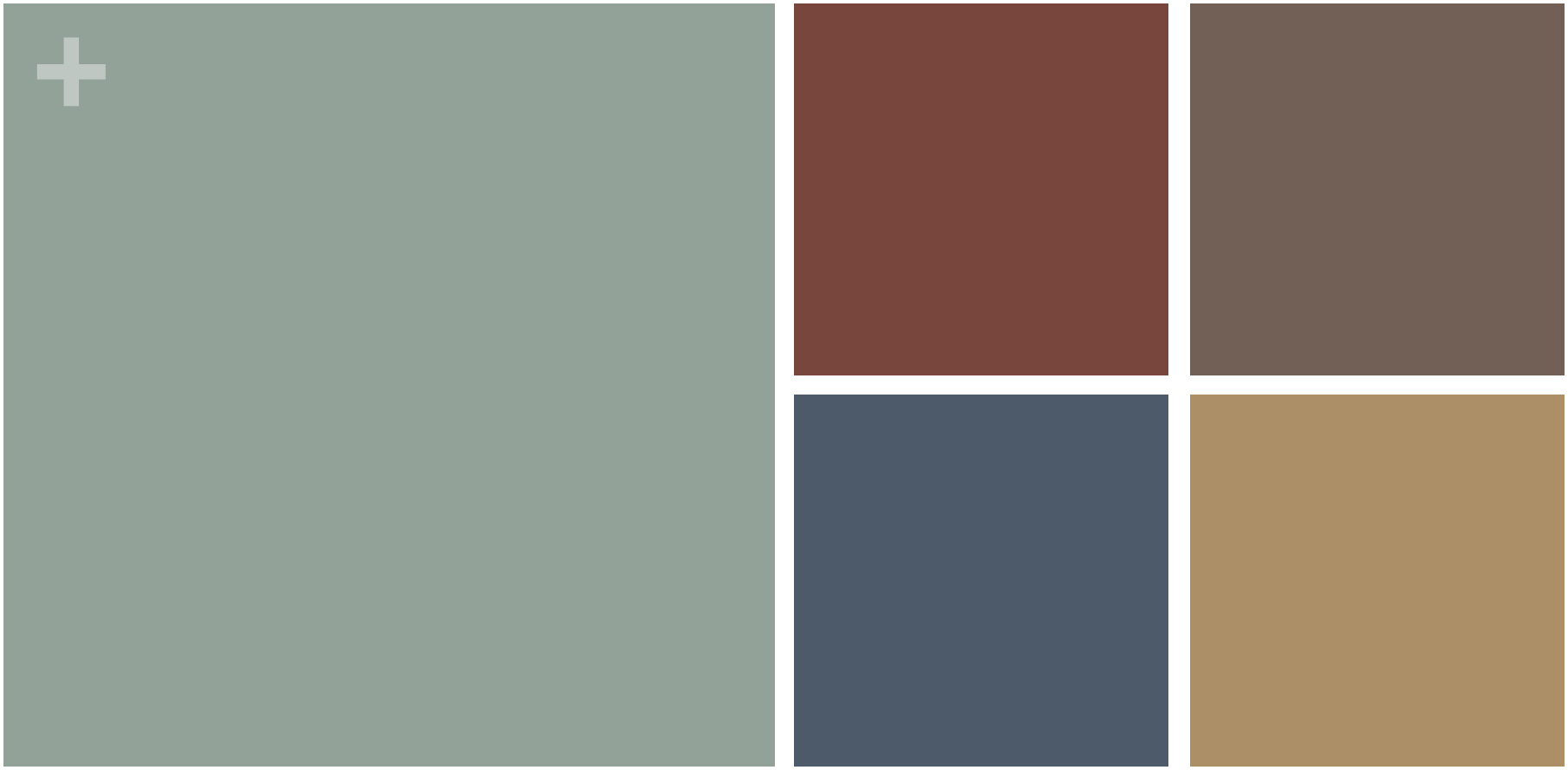


Chap. 23
스레드 (Thread)

1. Simple Thread Programming

- 1.1 다중스레드 프로그램 작성
  - 스레드 클래스를 수퍼클래스로 갖는 클래스(MyThread)를 작성하라.
  - MyThread에서 실행되는 스레드는 스레드 이름과 실행 회수 (실행시마다 순차적으로 증가하는 정수 값)를 출력한다.(실행 회수는 100까지만 출력한다)
  - 실행 예

    [Thread-1] 1
    [Thread-1] 2
    [Thread-2] 1
    [Thread-1] 3
    [Thread-2] 2

```
Class MyThread extends Thread {
    …
    public void run() {
    …
    }
}
```

1. Simple Thread Programming

- 테스트 코드

```
public class test {

    private static MyThread t1 = new MyThread("Thread-1");
    private static MyThread t2 = new MyThread("Thread-2");

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

        t1.start();
        t2.start();
    }

}
```

1. Simple Thread Programming

- 1.2 1.1의 프로그램을 인터페이스 **Runnable**을 구현하는 **MyThread_Runnable** 클래스로 작성하려 동일한 결과를 보여라.

```
Class MyThread_Runnable implements Runnable {
    …
    public void run() {
    …
    }
}
```

1. Simple Thread Programming

- 테스트 코드

```
public class test {

    private static Thread t1 = new Thread(new MyThread_Runnable("Thread-1"));
    private static Thread t2 = new Thread(new MyThread_Runnable("Thread-2"));

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

        t1.start();
        t2.start();
    }

}
```

2. Get Current Time

- 2.1 1에서 작성한 프로그램의 실행 시간을 보여라.
  - Multi-thread (Thread를 상속하고, Runnable 인터페이스 구현한 각 클래스)
  - [참고] 현재 시각을 읽는 프로그램 코드
    - System.nanoTime

3. BusySleep

- 1.3 시간의 경과를 확인하여 일정시간을 대기하는 BusySleep을 구현하라
  - Sleep요청할 시간을 전달받는 생성자를 구현
  - Thread 또는 Runnable을 사용하여 run을 구현
  - BusySleep이란
    - 반복문을 사용하여 시간의 경과를 계속 확인한다.
    - 이때 경과 시간이 Sleep을 요청한 시간을 넘긴 경우 반복문을 멈춘다.
    - 즉, 프로그램은 Sleep 요청 시간만큼 반복문 만을 수행한다.

3. BusySleep

- 테스트 코드
  - 입력되는 대기 시간을 바꿔가며 테스트
  - Join() – 쓰레드의 작업 종료/완료를 기다리는 메소드

```
public class test {

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

        BusySleepThread t = new BusySleepThread(1000);  //ms 단위


        try {
            t.run();
            t.join();
            System.out.println("대기 완료");
        } catch (InterruptedException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }

}
```

```
<terminated> test (4) [Java Application] C:₩
대기 완료
```

[과제-1] Multi-threading 성능

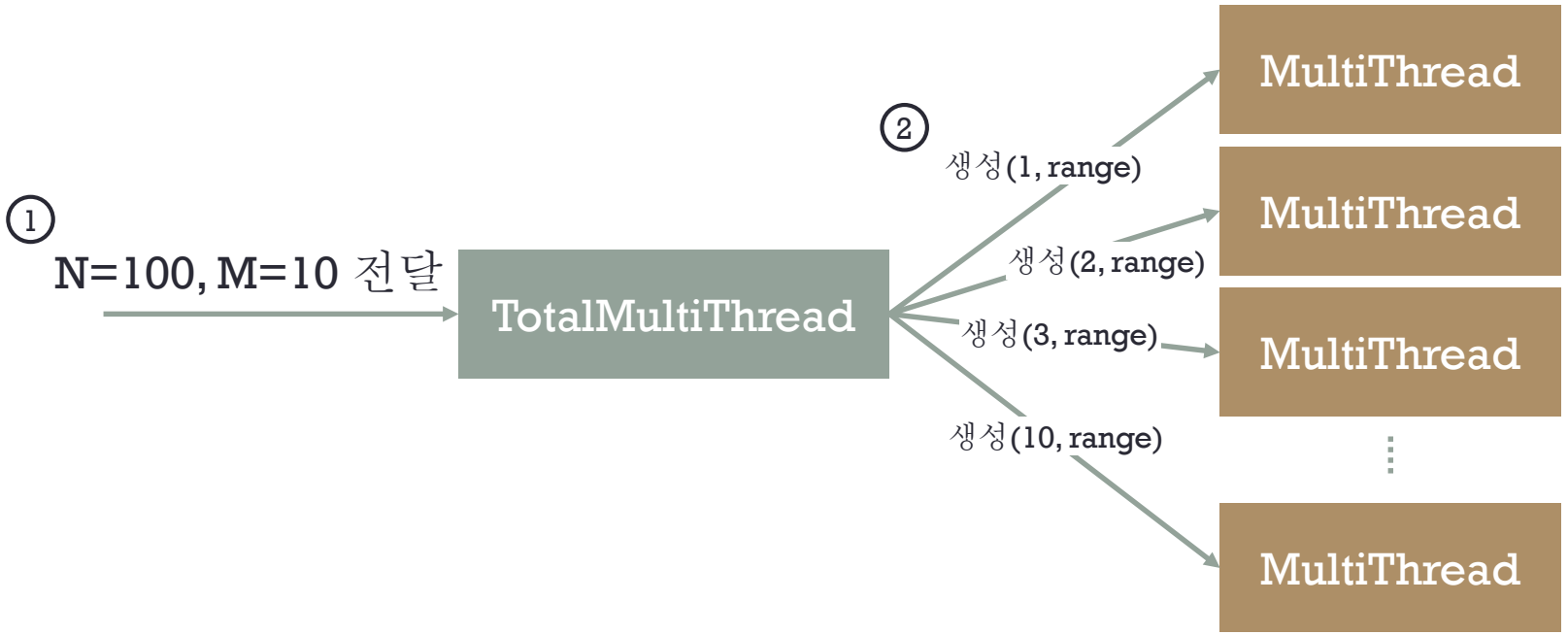
- 1) 1부터 N(충분히 큰 값)까지의 합을 구하는 프로그램을 작성하라.

$$\sum_{k=1}^{n} k = \sum_{i=0}^{m-1} \sum_{j=1}^{n/m} (i * n/m) + j$$

  - 1.1) 각 프로그램에서 N과 M(멀티스레드 개수)의 값에 따른 실행 시간을 보여라.
  - 1.2) 실행 시간을 그래프로 보여라. (N과 M에 따른 실행 시간 t에 대한 3차원 그래프)

[과제-1] Multi-threading 성능

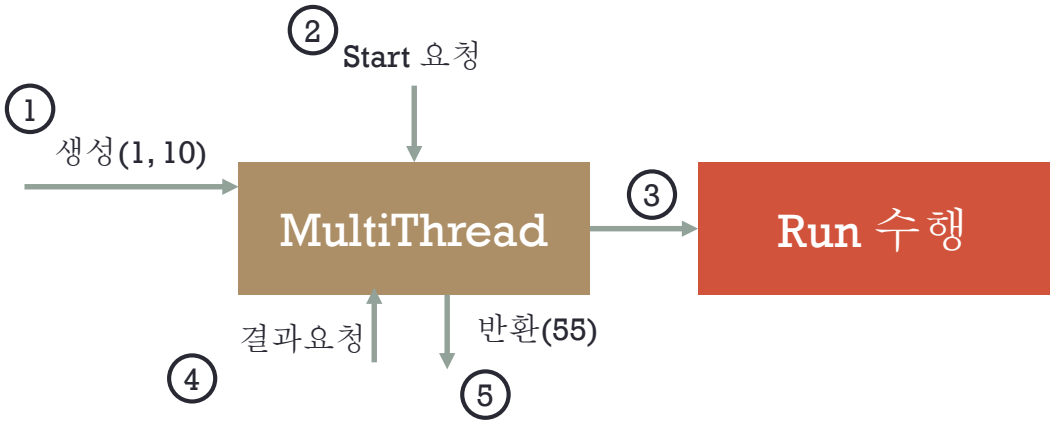
- 2)동작 과정(N=100,M=10)

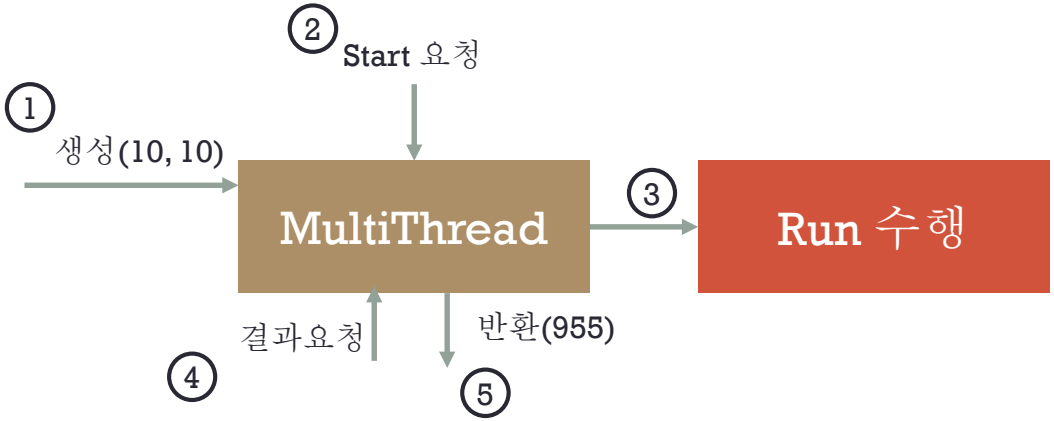


RANGE = N/M

[과제-1] Multi-threading 성능

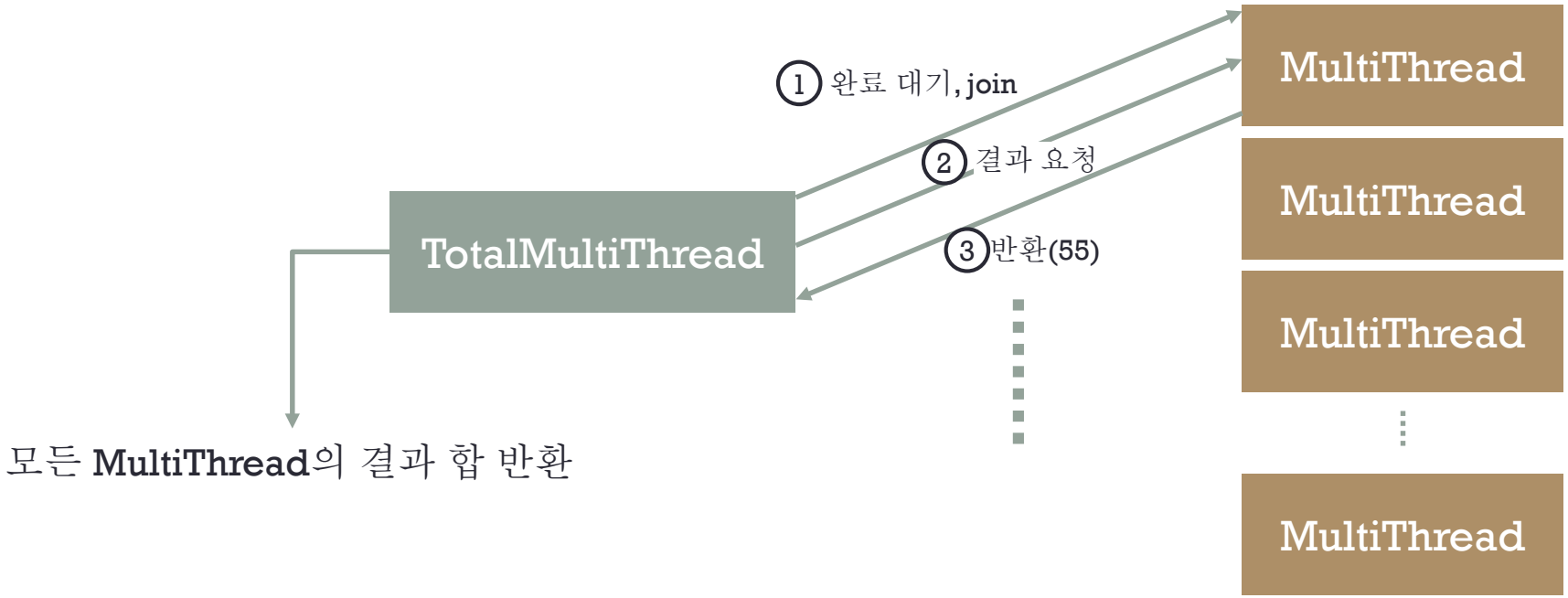
- 2)동작 과정(N=100,M=10)





[과제-1] Multi-threading 성능

- 2)동작 과정(N=100,M=10)



[과제-1] Multi-threading 성능

- 3)클래스 설명
  - **MultiThread** 클래스
    - 생성자에서 **i**값과 **n/m**값(**range**값)을 전달받아 초기화
    - **Thread**를 상속받고 **run**메소드를 오버라이드하여 다음 식을 연산하도록 반복문을 이용해 메소드 구현(연산 결과는 별도의 필드에 저장)

$$\sum_{j=1}^{n/m} (i * n/m) + j$$

    - 별도의 필드에 저장한 결과 값을 반환하는 메소드 구현

[과제-1] Multi-threading 성능

- 3)클래스 설명
  - TotalMultiThread 클래스
    - 1~N까지의 합을 구하기 위한 N과, 이 연산을 나누어 수행할 Thread의 개수를 전달받는 정적 메소드 구현

```
static public long sum(int num, int numOfThread)
```

    - 위 메소드에서 전달받은 Thread 개수만큼 MultiThread를 생성하여 1~N까지의 합을 구하는 작업을 분할하여 작업을 요청
    - 요청 작업을 대기하고(Thread의 join 사용), 작업 완료된 MultiThread에게 결과값을 받아와 모든 결과값을 합하여 반환

[과제-1] Multi-threading 성능

- 4) 주의 사항
  - 단, **N**이 증가할 때 실행 시간도 증가하는 (**or** 변화를 보이는) 충분히 큰 **N**의 값으로 실행해야 함으로 **int**가 아닌 **long**을 사용
  - **M**은 **N**의 약수 만을 입력 받는다고 가정
  - **M**과 **N**에 따른 성능 분석하여 보고서 작성
  - 결과값이 실제 합과 다르게 나오는 경우 감점

[과제-1] Multi-threading 성능

- 5) 테스트코드 & 결과화면

```
public class test {

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


        System.out.println("1~100까지의 합 : " + TotalMultiThread.sum(100, 10));

        System.out.println("<<1~N까지의 합 M개의 쓰레드로 계산 시간(ns)>>");
        for(int i=16; i<=134217728; i=i*2)
        {
            for(int j=1; j<=16; j=j*2)
            {
                start = System.nanoTime();
                TotalMultiThread.sum(i, j);
                System.out.printf("%8d\t", System.nanoTime()-start);
            }
            System.out.println();
        }
    }

}
```

```
<terminated> test (3) [Java Application] C:₩Program Files₩Java₩jre1.8.0_101₩bin₩javaw.exe (201
1~100까지의 합 : 5050
<<1~N까지의 합 M개의 쓰레드로 계산 시간(ns)>>
  124366        254166        252959        540330        1040211
  136139        189870        279221        553009        1030852
  121046        154251        265637        537010        1029041
   98406        144289        308200        559951        1118090
   99916        147609        248129        548178        1152201
  118027        158476        317255        569610        1015760
  108367        148213        243903       1132882        1102998
  109877        152439        311822        612172        1169709
  111387        151232        264732        550292        1081263
  124669        168437        342310        607946        1251211
  132517        242696        343819        705146        1241552
  214925        224584        332952        581685        1130467
  140063        177192        317859        604023        1328789
  154854        211906        294314        658056        1126241
  206171        207680        284956        616097        1078245
  316047        252959        348950        562969        1089112
  532180        352874        396644        645981        1228571
  967161        578063        460337        691863        1160351
 1787919       1058021        706051        874188        1232797
 3601495       2413977       2215654       1699171       1940660
 7011309       3753029       2054159       2069856       2835977
13999374       7072284       4373050       3876187       4989448
27572521      13854179       7452931       8377528       7465307
56092582      28319022      24255379      14518876      14755232
```

과제 제출 방법 및 기한

- 주석을 사용하여 코드 설명
- 보고서 제출 있음(성능분석보고서)
- 프로젝트를 **export**한 압축파일 제출, 코드만 제출시 감점
- 압축파일 명 : **"Lab05"**
- 파일명에 학번, 이름 없이 위 압축파일 명으로 제출
- 다음주 화요일(10/18) 23:59까지
- 2차 추가 제출 – 수업시간 전까지