자바 스터디 발표

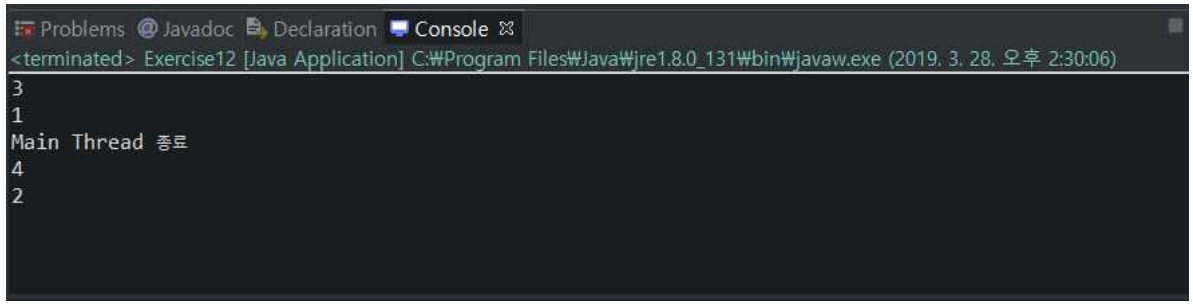
Enjoy your stylish business and campus life with BIZCAM

PPT PRESENTATION

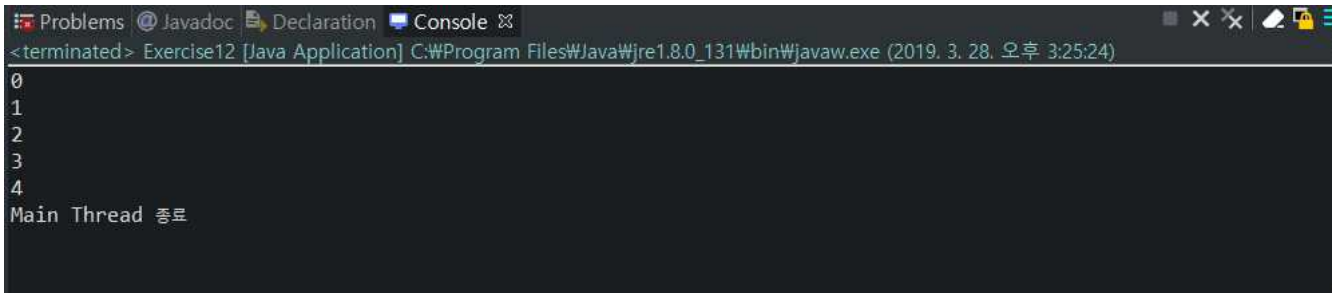
```
class testThread extends Thread {
    int testNum;

    testThread() {
    }

    testThread(int num) {
        testNum = num;
    }

    @Override
    public void run() {
        System.out.println(testNum);
    }
}
```

```
public class Exercise12 {

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

        for(int i=0;i<5;i++) {
            testThread testTrd=new testThread(i);
            testTrd.start();
        }
        System.out.println("Main Thread 종료");
    }

}
```

```
public class Exercise12 {

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

        for(int i=0;i<5;i++) {
            testThread testTrd=new testThread(i);
            testTrd.run();
        }
        System.out.println("Main Thread 종료");
    }

}
```


Problems
Javadoc
Declaration
Console
<terminated> Exercise12 [Java Application] C:₩Program Files₩Java₩jre1.8.0_131₩bin₩javaw.exe (2019. 3. 28. 오후 2:30:06)
3
1
Main Thread 종료
4
2


Main Thread 종료
4
2
1
3


Problems
Javadoc
Declaration
Console
<terminated> Exercise12 [Java Application] C:₩Program Files₩Java₩jre1.8.0_131₩bin₩javaw.exe (2019. 3. 28. 오후 3:25:24)
0
1
2
3
4
Main Thread 종료

Thread.start()	Thread.run()
Causes this thread to begin execution; the Java Virtual Machine calls the run method of this thread. 이 스레드가 실행을 시작하도록 합니다. Java Virtual Machine은 이 스레드의 run 메소드를 호출 합니다.	If this thread was constructed using a separate Runnable run object, then that Runnable object's run method is called; otherwise, this method does nothing and returns. 이 thread가 개별의 Runnable 실행 객체를 사용해 구축되었을 경우, 그 Runnable 객체의 run 메소드가 불려갑니다. 그렇지 않은 경우,이 메소드는 아무것도 실시하지 않고 리턴합니다.

start()할 경우 개인의 thread를 만들고 개인의 call stack을 이용한다.

run() method로 쓰레드를 사용할 경우 main thread의 call stack을 이용한다.

Thread State

```
public static enum Thread.State
extends Enum<Thread.State>
```

A thread state. A thread can be in one of the following states:

- `NEW`
  A thread that has not yet started is in this state.
- `RUNNABLE`
  A thread executing in the Java virtual machine is in this state.
- `BLOCKED`
  A thread that is blocked waiting for a monitor lock is in this state.
- `WAITING`
  A thread that is waiting indefinitely for another thread to perform a particular action is in this state.
- `TIMED_WAITING`
  A thread that is waiting for another thread to perform an action for up to a specified waiting time is in this state.
- `TERMINATED`
  A thread that has exited is in this state.

A thread can be in only one state at a given point in time. These states are virtual machine states which do not reflect any operating system thread states.

PPT PRESENTATION

LoginExample.ja | Exercise12.java | StatePrintThrea | TargetThread.ja | ThreadStateExam

```java
package example;

public class ThreadStateExam {
    public static void main(String[] args) {
        StatePrintThread thread = new StatePrintThread(new TargetThread());
        thread.start();
    }

}
```

LoginExample.ja | Exercise12.java | StatePrintThrea | TargetT

```java
package example;

public class TargetThread extends Thread {
    @Override
    public void run() {
        for (long i = 0; i < 2000000000; i++) {

        }

        try {
            Thread.sleep(1500);
        } catch (Exception e) {}

        for (long i = 0; i < 2000000000; i++) {

        }
    }
}
```

LoginExample.ja | Exercise12.java | StatePrintThrea | TargetThread.ja | ThreadStateExam

```java
package example;

public class StatePrintThread extends Thread {
    private Thread targetThread;

    public StatePrintThread(Thread targetThread) {
        this.targetThread = targetThread;
    }

    @Override
    public void run() {
        while (true) {
            Thread.State state = targetThread.getState();
            System.out.println("타겟 스레드 상태: " + state);

            if (state == Thread.State.NEW) {
                targetThread.start();
            }

            if (state == Thread.State.TERMINATED) {
                break;
            }

            try {
                Thread.sleep(500);
            } catch (Exception e) {
            }
        }
    }
}
```

```
타겟 스레드 상태: NEW
타겟 스레드 상태: RUNNABLE
타겟 스레드 상태: RUNNABLE
타겟 스레드 상태: RUNNABLE
타겟 스레드 상태: TIMED_WAITING
타겟 스레드 상태: TIMED_WAITING
타겟 스레드 상태: TIMED_WAITING
타겟 스레드 상태: RUNNABLE
타겟 스레드 상태: RUNNABLE
타겟 스레드 상태: RUNNABLE
타겟 스레드 상태: TERMINATED
```

synchronizedCollection

```
public static <T> Collection<T> synchronizedCollection(Collection<T> c)
```

Returns a synchronized (thread-safe) collection backed by the specified collection. In order to guarantee serial access, it is critical that **all** access to the backing collection is accomplished through the returned collection.

It is imperative that the user manually synchronize on the returned collection when traversing it via `Iterator`, `Spliterator` or `Stream`:

```
Collection c = Collections.synchronizedCollection(myCollection);
   ...
synchronized (c) {
    Iterator i = c.iterator(); // Must be in the synchronized block
    while (i.hasNext())
       foo(i.next());
}
```

Failure to follow this advice may result in non-deterministic behavior.

The returned collection does *not* pass the `hashCode` and `equals` operations through to the backing collection, but relies on `Object`'s equals and hashCode methods. This is necessary to preserve the contracts of these operations in the case that the backing collection is a set or a list.

The returned collection will be serializable if the specified collection is serializable.

지정된 컬렉션을 기본으로하는 동기(thread에 대해서 안전한) 컬렉션을 리턴합니다. 직렬 액세스를 보증하려면, 배킹 콜렉션에의 모든 액세스가, 돌려 주어진 콜렉션을 개입시켜 행해지는 것이 중요합니다.


Thread Pool

Main 스레드에서 블로킹이 될 수 있는 메소드가 있다.
-> 다른 스레드를 만들어서 메인Thread에서 호출
-> 병렬 처리를 위해 Thread가 무수히 많아지면? 성능의 저하가 일어난다.
-> ThreadPool 사용


ThreadPool이란?
Thread Pool은 작업 처리에 사용되는 Thread를 제한된 개수만큼 정해 놓고 작업 큐(Queue)에 들어오는 작업들을 하나씩 Thread가 맡아 처리합니다. 작업 처리가 끝난 스레드는 다시 작업 큐에서 새로운 작업을 가져와 처리해야 합니다.