤버변수를 만들어 보자.

- 취미는 soccer,baseball,cook,running 가 있다.

```
People p1 = new People();  
  
p1.hobby = People.HOBBY.BASEBALL;  
System.out.println(p1.hobby);
```

```
<terminated> Main (4  
RUNNING
```

1-1 문제풀이 (normal)

Pai

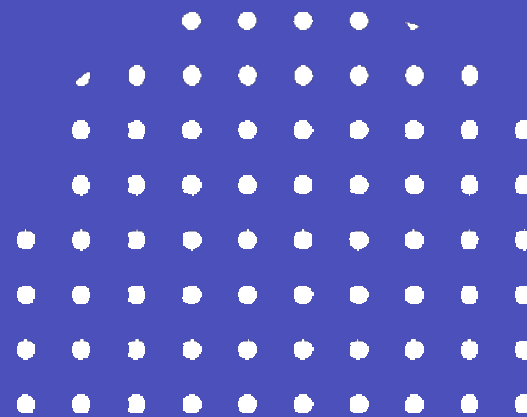
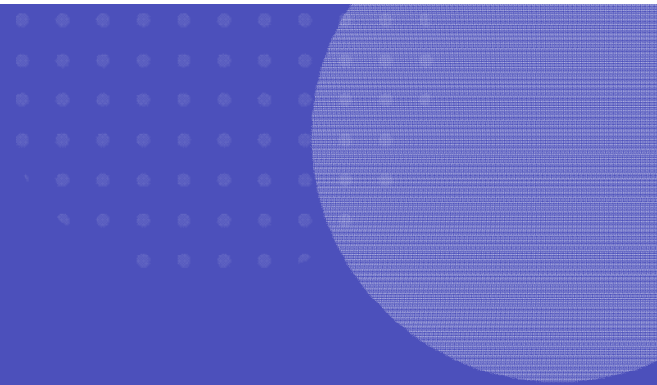
byeong

Park Ju

g

— 02

자료구조



지식백과

자료구조 Data Structure

개념 및 정의] 자료구조(資料構造, data structure)는 컴퓨터에서 처리할 자료를 효율적으로 관리하고 구조화시키기 위한 학문이다. 즉, 자료를 효율적으로 사용하기 위해서 자료의 특성에 따라서 분류하여 구성하고 저장 및 처리하는 모든 작업을...
프로그래밍 언어, 접근방법 및 주요 연구영역, 주요 용어 및 관련 직업군
학문명백과 : 공학

자료구조 data structure

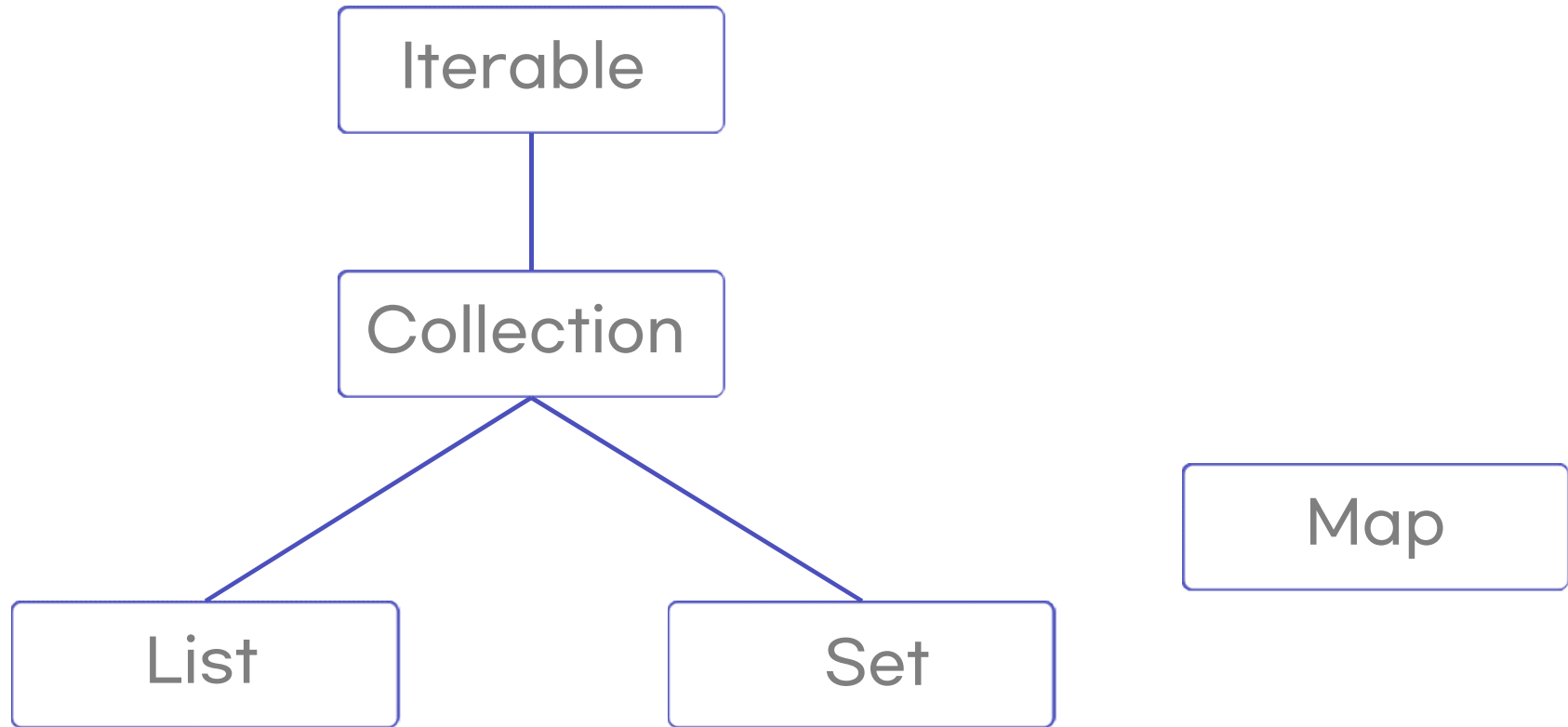
컴퓨터에서 자료를 효율적으로 관리하고 구조화시키는 방법. 물건을 정리해놓으면 깔끔하여 필요한 물건을 찾기도 쉽습니다. 컴퓨터도 마찬가지입니다. 컴퓨터가 효율적으로 문제를 처리하기 위해서는 자료를 보관하고 정리하는 기술이 필요한데...
천재학습백과 초등 소프트웨어 용어사전

자료구조

자료구조란 자료의 사용 방법이나 성격에 따라 효율적으로 사용하기 위하여 조직하고 저장하는 방법입니다. 컴퓨터작업에서 어떤 자료구조를 사용하느냐에 따라 처리시간을 단축하고 기억장치 공간을 효율적으로 사용할 수 있습니다. 일상생활에...
소프트웨어 어휘다지기 - 중등

데이터들을 효율적으로 관리하기 위한 클래스

자료구조의 인터페이스 계층도

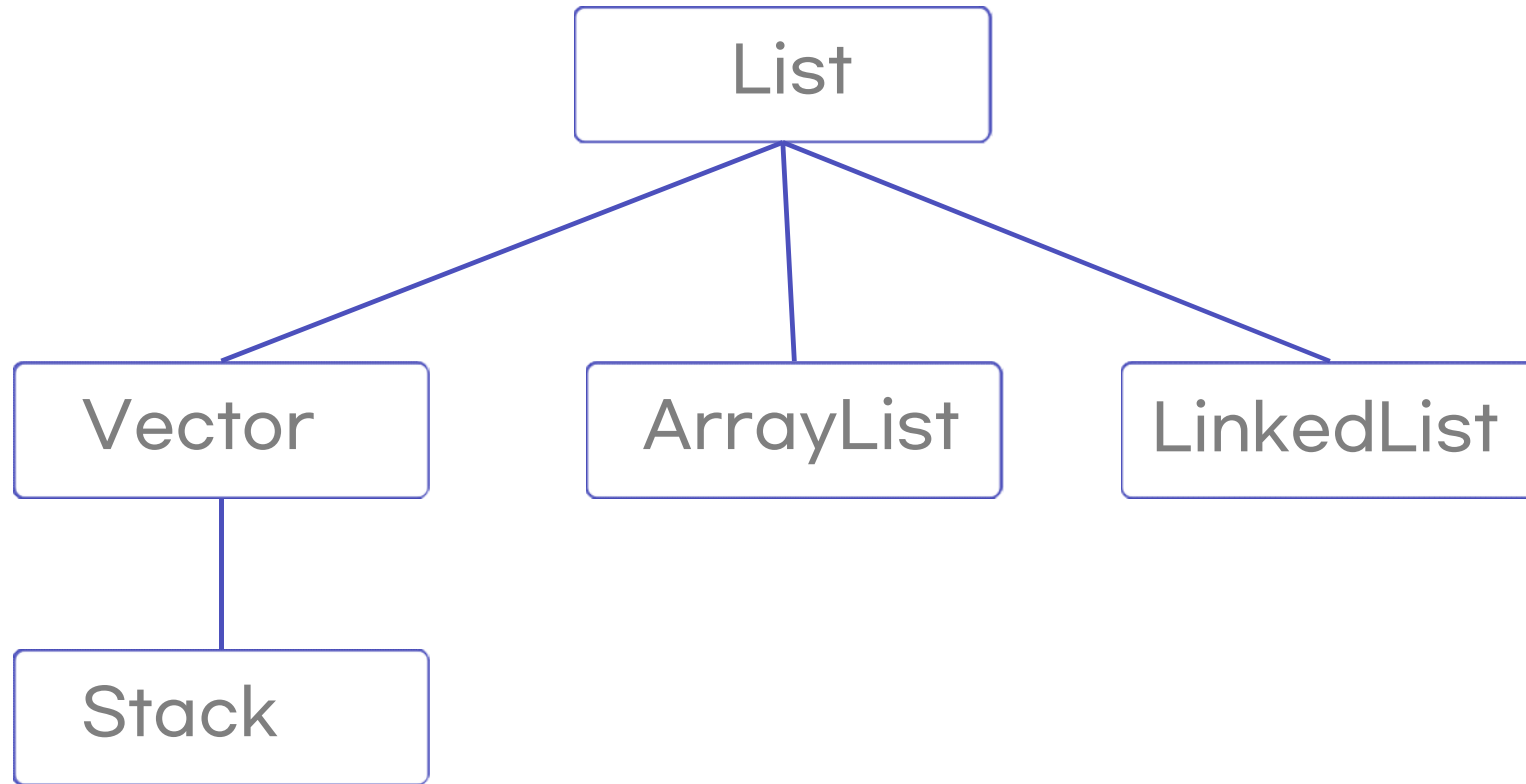


모든 자료구조들은 3개중 하나의 자식들이다.

인터페이스	설명
List	<p>차례대로 모여 있는 데이터 중복허용</p> <p>ArrayList, LinkedList, Stack, Vector 등등</p>
Set	<p>순서가 없는 데이터 중복허용 안함</p> <p>HashSet, TreeSet</p>
map	<p>키와 값의 쌍으로 이루어진 데이터 키는 중복허용 안함</p> <p>HashMap, TreeMap, Hashtable, Properties</p>

List 인터페이스

- 중복허용
- 저장순서 있음



ArrayList

- Vector를 개선한것, 따라서 vector보다 ArrayList를 권장한다.

```
List<String> list = new ArrayList();

list.add("가");
list.add("나");
list.add("다");
list.add("라");

System.out.println(list.size()); //요소의 갯수를 반환한다.
System.out.println(list); //toString을 오버라이딩하였다
list.remove(2); //임의 위치의 요소를 삭제할수 있다. 그뒤에 index번호는 자동으로 당겨진다.
System.out.println(list);

System.out.println(list.indexOf("나")); //해당 요소의 위치를 반환한다.
if(list.contains("나")) //해당요소가 들어있는지 여부만 판단한다. indexOf보다 빠르다.
    System.out.println("나 포함되어 있습니다");
```

```
list.add(0,"마"); //임의 위치에 요소 삽입
System.out.println(list);
System.out.println(list.lastIndexOf("라")); //뒤에서 부터 탐색한다.
Collections.sort(list); //기본 오름차순 정렬
|
String[] arr=new String[10];
list.toArray(arr); //만들어둔 배열에 담는다.

for(String str : arr)
    System.out.println(str);
```

```
[마, 가, 나, 라]
3
가
나
라
마
null
null
null
null
null
null
null
```

```
<terminated> Main (1) [Java A
4
[가, 나, 다, 라]
[가, 나, 라]
1
나 포함되어 있습니다
```

ArrayList

```
* The capacity of the ArrayList is the  
* empty ArrayList with elementData ==  
* will be expanded to DEFAULT_CAPACITY  
*/  
transient Object[] elementData; // non-  
  
/**  
 * The size of the ArrayList (the number  
 *  
 * @serial
```

내부에서 배열을 사용 한다.

```
List<String> list = new ArrayList(); //기본은 길이10 배열을 생성한다.
```

```
list.add("1");  
list.add("2");  
list.add("3");  
list.add("4");  
list.add("5");  
list.add("6");  
list.add("7");  
list.add("8");  
list.add("9");  
list.add("10");  
list.add("11");
```

→ 내부적으로 배열[20]을 만들어 교체한다.

```
System.out.println(list.size());
```

Cons

<terminat

11

→ 그렇다고 ArrayList의 사이즈가 20인건 아니다.

```
List<String> list = new ArrayList(); //기본은 길이10 배열을 생성한다.
```

```
List<String> list2 = new ArrayList(2); //생성자로 배열의 사이즈를 지정할수 있다.
```

→ 작은 배열교체는 성능저하를 불러온다.

LinkedList

- 연속되지 않은 공간에 데이터를 할당하여 앞뒤로 링크 시켜 놓음

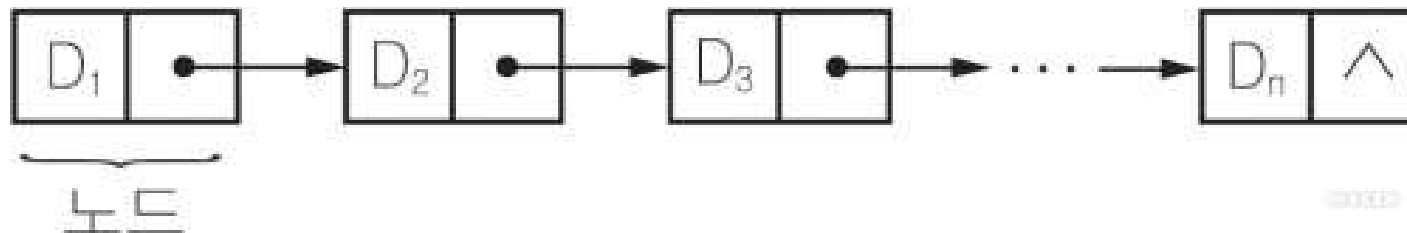
ArrayList



중간 삭제시 뒤의 요소들을
앞으로 당기는 작업을 한다.

LinkedList

데이터 링크



LinkedList

```
List<String> list = new LinkedList(); //기본은 길이10 배열을 생성한다.  
list.add("1");  
list.add("2");  
list.add("3");  
list.add("4");  
list.add("5");  
list.add("6");  
list.add("7");  
list.add("8");  
list.add("9");  
list.add("10");  
list.add("11");  
  
System.out.println(list.size());
```

이 역시 List 인터페이스를 상속받아
구현한 클래스 이기에
ArrayList 와 사용법이 똑같다.

다형성을 습관화 하자.



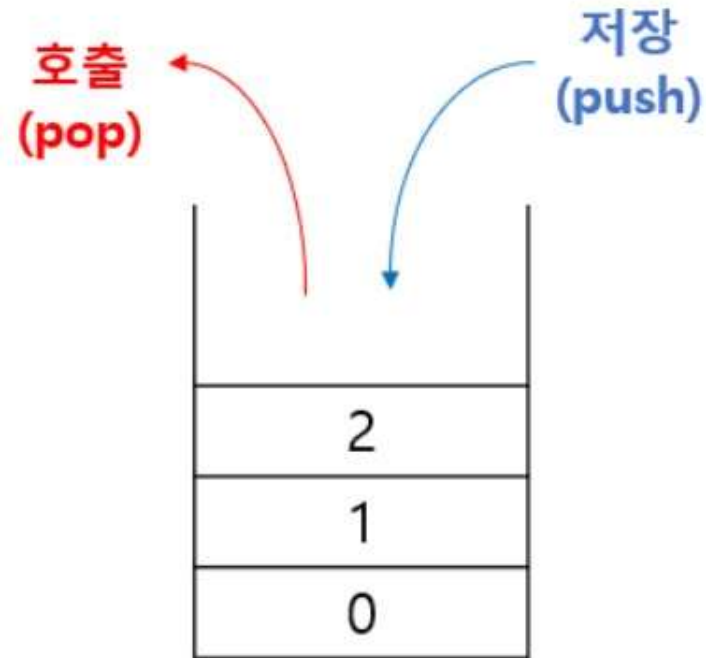
ArrayList VS LinkedList

- 연속된 공간을 할당하고 서로의 주소를 보관할 변수를 사용하지 않아 메모리가 절약된다.
- 요소의 가장뒤에 값이 추가,삭제 되는 경우에 빠르다.
- 요소의 접근시간이 빠르다.

- 서로의 주소를 가지고 있어야 하기에 메모리를 상대적으로 많이 차지 않는다.
- 요소의 중간에 추가, 삭제시 빠르다.
- 요소의 접근시간이 느리다.

Stack

- 후입선출(나중에 들어간것이 먼저 나온다) 방식의 자료 구조



Stack 예제

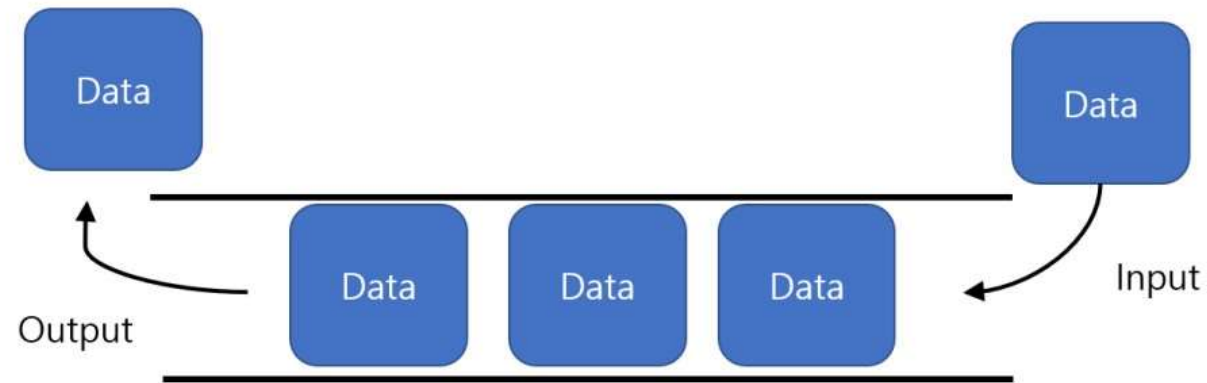
```
Stack<String> st = new Stack();

st.push("가");
st.push("나");
st.push("다");

System.out.println(st.pop()); //데이터를 꺼냄과 동시에 요소를 삭제한다.
System.out.println(st.pop());
System.out.println(st.peek()); //데이터를 꺼내기만 하고 요소는 그대로 놔둔다.
System.out.println(st.peek()); //데이터를 꺼내기만 하고 요소는 그대로 놔둔다.
System.out.println(st.peek()); //데이터를 꺼내기만 하고 요소는 그대로 놔둔다.
```

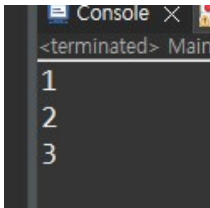
Queue

- 선입선출(먼저들어간것이 먼저 나온다) 방식의 자료구조



Queue 예제

```
Queue<String> q = new LinkedList();  
  
q.offer("1");  
q.offer("2");  
q.offer("3");  
  
System.out.println(q.poll());  
System.out.println(q.poll());  
System.out.println(q.poll());
```



```
Console X  
<terminated> Main  
1  
2  
3
```

Stack

뒤에 들어간 요소에서 지속적으로 추가 삭제가 일어난다.
ArrayList로 구현하는것이 적합하다.

Queue

먼저 들어간것이 먼저 삭제된다. ArrayList로 만들면 뒤에 있는
요소들이 전부 이동해야 한다.
LinkedList로 구현하는것이 적합하다.

HashSet

Set인터페이스를 구현하고 요소들의 값이 **중복되지 않는다.**

```
Set set = new HashSet();  
  
set.add(1);  
set.add(2);  
set.add(1);  
set.add(2);  
  
System.out.println(set);  
System.out.println(set.size());
```

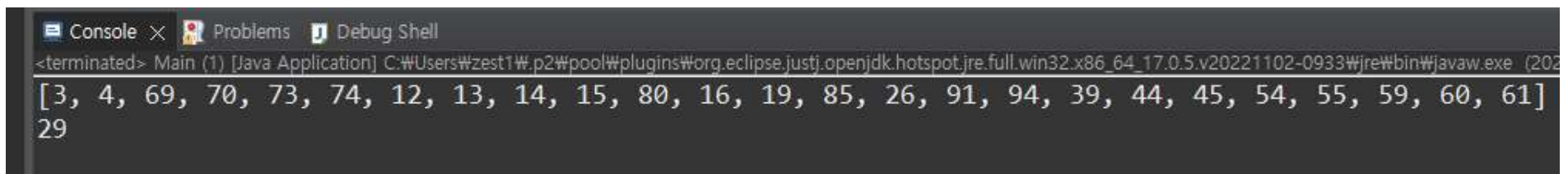
```
<terminated> Main (
[1, 2]
2
```

HashSet 예제

```
Set set = new HashSet();

int count=0;
while(set.size()<25)
{
    int random = ((int)(Math.random()*100))+1;
    set.add(random);
    count++;
}

System.out.println(set);
System.out.println(count);
```



Console × Problems Debug Shell
<terminated> Main (1) [Java Application] C:\Users\zest1\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.5.v20221102-0933\jre\bin\javaw.exe (20221102-0933)
[3, 4, 69, 70, 73, 74, 12, 13, 14, 15, 80, 16, 19, 85, 26, 91, 94, 39, 44, 45, 54, 55, 59, 60, 61]
29

1~100 사이의 랜덤한 숫자를 중복되지 않게 25개를 뽑는다.

HashMap

- Map 인터페이스를 구현하고 중복되지 않는 Key와 Value로 이루어져 있다.

```
Map map = new HashMap();
```

```
map.put("사과", "apple");  
map.put("자동차", "car");  
map.put("책", "book");  
map.put("음식", "food");  
map.put("사과", "apple");
```

```
System.out.println(map);  
System.out.println(map.get("음식"));  
map.remove("사과");  
System.out.println(map);
```

→ 검색 속도가 빠르다.

```
<terminated> main (1) [Java Application] C:\Users\wzest\Idea\p2\src\main\plugins\src  
{책=book, 사과=apple, 음식=food, 자동차=car}  
food  
{책=book, 음식=food, 자동차=car}
```


1-1 실습문제 (normal)

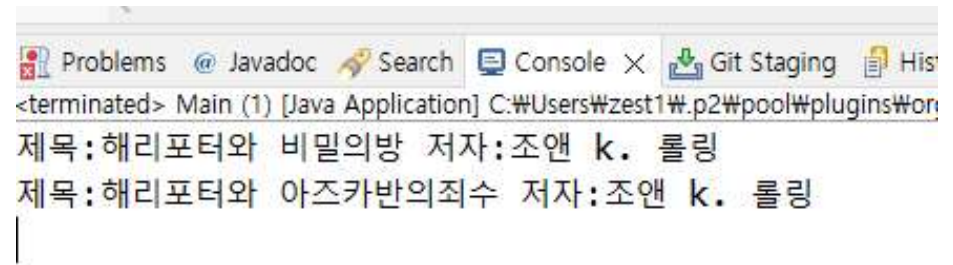
ArrayList를 사용하여 도서의 제목과 저자를 관리하는 프로그램을 만들어 보자. 각 도서는 Book 클래스의 객체로 관리된다.

ArrayList에 제목과 저자 데이터가 있는 Book 객체 여러 개를 넣고 특정 저자의 책들만 출력해보자.

Book 클래스(코드 예시를 보고 이외에 필요한 메서드나 생성자를 적절하게 만들어보자)

- 멤버변수 : String title , String author

```
bookList.add(new Book("해리포터와 비밀의방", "조앤 k. 롤링"));  
bookList.add(new Book("해리포터와 아즈카반의죄수", "조앤 k. 롤링"));  
bookList.add(new Book("백종원이 추천하는 집밥 메뉴", "백종원"));  
bookList.add(new Book("물입", "황농문"));
```



Park Ju Byeong

1-1 문제풀이 (normal)

■

Par. → Byeong

Park Ju → eong

1-2 실습문제 (hard)

제네릭을 활용하여 나만의 MyStack 자료구조를 만들자.

- void push(T value), T pop() 메서드를 구현하자.
- 내부에서 실질적으로 데이터는 ArrayList로 관리하자.
- ex) push의 내부는 ArrayList 객체의 add 메서드를 사용하여 요소를 추가한다.

```
MyStack<String> stack = new MyStack<>();

stack.push("1번");
stack.push("2번");
stack.push("3번");

System.out.println(stack.pop());
System.out.println(stack.pop());
System.out.println(stack.pop());
```

```
<terminated>
3번
2번
1번
```

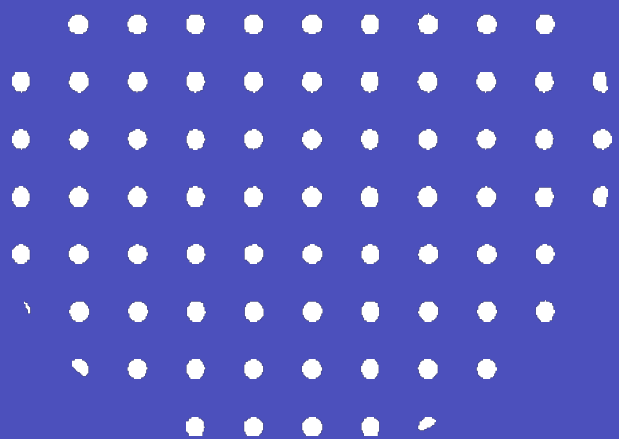
1-2 문제풀이 (hard)

Park Ju

'19

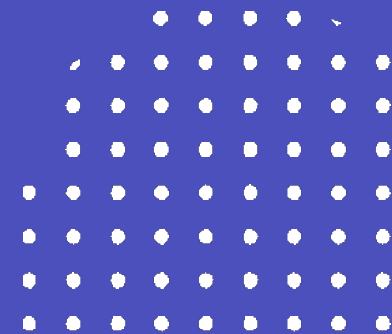
ju Byeong

r

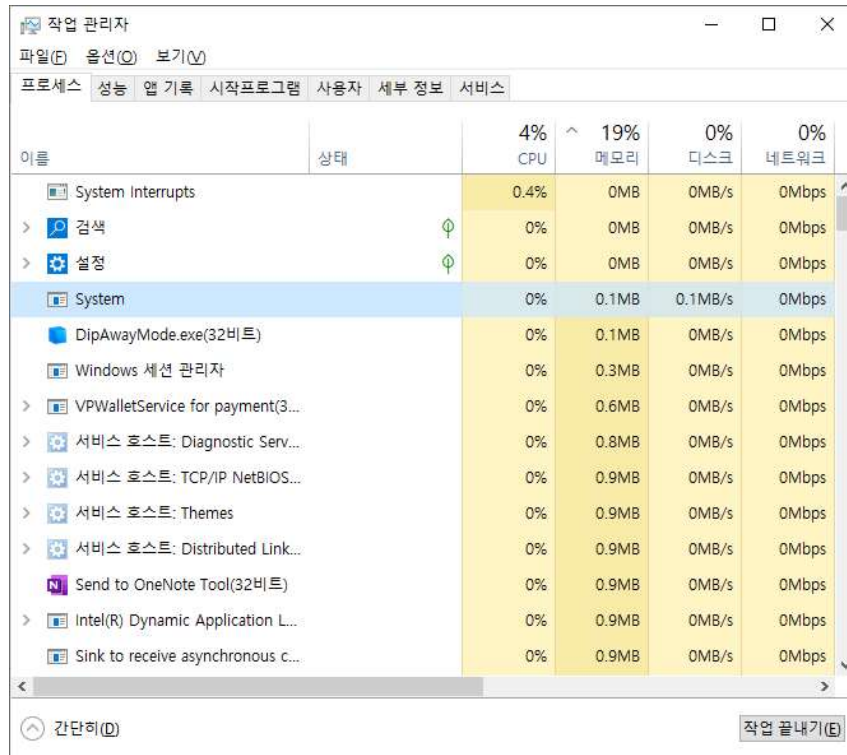


03

쓰레드

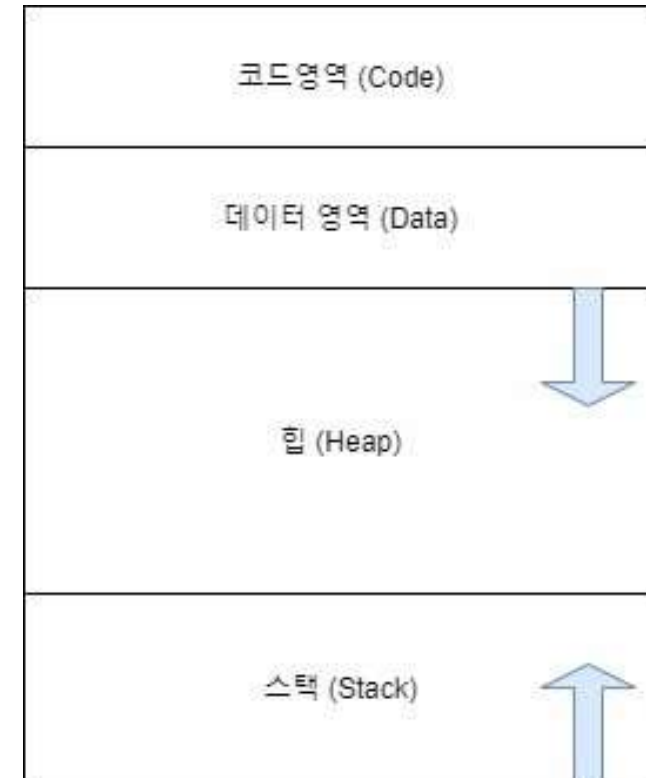


프로세스

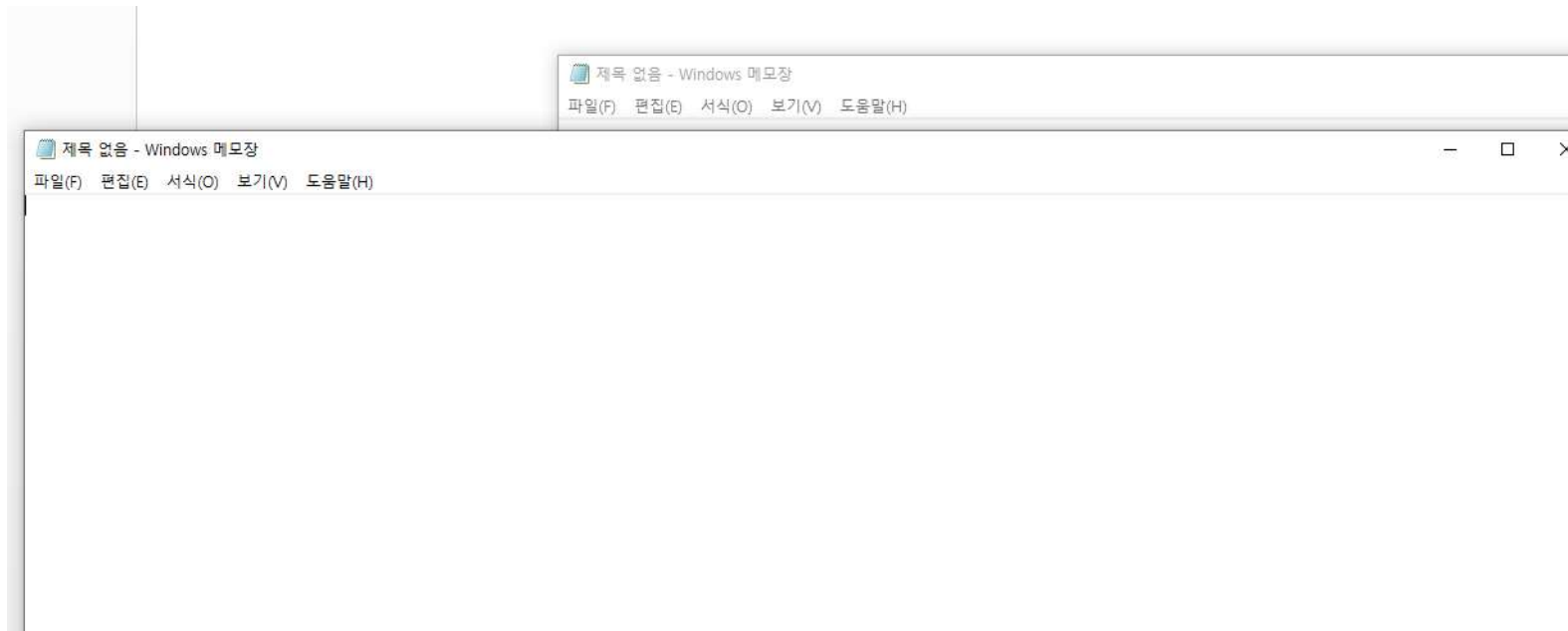


The screenshot shows the Windows Task Manager window with the 'Processes' tab selected. The table lists various system and user processes with their respective CPU, memory, disk, and network usage.

이름	상태	4% CPU	19% 메모리	0% 디스크	0% 네트워크
System Interrupts		0.4%	0MB	0MB/s	0Mbps
> 검색		0%	0MB	0MB/s	0Mbps
> 설정		0%	0MB	0MB/s	0Mbps
System		0%	0.1MB	0.1MB/s	0Mbps
DipAwayMode.exe(32비트)		0%	0.1MB	0MB/s	0Mbps
Windows 세션 관리자		0%	0.3MB	0MB/s	0Mbps
> VPMWalletService for payment(3...		0%	0.6MB	0MB/s	0Mbps
> 서비스 호스트: Diagnostic Serv...		0%	0.8MB	0MB/s	0Mbps
> 서비스 호스트: TCP/IP NetBIOS...		0%	0.9MB	0MB/s	0Mbps
> 서비스 호스트: Themes		0%	0.9MB	0MB/s	0Mbps
> 서비스 호스트: Distributed Link...		0%	0.9MB	0MB/s	0Mbps
Send to OneNote Tool(32비트)		0%	0.9MB	0MB/s	0Mbps
> Intel(R) Dynamic Application L...		0%	0.9MB	0MB/s	0Mbps
Sink to receive asynchronous c...		0%	0.9MB	0MB/s	0Mbps



하나의 실행되고 있는 프로그램이다.



메모장을 2개 띄워 봤다면 메모장
프로세스가 2개가 돌고 있는것이다.

쓰레드

프로세스 내부에서의 실행흐름 단위이다.
프로세스는 최소한 1개이상의 쓰레드를 가진다.

프로세스

```
public static void main(String[] args) {  
    // TODO Auto-generated method stub  
  
    |  
  
    Box<String> box = new Box<>();  
    Box<Integer> box2 = new Box<>();  
    box.setItem("가나다라");  
  
    System.out.println(box.getItem());  
}
```

실행흐름(main 쓰레드)

이미 쓰레드를 사용하고 있었던 것이다.

프로세스

```
public static void main(String[] args) {  
    // TODO Auto-generated method stub  
  
    |  
  
    Box<String> box = new Box<>();  
    Box<Integer> box2 = new Box<>();  
    box.setItem("가나다라");  
  
    System.out.println(box.getItem());  
}
```

실행흐름

```
public static void main(String[] args) {  
    // TODO Auto-generated method stub  
  
    |  
  
    Box<String> box = new Box<>();  
    Box<Integer> box2 = new Box<>();  
    box.setItem("가나다라");  
  
    System.out.println(box.getItem());  
}
```

실행흐름

//http는 비연결성이기때 매 채팅마다 연결을 새로 해줘야 한다.

```
con = (URLConnection) url.openConnection();
con.setRequestMethod("POST");
con.setRequestProperty("Content-Type", "application/json; utf-8");
con.setRequestProperty("Authorization", "Bearer "+key);
con.setRequestProperty("Retry-After", "3600");

con.setDoOutput(true);

try(OutputStream out = con.getOutputStream())
{
    //문자열을 보내기전에 개행을 제거 하고 보낸다.
}
```

→ 서버와의 통신으로 행이 걸린다.

↓

실행흐름



서버로 부터 응답을 받은후
진행된다.

↓

실행흐름

단일쓰레드

멀티쓰레드

```
//http는 비연결성이기에 매 채팅마다 연결을 새로 해줘야 한다.  
con = (URLConnection) url.openConnection();  
con.setRequestMethod("POST");  
con.setRequestProperty("Content-Type", "applicat  
con.setRequestProperty("Authorization","Bearer "  
con.setRequestProperty("Retry-After","3600");  
  
con.setDoOutput(true);  
  
try(OutputStream out = con.getOutputStream())  
{  
    //문자열을 보내기전에 개행을 제거 하고 보낸다.  
    byte[] inputBytes = this.msg.toString().repl  
    out.write(inputBytes,0,inputBytes.length);  
    Thread.sleep(2000);  
    con.getResponseCode();  
}catch(Exception ex)  
{  
}
```

서버의 응답과
상관없이 진행할
코드

Park Ju Byeong

Park Ju Byeong

쓰레드 구현

1. Thread 를 상속 받는 방법
2. Runnable인터페이스를 구현하는 방법

```
public class MyThread extends Thread{  
    |  
    @Override  
    public void run() {  
        System.out.println("쓰레드 내부에서 진행될 코  
        |  
        for(int i =0; i<100;i++)  
            System.out.println("서브쓰레드"+i);  
    }  
}  
  
public static void main(String[] args) {  
    MyThread t1 = new MyThread();  
    System.out.println("쓰레드 시작 전");  
    t1.start();  
    for(int i =0; i<100;i++)  
        System.out.println("메인쓰레드"+i);  
}
```

서브쓰레드

메인쓰레드

<terminated> Main (1) [Java

쓰레드 시작 전

메인쓰레드0

메인쓰레드1

메인쓰레드2

메인쓰레드3

메인쓰레드4

메인쓰레드5

메인쓰레드6

메인쓰레드29

메인쓰레드30

메인쓰레드31

쓰레드 내부에서 진행될 코드입니다.

메인쓰레드32

메인쓰레드33

서브쓰레드0

서브쓰레드1

서브쓰레드2

서브쓰레드3

메인쓰레드34

메인쓰레드35

메인쓰레드36

메인쓰레드37

서브쓰레드7

메인쓰레드60

서브쓰레드8

메인쓰레드61

서브쓰레드9

서브쓰레드10

메인쓰레드62

서브쓰레드11

서브쓰레드12

서브쓰레드13

Park Ju Byeong

Runnable 인터페이스 구현

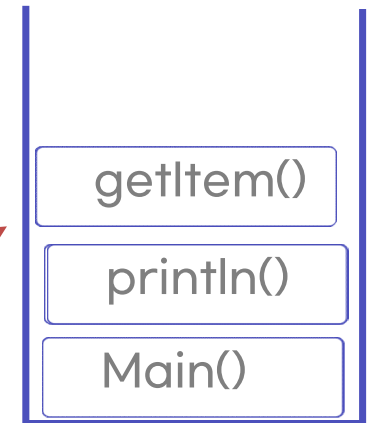
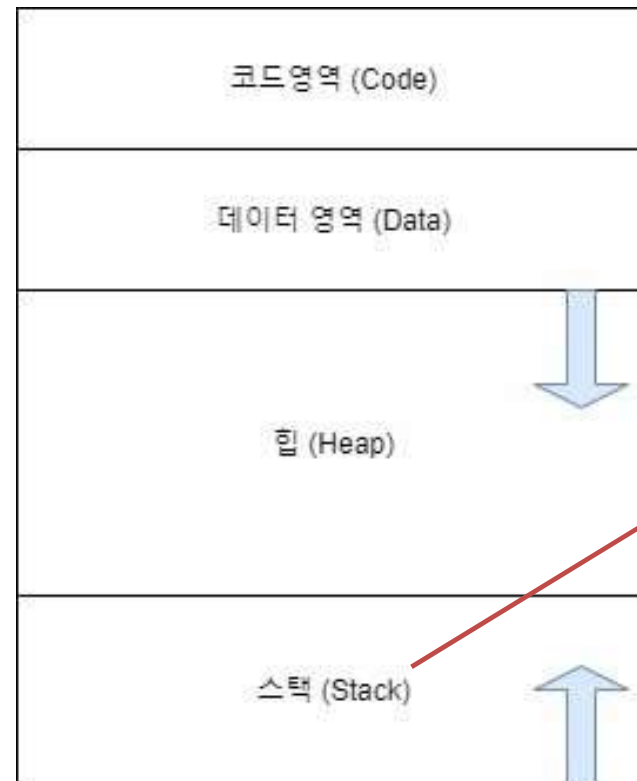
```
public class MyThread2 implements Runnable{  
    @Override  
    public void run() {  
        System.out.println("MyThread2");  
    }  
}
```

→ 서브쓰레드로 실행되는 코드이다.

```
Thread th = new Thread(new MyThread2()); // 생성자로 러너블을 구현한 객체를 넘겨준다.  
th.start();
```

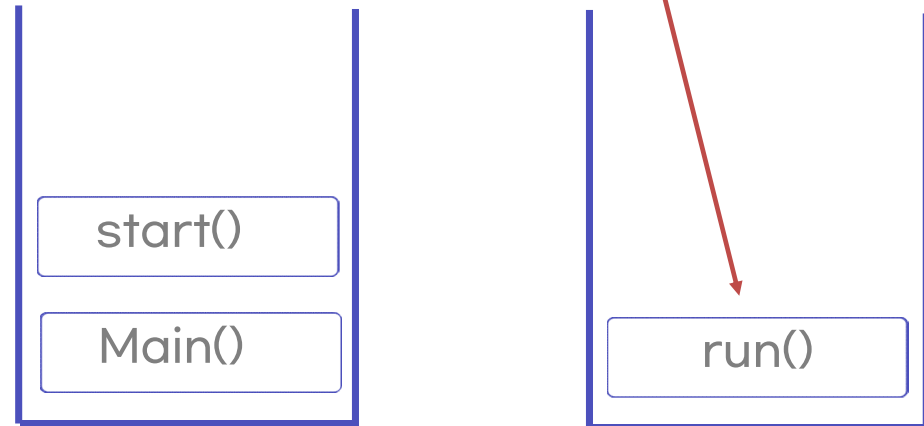
그런데 우리가 구현한건 `run()` 메서드인데 왜 `start()`를 호출해서 스레드를 시작할까?

```
public static void main(String[] args) {  
    // TODO Auto-generated method stub  
  
    Box<String> box = new Box<>();  
    Box<Integer> box2 = new Box<>();  
    box.setItem("가나다라");  
  
    System.out.println(box.getItem());  
}
```



```
public static void main(String[] args) {  
    // TODO Auto-generated method stub  
  
    Thread th = new Thread(new MyThread2()  
    th.start();  
  
}
```

쓰레드 스케줄러



start()은 메인쓰레드에서 실행되며 run()은 서브쓰레드에서 실행시키기 위해 직접호출하지 않는다.

start() 말고 run()을 직접 실행하면 어떻게 될까?

```

public static void main(String[] args) {
    // TODO Auto-generated method stub

    Thread th = new Thread(new MyThread2());
    th.run();

    for(int i=0;i<10;i++)
        System.out.println("메인쓰레드"+i);
}

```

```

<terminated> Main (
서브쓰레드0
서브쓰레드1
서브쓰레드2
서브쓰레드3
서브쓰레드4
서브쓰레드5
서브쓰레드6
서브쓰레드7
서브쓰레드8
서브쓰레드9
메인쓰레드0
메인쓰레드1
메인쓰레드2
메인쓰레드3
메인쓰레드4

```

run()

Main()

그냥 main쓰레드에서 run() 메서드를 호출한것이다.

start() 메서드

OS가 스레드 스케줄러를 통해 언제 실행 할지를 결정한다.

```
t1.start();  
t2.start();  
  
t1.start();
```

→ 이미 사용한 스레드를 재사용 불가능하다.

```
Problems Javadoc Declaration Search Console × Git Staging History Debug  
<terminated> Main (1) [Java Application] C:\Users\zest1\p2\pool\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64.jre\bin\java.exe  
main  
Thread 를 상속받은 스레드  
MyThread  
인터페이스 구현한 스레드  
MyThread2  
Exception in thread "main" java.lang.IllegalThreadStateException  
    at java.base/java.lang.Thread.start(Thread.java:793)  
    at joo.강의14.Main.main(Main.java:26)
```



```
t1.start();  
t2.start();  
  
t1 = new MyThread();  
t1.start();
```

서브쓰레드의 예외발생

```
public static void main(String[] args) {  
    // TODO Auto-generated method stub  
  
    Thread th = new Thread(new MyThread2());  
    th.start();  
  
    for(int i=0;i<100;i++)  
        System.out.println(i);  
}
```

```
public class MyThread2 implements Runnable{  
    @Override  
    public void run() {  
        try  
        {  
            throw new Exception();  
        } catch (Exception ex)  
        {  
            ex.printStackTrace();  
        }  
    }  
}
```

```
42  
43  
44 java.lang.Exception  
45  
46  
47  
48  
49  
50  
51  
52  
53  
54  
55  
56  
57  
58  
59  
60  
61  
62  
63  
64  
65  
66  
at joo.강의14.MyThread2.run(MyThread2.java:13)  
at java.base/java.lang.Thread.run(Thread.java:833)
```

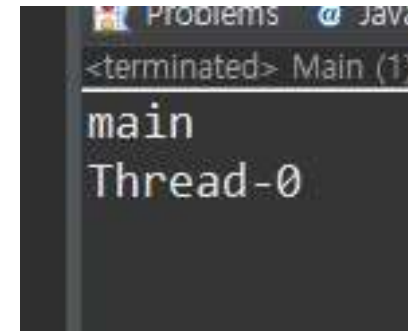
호출스택에 Main 메서드가
없다. 즉 별도의 호출스택을
가지는 것이다.

서브쓰레드의 예외는 메인쓰레드에 영향이 없다.

쓰레드의 이름

```
public static void main(String[] args) {  
    // TODO Auto-generated method stub  
  
    //currentThread()현재 해당 코드를 수행중인 쓰레드를 가지고 온다.  
    System.out.println(Thread.currentThread().getName());  
  
    Thread th = new Thread(new MyThread2()); //생성자로 러너블을  
    th.start();  
  
}
```

```
public class MyThread2 implements Runnable{  
  
    @Override  
    public void run() {  
        System.out.println(Thread.currentThread().getName());  
    }  
}
```



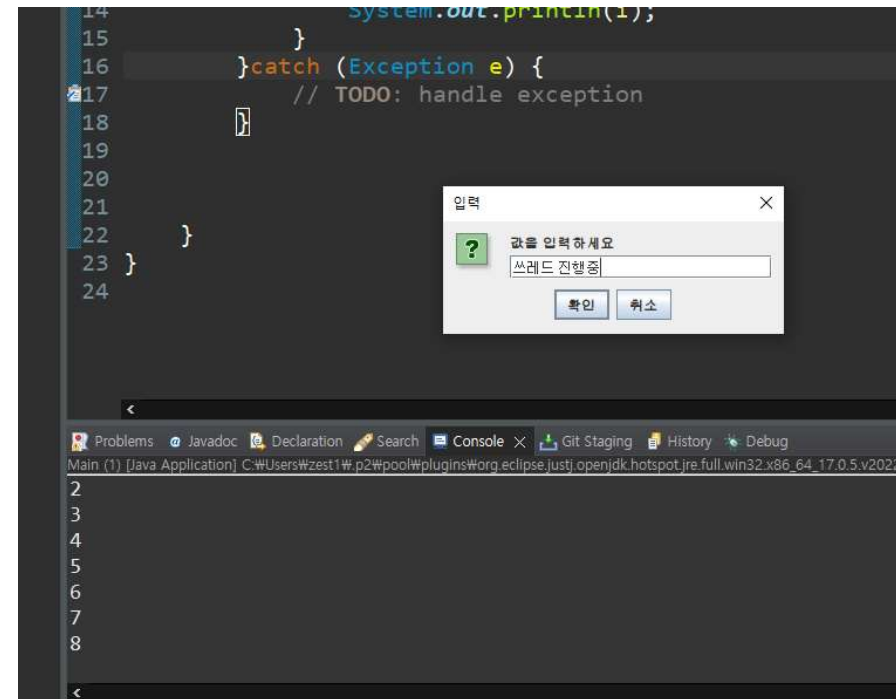
쓰레드의 활용

```
@Override
public void run() {

    try
    {
        for(int i =0 ;i<10;i++)
        {
            Thread.sleep(1000);
            System.out.println(i);
        }
    }catch (Exception e) {
        // TODO: handle exception
    }

}
```

```
Thread th = new Thread(new MyThread2()); //생성자
th.start();
String input = JOptionPane.showInputDialog("값을 입력하세요");
```

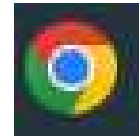
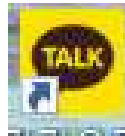


다이얼로그 UI를 띄우고 움직이고 등등의 일을 하는 동안 서브쓰레드는 카운트를 세고 있다

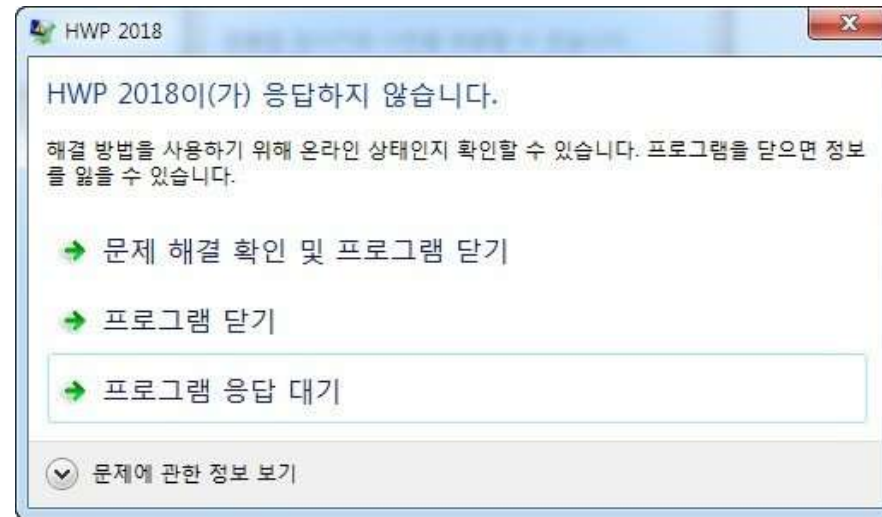
운영체제

시분할 방식으로 CPU가 각각의 프로그램에게 번갈아 가면서 방문하여 소스코드를 수행해준다.

```
public static void main(String  
    // TODO Auto-generated met  
  
    Thread th = new Thread(new  
    th.start();  
    String input = JOptionPane
```



CPU가 메인쓰레드를 우리 프로그램에 할당해줬는데
중간에 끊을수 없는 작업으로 계속 잡고 있다면?



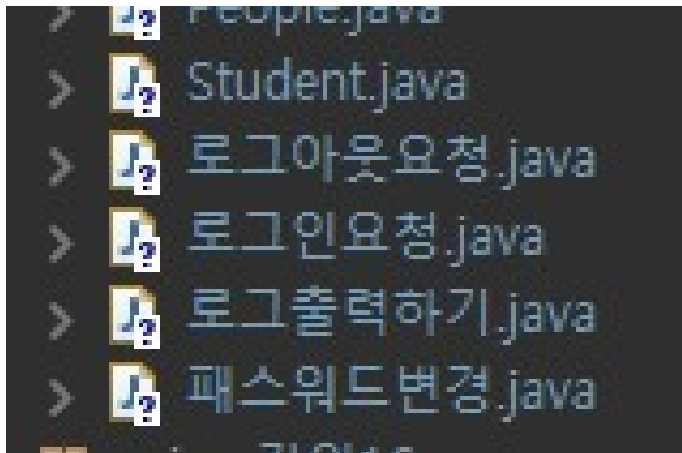
운영체제에서 응답없음으로 간주하고 제한시간 지나면 가버린다.

쓰레드의 활용

쓰레드가 없다면 서버와의 통신, 프린터, 로그 남기기 등등 백그라운드에서 처리해야 하는 일을 수행할때 마다 UI가 멈춘다.

쓰레드의 활용

쓰레드를 쓰고싶을때 마다 Runnable
인터페이스를 구현하는 클래스를 생성 해야 하나..?



게다가 이것들은 다 기능들 아닌가..? 이걸 하나의 객체라고 볼수 없는데.. 메서드 단위여야 하는거 아닌가?

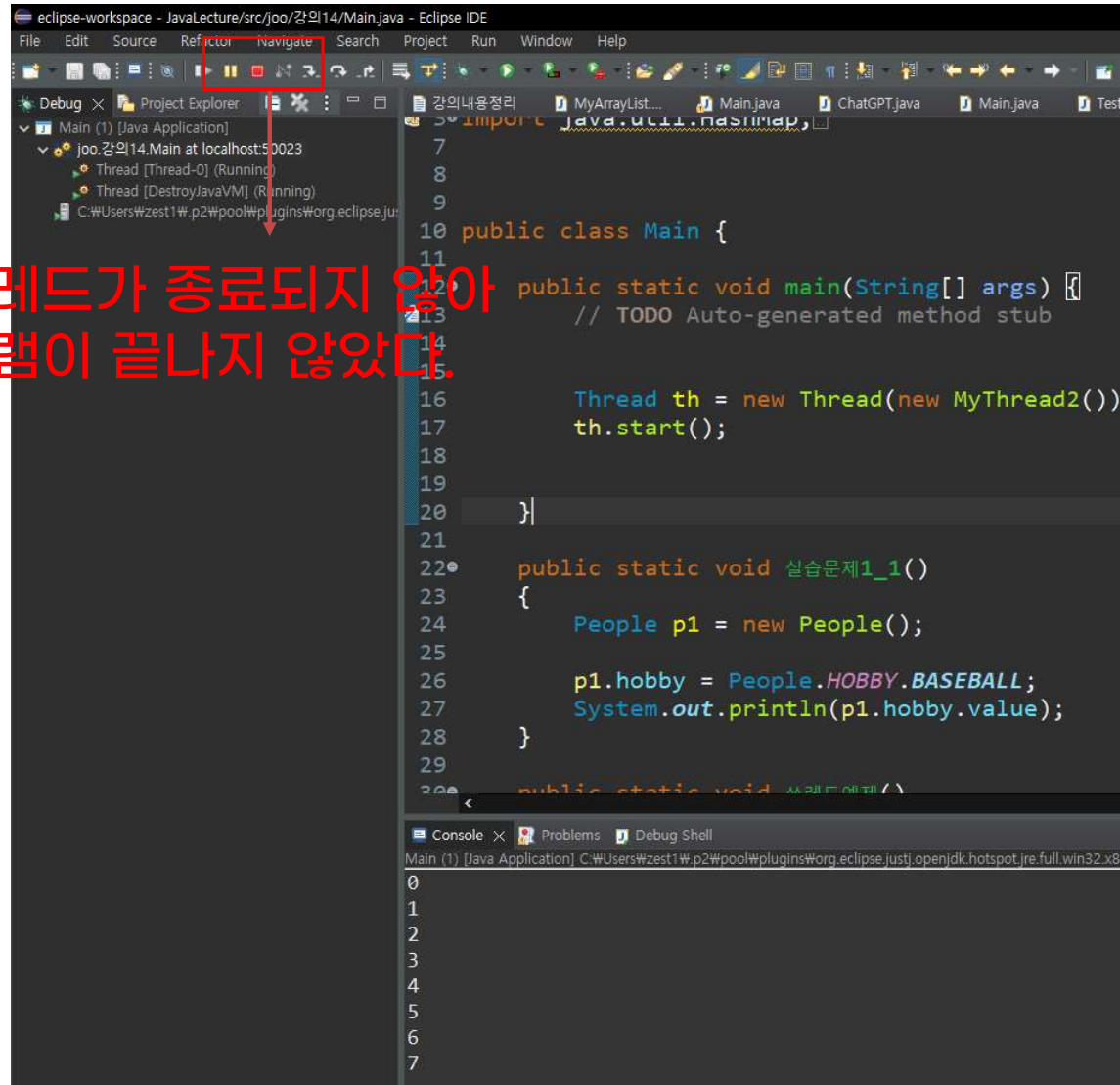
전혀 객체지향적이지 않아!

```
public static void main(String[] args) throws Exception {  
    // TODO Auto-generated method stub  
  
    new Thread(new Runnable(){  
        @Override  
        public void run() {  
            System.out.println("서브스레드에서 실행!");  
        }  
    }).start();  
  
    JOptionPane.showInputDialog("값을 입력하세요");  
}
```

무명클래스를 자주 이용한다.

데몬 쓰레드

서브쓰레드가 종료되지 않아
프로그램이 끝나지 않았다.



```
eclipse-workspace - JavaLecture/src/joo/강의14/Main.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Main (1) [Java Application]
  joo.강의14.Main at localhost:50023
    Thread [Thread-0] (Running)
    Thread [DestroyJavaVM] (Running)
    C:\Users#wzest1#p2#pool#plugins#org.eclipse.jdt.core.jar

import java.util.concurrent.*;

7
8
9
10 public class Main {
11
12     public static void main(String[] args) {
13         // TODO Auto-generated method stub
14
15
16         Thread th = new Thread(new MyThread2());
17         th.start();
18
19
20     }
21
22     public static void 실습문제1_1()
23     {
24         People p1 = new People();
25
26         p1.hobby = People.HOBBY.BASEBALL;
27         System.out.println(p1.hobby.value);
28     }
29
30     public static void 쓰레드예제()
31     {
32
33     }
34 }
```

Console × Problems Debug Shell

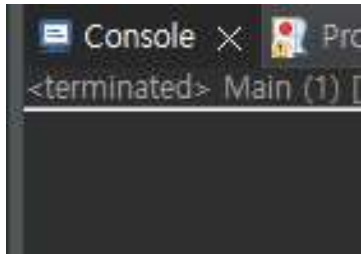
Main (1) [Java Application] C:\Users#wzest1#p2#pool#plugins#org.eclipse.jdt.core.jar

```
0
1
2
3
4
5
6
7
```

서브쓰레드가 무한루프로 계속 돌다가 메인쓰레드가 끝날때
자동으로 같이 끝내고 싶다면?
ex) 문서프로그램 자동저장기능

```
public static void main(String[] args) {  
    // TODO Auto-generated method stub  
  
    Thread th = new Thread(new MyThread2())  
    th.setDaemon(true); //데몬쓰레드로 지정  
    th.start();  
}
```

데몬쓰레드로 지정, start() 이전에 셋팅해줘야 한다



자신을 호출한 부모 쓰레드가 종료되면 자동으로 본인도 종료된다.

임계영역

```
public class Account {  
    static int money;  
  
    public static void withdraw(int money)  
    {  
        //출금하려는 금액보다 많아야 출금을 진행한다.  
        if(Account.money >= money)  
        {  
            try{  
                Thread.sleep(2000);  
            }catch(Exception ex)  
            {  
                //금액이 마이너스가 될수 없다.  
            }  
            Account.money -= money;  
        }  
    }  
}
```

```
public class MyThread2 implements Runnable{  
  
    @Override  
    public void run() {  
        Account.withdraw(7000);  
    }  
}
```

```
//만원 입금  
Account.money=10000;  
  
Thread th = new Thread(new MyThread2());  
Thread th2 = new Thread(new MyThread2());  
//각자 7천원씩 인출한다.  
th.start();  
th2.start();  
  
Thread.sleep(2000);  
  
System.out.println(Account.money);
```

Problems Javadoc Declarations
<terminated> Main (1) [Java Application]
-4000

```
public class Account {  
    static int money;  
  
    public static void withdraw(int money)  
    {  
        //출금하려는 금액보다 많아야 출금을 진행한다.  
        if(Account.money >= money)  
        {  
            try{  
                Thread.sleep(2000);  
            }catch(Exception ex)  
            {  
            }  
            Account.money -= money;  
        }  
    }  
}
```

```
public class MyThread2 implements Runnable{  
    @Override  
    public void run() {  
        Account.withdraw(7000);  
    }  
}
```

```
public class MyThread2 implements Runnable{  
    @Override  
    public void run() {  
        Account.withdraw(7000);  
    }  
}
```

아직 첫번째 스레드가 인출을 하지 않았다.
그러므로 7천원 인출이 가능하다.

스레드끼리 공유해서 쓰는 자원일 경우 한번에 하나의 스레드만 쓰는 것이 안전하다.

withdraw 메서드 전체를 임계영역
설정 (블록으로 특정영역만 설정도
가능하다)

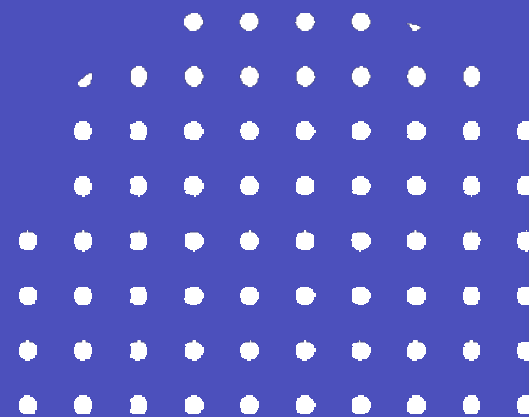
```
public static synchronized void withdraw(int money)
{
    //출금하려는 금액보다 많아야 출금을 진행한다.
    if(Account.money >= money)
    {
        try{
            Thread.sleep(2000);
        }catch(Exception ex)
        {}

        Account.money -= money;
    }
}
```

이 외에도 쓰레드에 관해 상태제어, 동기화,
데드락 등등 많은 내용들이 있다.

— 04


실습문제



2-1 실습문제 (normal)

0~10초까지 카운트를 출력해보자

- 1초에 한번 숫자가 출력되어야 한다.
- 출력하기전 현재의 쓰레드를 1초간 정지 해야 한다.



```
Problems @ Javad
<terminated> Main (6) [
0
1
2
3
4
5
6
7
8
9
10
```

2-1 문제풀이 (normal)

Park Ju Byeong

Park Ju Byeong

2-2 실습문제 (normal)

카운트가 올라가면서 동시에 다이얼로그 창이 띄워지도록 하자

- 다이얼로그가 띄워진 채 숫자가 올라가야 한다.
- 카운트 출력이 서브쓰레드로 실행이 되어야 가능하다.
- Thread 클래스, Runnable 인터페이스 둘 중 편한 것을 사용하자.



2-2 문제풀이 (normal)

Park Ju Byeung

Pu
eong

2-3 실습문제 (normal)

1번 문제의 카운트가 다이얼 로그창을 닫을 때 까지 계속 나오게 하고
다이얼로그 창을 닫으면 프로그램이 종료되도록 하자

2-3 문제풀이 (normal)



THANK YOU



강사 박주병