제 11 장 스레드(실습)

□ 개념 확인 학습

1. 어떤 경우에 스레드는 어떤 경우에 필요할까요?

2. CPU가 하나뿐인 시스템에서 여러 개의 스레드가 동시에 실행될 수 있도록 하는 방법에는 어떤 것이 있을까요?

3. 스레드를 생성하는 방법 두 가지는 무엇인가요?

4. Thread 클래스를 상속받아 스레드를 만드는 경우, 구현방법을 설명하세요.

5. Runnable 인터페이스를 이용해 스레드를 만드는 경우, 구현방법을 설명하세요.

6. 다음의 메소드는 어떠한 경우에 사용되나요?
(1) sleep()
(2) yield()
(3) start()
(4) run();
(5) stop();

□ 적용 확인 학습 & 응용 프로그래밍

1. 다음 프로그램의 출력을 예상해 보세요.

```
class Job implements Runnable {
        public void run() {
            int n = 0;
            while(true) {
                System.out.println(""+n++);
                if(n>100) break;
            }
        }
}
```

```
public class Test {
    public static void main(String[] args) {
        Thread t = new Thread(new Job());
        t.start();
    }
}
```

2. 위 1번 문제를 Thread 클래스를 상속받는 형태로 다시 작성해 보세요.

3. 위 2번에 Job 클래스의 생성자를 작성하고 생성자 매개 변수로 문자열을 전달합니다. 이후 스레드가 실행되면서 생성자에 전달된 문자열을 출력합니다. 이러한 동작을 하는 동일한 스레드를 두 개 실행시켜 어떠한 출력이 나타나는지 확인해 보세요.

4. 다음 프로그램의 출력을 예상해 보세요. 실행 후 예상한 출력과 비교하고 그렇게 출력된 이유를 설명해 보세요.

```
class MyJob implements Runnable {
        public void run() {
            try {
                System.out.println("A");
                Thread.sleep(1000);
                System.out.println("B");
            } catch (InterruptedException e) {
                System.out.println("C");
            }
            System.out.println("D");
        }
}
public class Test {
        public static void main(String[] args) {
            Thread t = new Thread(new MyJob());
            t.start();
            t.interrupt();
        }
}
```

5. 다음 프로그램의 출력을 예상해 보세요.

```
public class Test {
   private int count = 1;

   public synchronized void sub(String n) {
        for (int i = 0; i < 10; i++)
             System.out.println(n + ", " + count++);
   }
   public static void main(String[] args) {
        Test demo = new Test();
        Thread a1 = new A(demo, "a1");
        Thread a2 = new A(demo, "a2");
        a1.start();
        a2.start();
   }
}
class A extends Thread {
   Test demo;
   String name;

   public A(Test td, String name) {
        demo = td;
        this.name = name;
   }
   public void run() { demo.sub(name); }
}
```

6. 다음 설명에 해당하는 프로그램을 작성하세요.

(1) Thread를 상속받은 Job 클래스를 작성하세요.

: 필드(private String name, int num)

: 생성자(이름을 전달하여 저장)

: 스레드 메소드 (0.5초간 sleep() 후 getNum()메소드 호출 후 린턴 받은 결과 출력, getNum()에서 리턴 받은 값이 10 이상 이면 반복 탈출)

: getNum() (한 순간에 하나의 스레드만 접근할 수 있는 메소드, num값을 1증가 시킨 후 증가된 num값을 리턴)

(2) Test 클래스를 작성하세요.

: 메인 메소드에서 각각의 이름을 부여한 두 개의 스레드를 생성한 후 스레드 메소드를 호출하세요.

7. 다음 소스에서 주석의 설명을 고려하여 빈칸을 채우세요.

```
public class ThreadA extends Thread {
    public boolean stop = false;
    public boolean work = true;

    public void run() {
        while(!stop) {
            //work필드가 true이면 “ThreadA 작업” 출력, false이면 스레드의 실행을 양보한다.
            _____
            _____
        }
        System.out.println("ThreadA 종료");
    }
}

public class ThreadB extends Thread {
    public boolean stop = false;
    public boolean work = true;

    public void run() {
        while(!stop) {
            //work필드가 true이면 “ThreadB 작업” 출력, false이면 스레드의 실행을 양보한다.
            _____
            _____
        }
        System.out.println("ThreadB 종료");
    }
}

public class Example {
    public static void main(String[] args) {
        ThreadA threadA = new ThreadA();
        ThreadB threadB = new ThreadB();
        threadA.start();
        threadB.start();

        try { Thread.sleep(3000); } catch (InterruptedException e) {}
```

```
        threadA.work = false;

        try { Thread.sleep(3000); } catch (InterruptedException e) {}
        threadA.work = true;

        try { Thread.sleep(3000); } catch (InterruptedException e) {}
        threadA.stop = true;
        threadB.stop = true;
    }
}
```

8. 다음 소스에서 주석의 설명을 고려하여 빈칸을 채우세요.

```
public class SumThread extends Thread {
    private long sum;

    public long getSum() { return sum; }
    public void setSum(long sum) { this.sum = sum; }

    public void run() {
        for(int i=1; i<=100; i++) { sum+=i; }
    }
}
//
public class JoinExample {
    public static void main(String[] args) {
        SumThread sumThread = new SumThread();
        sumThread.start();
        try {
            //SumThread가 끝날 때 까지 기다린다.
            ____________________________________________________
        } catch (InterruptedException e) {
        }
        System.out.println("1~100 합: " + sumThread.getSum());
    }
}
```

9. 다음 소스에서 주석의 설명을 고려하여 빈칸을 채우세요.

```
public class DataBox {
    private String data;

    public _____    String getData() { //이 메소드는 동기화 되어 있다.
        //this.data가 null이면 다른 스레드가 알려줄 때 까지 기다린다.
        _____
        _____
        _____
        String returnValue = data;
        System.out.println("ConsummerThread가 읽은 데이터: " + returnValue);

        //this.data를 null로 만들고 다른 스레드에게 알려준다.
        _____
        _____
        return returnValue;
    }
    public _____ void setData(String data) { //이 메소드는 동기화 되어 있다.
        //this.data가 null이 아니면 다른 스레드가 알려줄 때 까지 기다린다.
        _____
        _____
        _____

        System.out.println("ProducerThread가 생성한 데이터: " + data);

        //this.data에 전달되어 온 데이터를 저장하고 다른 스레드에게 알려준다.
        _____
        _____
    }
}

//
public class ProducerThread extends Thread {
    private DataBox dataBox;

    public ProducerThread(DataBox dataBox) { this.dataBox = dataBox; }

    @Override
    public void run() {
        for(int i=1; i<=3; i++) {
            String data = "Data-" + i;
            dataBox.setData(data);
```

```
        }
    }
}
//
public class ConsumerThread extends Thread {
    private DataBox dataBox;

    public ConsumerThread(DataBox dataBox) { this.dataBox = dataBox; }

    @Override
    public void run() {
        for(int i=1; i<=3; i++) {
            String data = dataBox.getData();
        }
    }
}
//
public class WaitNotifyExample {
    public static void main(String[] args) {
        DataBox dataBox = new DataBox();

        ProducerThread producerThread = new ProducerThread(dataBox);
        ConsumerThread consumerThread = new ConsumerThread(dataBox);
        producerThread.start();
        consumerThread.start();
    }
}
```