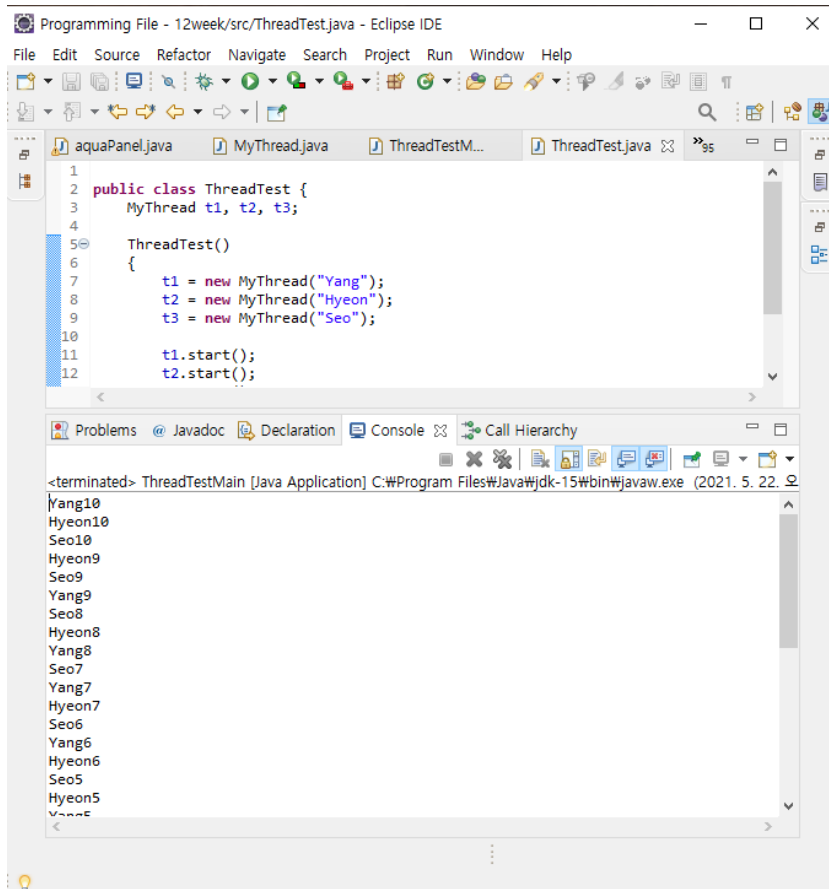


객체지향프로그래밍 12주차 정리

프로그래밍

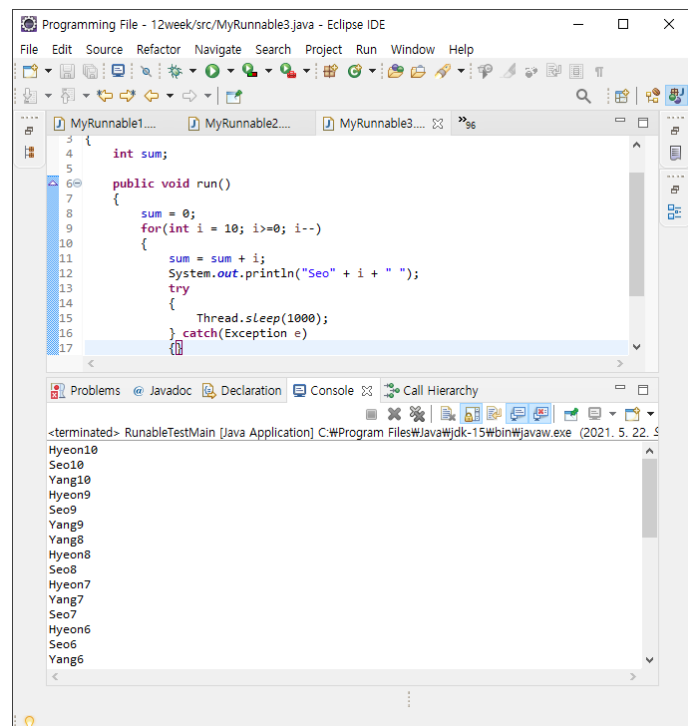


The screenshot shows the Eclipse IDE with the file `ThreadTest.java` open. The code defines a `ThreadTest` class with a constructor that creates three `MyThread` objects named `t1`, `t2`, and `t3`, each with a name ("Yang", "Hyeon", "Seo"). The constructor also calls `t1.start()` and `t2.start()`. The console output shows the execution of the program, displaying the names of the threads in a specific order: Yang10, Hyeon10, Seo10, Hyeon9, Seo9, Yang9, Seo8, Hyeon8, Yang8, Seo7, Yang7, Hyeon7, Seo6, Yang6, Hyeon6, Seo5, Hyeon5, Yang5.

```
1 public class ThreadTest {
2     MyThread t1, t2, t3;
3
4
5     ThreadTest()
6     {
7         t1 = new MyThread("Yang");
8         t2 = new MyThread("Hyeon");
9         t3 = new MyThread("Seo");
10
11         t1.start();
12         t2.start();
13     }
14 }
```

Console Output:

```
<terminated> ThreadTestMain [Java Application] C:\Program Files\Java\jdk-15\bin\javaw.exe (2021. 5. 22. 5. 22. 5)
Yang10
Hyeon10
Seo10
Hyeon9
Seo9
Yang9
Seo8
Hyeon8
Yang8
Seo7
Yang7
Hyeon7
Seo6
Yang6
Hyeon6
Seo5
Hyeon5
Yang5
```

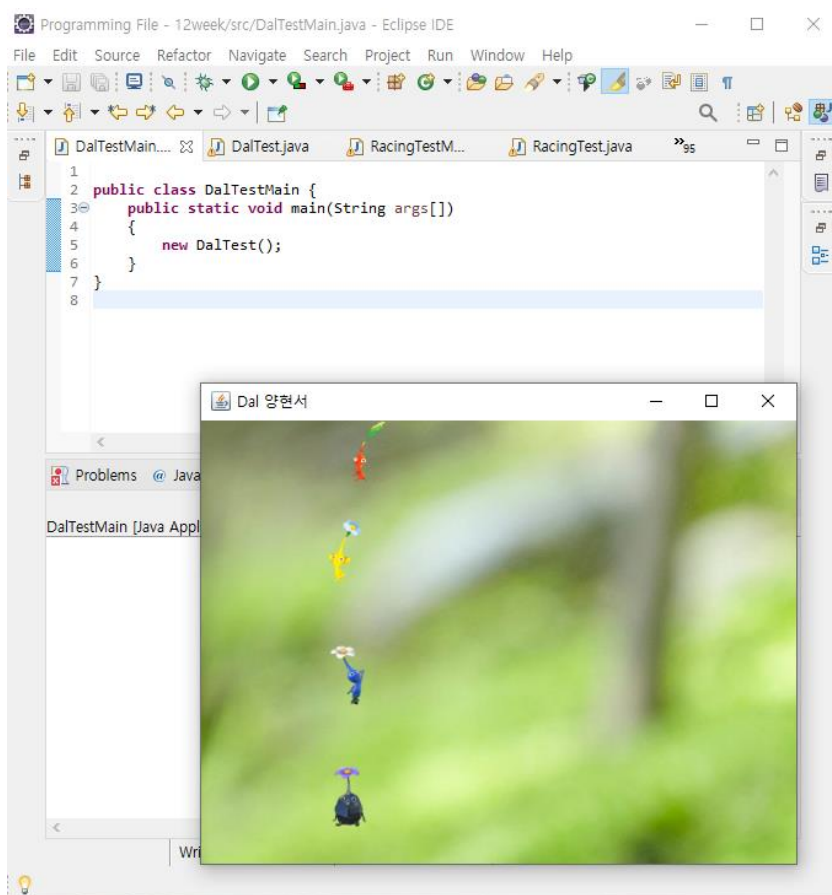
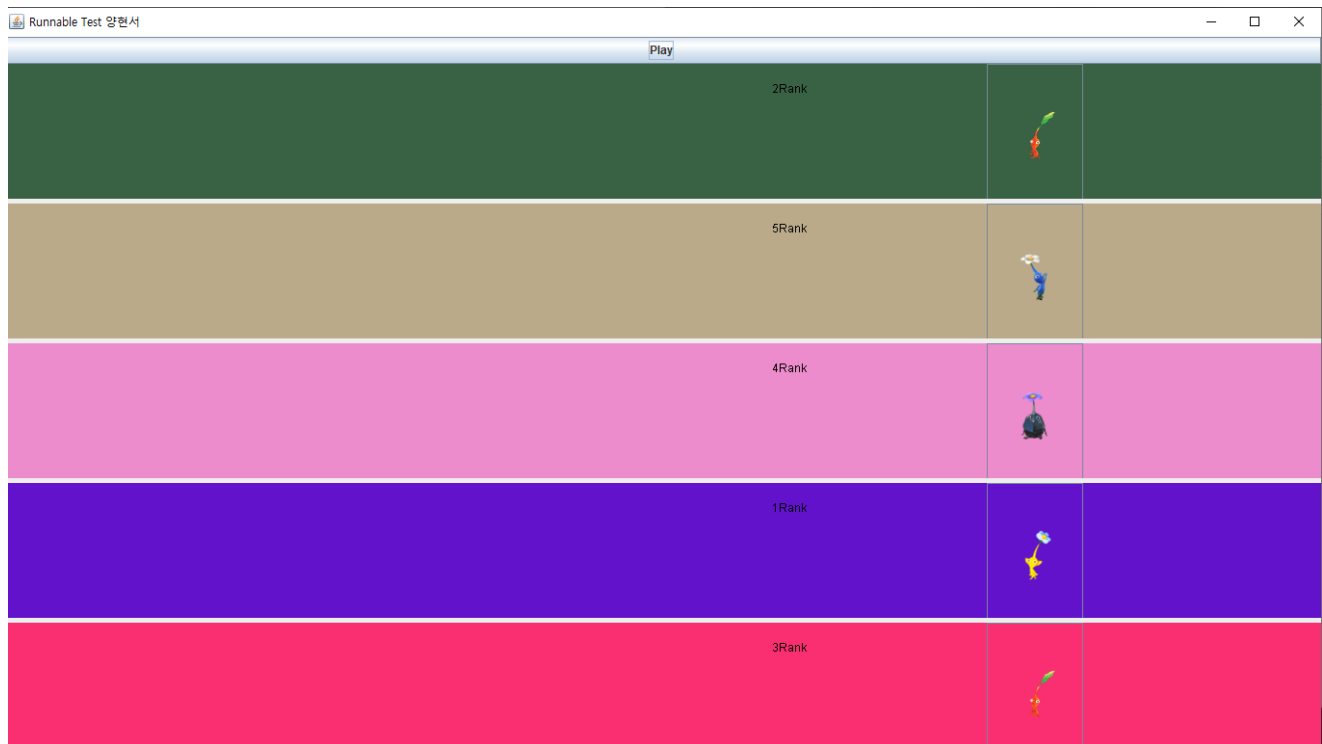


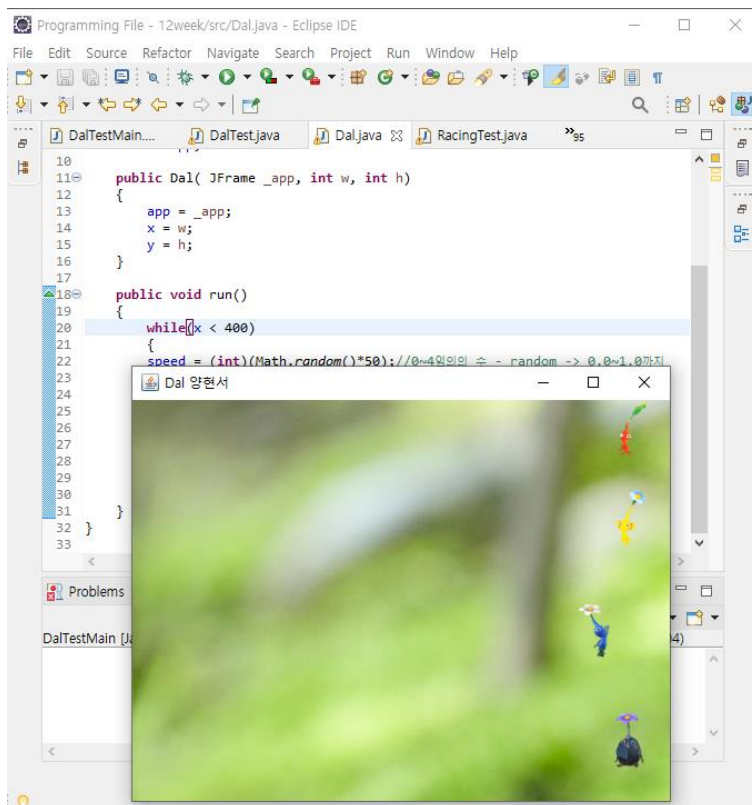
The screenshot shows the Eclipse IDE with the file `MyRunnable3.java` open. The code defines a `MyRunnable3` class with a `run()` method. The `run()` method initializes a `sum` variable to 0 and enters a `for` loop that iterates from 10 down to 0. In each iteration, it calculates the sum, prints the name "Seo" followed by the current value of `i`, and sleeps for 1000 milliseconds. The console output shows the execution of the program, displaying the names of the threads in a specific order: Hyeon10, Seo10, Yang10, Hyeon9, Seo9, Yang9, Seo8, Hyeon8, Yang8, Seo7, Yang7, Hyeon7, Seo6, Yang6.

```
1 int sum;
2
3 public void run()
4 {
5     sum = 0;
6     for(int i = 10; i>=0; i--)
7     {
8         sum = sum + i;
9         System.out.println("Seo" + i + " ");
10        try
11        {
12            Thread.sleep(1000);
13        } catch(Exception e) {}
14    }
15 }
```

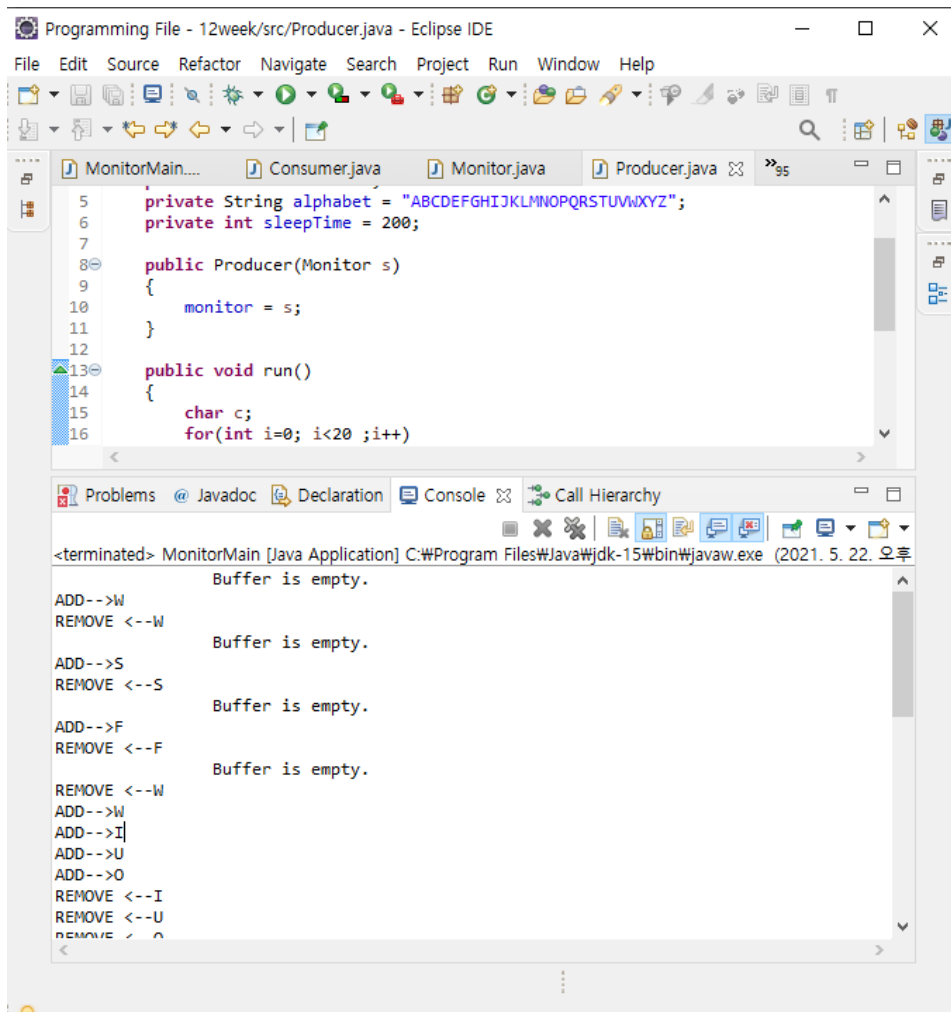
Console Output:

```
<terminated> RunnableTestMain [Java Application] C:\Program Files\Java\jdk-15\bin\javaw.exe (2021. 5. 22. 5. 22. 5)
Hyeon10
Seo10
Yang10
Hyeon9
Seo9
Yang9
Seo8
Hyeon8
Yang8
Seo7
Yang7
Hyeon7
Seo6
Yang6
```





Runnable Test 양현서			
Play	Stop	Suspend	Resume
1Rank			
4Rank			
5Rank			
2Rank			
3Rank			



쓰레드

프로그램 실행 유닛

인스턴스(실체)

프로그램 돌릴 때 Main Memory 에 만들어진다

Ex) 하나의 프로그램(=소프트웨어의 유닛)

프로세스(하나의 프로그램 유닛을 구성하는 여러 작은 실행 유닛 = Task)

여러 개-> multiprocessing

C 언어, 일반적인 프로그래밍 언어

객체(클래스 객체)

JAVA, 객체지향프로그래밍 언어

Heavy-weight Process : 실행코드가 모든 프로세스마다 독립적으로 존재

쓰레드

클래스 또는 프로세스 내의 실행 유닛

Light-weight Process : 코드 세그먼트와 데이터 세그먼트 공유 -> 빠르다

⇒ 하나의 프로그램 유닛 안에는 작은 실행 유닛들을 만들 수 있다

Batch-Programming(하나의 프로그램만 들어간다) -> Multi-Programming(여러 개의 프로그램이 가동 가능) -> Multi-Processing(하나의 프로그래밍 실행 코드를 프로세스들로 쪼갤 수 있음, 동시에 돌아간다) -> Multi-Thread(한 프로세스 내에서도 쪼갬)(개발자의 영역)

자바 실행 유닛

자바 응용프로그램 ⊃ 프로세스(클래스 오브젝트 인스턴스) ⊃ 쓰레드(쓰레드 클래스 오브젝트 인스턴스)

Concurrent Processing Concept(동시에 여러 개 시행)

Concurrent processing

멀티 프로세서들이 더 나은 수행을 위해 동시에 명령을 수행하는 컴퓨팅 모델

(출처 : <https://searchoracle.techtarget.com/definition/concurrent-processing>)

프로그램의 효율성, 논리성

■ 동기화

n 개의 실행 유닛이 실행 시점을 맞추고 싶을 때(특정 시점에 종료)

프로세스간의 지연 큐

Semaphore's Lock / Unlock

■ Mutual Exclusive

두 유닛의 반대되는 조절

Ex) input / output, read/write

Monitor

➔ 동기화 : 두개의 유닛이 한 공간에 동시에 돌아가지 못한다

➤ Input / output, read/write

■ Communication

커뮤니케이션 지원

Ex)Mailbox, Socket

위의 세가지 기능이 필요할 때 스레드 이용

인스턴스를 만들 때, context switch(CPU 할당), 멀티 프로세스 컨트롤에서의 오버헤드 줄이는 역할

⇒ 공간을 공유하기 때문

⇒ 프로그램의 논리를 빠르고 쉽게 개발할 수 있도록

Context switch

멀티 프로세스 환경에서 CPU 가 하나의 프로세스를 실행하고 있는 상태에서 인터럽트 요청에 의해 다른 우선순위의 프로세스가 실행되어야 할 때 기존의 프로세스의 상태 또는 레지스터 값을 저장하고 CPU 가 다음 프로세스를 수행하도록 새로운 프로세스의 상태 또는 레지스터 값(Context)를 교체하는 작업

Context : CPU 가 해당 프로세스를 실행하기 위한 해당 프로세스의 정보들

→ 프로세스의 PCB(Process Control Block)에 저장

(프로세스의 상태(생성, 준비, 수행, 대기, 중지), 프로그램 카운터(프로세스가 다음에 실행할 명령어 주소), 레지스터(누산기, 스택, 색인 레지스터), 프로세스 번호)

→ Context Switching 이 일어나는 인터럽트 요청

I/O request

Time slice expired(CPU 사용시간 만료)

Fork a child(자식프로세스를 만들 때)

wait for an interrupt(인터럽트 처리를 기다릴 때)

등

(출처 : <https://jeong-pro.tistory.com/93>)

쓰레드 유닛들은 코드 세그먼트와 데이터 세그먼트(힙영역)을 공유 => 빨라진다

Context switch 는 Code Segment 와 Data Segment(Heap 영역)의 값을 바꿔주기 위함

(프로세스 유닛 간)

프로그램이 돌아갈 때 -> 프로그램 카운트 => Code Segment 지칭

레이스, 워킹 메모리 => Data Segment

Code Segment 와 Data Segment 가 같은 값 => 바꿔줄 필요가 없다(쓰레드 간)

⇒ Stack Segment 만 지정해주면 된다(쓰레드마다 가지게 되는 워킹 메모리 값을 지정)

하나의 클래스 내에서 여러 개의 쓰레드를 만들 수 있다

⇒ 쓰레드의 개수만큼 CPU 가 존재한다면 문제가 없지만, 쓰레드의 수가 CPU 의 수보다 많은 경우 쓰레드를 관리해주어야 한다

스케줄링 유닛

자바 쓰레드

쓰레드 클래스를 통해 쓰레드를 만들 수 있고 **Runnable instance** 를 통해 쓰레드를 만들 수 있다

쓰레드 클래스 <- run()함수 고정됨

run()함수가 고정된 경우, 같은 일을 하는 경우에는 쓰레드 클래스가 좋다

ex) 게임에서 같은 일을 하는 유닛들을 만들 때

Runnable instance <-run()함수 재구현

매번 수행되는 쓰레드가 다른 기능을 하길 바란다면 Runnable instance 가 좋다

Ex) 게임에서 다른 기능을 하는 유닛을 만들 때

***용도가 다르다**

java.lang.Thread 를 import 해야 한다

Java Thread 를 위해 알아야 하는 메소드

void start()

void run()

void sleep() : 일정 시간동안 멈췄다가 run()으로 이동하여 다시 실행

int getPriority()

void suspend() : start()상태로 멈춘다 => 다른 스레드가 도착할 때까지 기다리는 것

스레드 인스턴스의 Life Cycle

new -> 스레드 인스턴스 생성 -(start())->실행(ready 상태)

-(Java Scheduler 에 의해 run())-> 실행(run) -((stop()))-> 종료

스레드 클래스 상속 => 스레드 정의

new ClassObject 이용해 생성

start()

run() : 스레드 클래스 내에 정의

객체를 생성하면 바로 스레드가 만들어진다

Runnable interface 를 implements => 스레드 클래스가 Runnable 을 받아서 구현

new ClassObject 이용해 생성

start()

run() : 스레드 클래스 내에 정의 / 직접 구현해야 한다

매번 오버라이딩

해당 클래스는 스레드가 아니기 때문에 객체생성 후 스레드로 다시 정의해야 한다

Thread rt = new Thread(ro); // 스레드로 선언

⇒ Runnable 스레드마다 run()을 바꿀 수 있다

*스레드 생성자

Thread()

Thread(Runnable)

등

* run 함수를 어떻게 구현하는지(가능하면 concurrent processing 이 있어야 좋다)

쓰레드 종료 법

1. run()메소드가 끝나면 자동으로 종료하는 방법(자체적인 종료)

제일 좋은 방법

2. 무한 반복에서 쓰레드를 종료하는 방법(다른 쓰레드의 도움)

1) 반복문의 조건문 이용

```
While(stop == false){ }
```

2) 반복문 내에 if 조건문 이용

```
While(true){  
  
    if(Thread.isInterrupted) break; }
```

3. InterruptedException 사용(수행 중에 Interrupt 발생, 자체적으로도 가능)

Timer 기능을 활용하여 InterruptedException 유발(thread.interrupt())

후에 isInterrupted()또는 interrupted()로 확인, 종료

4. stop()

잘 쓰지 않는다(오류의 가능성이 존재하기 때문)