

JAVA프로그래밍

과제2

2019305059

이현수

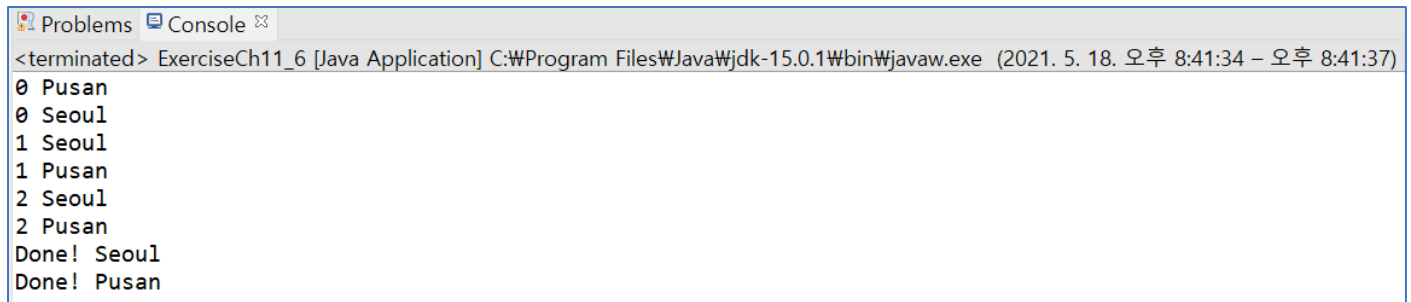
<문제>

11.6 다음 프로그램의 실행 결과를 쓰시오. 그리고 실행 결과를 분석하시오.

<코드>

```
1 class SimpleThread extends Thread{
2     public SimpleThread(String name) {
3         super(name);
4     }
5     public void run() {
6         for(int i=0;i<3;i++) {
7             System.out.println(i+" "+getName());
8             try {
9                 sleep((int)(Math.random()*1000));
10            }
11            catch(InterruptedException e) { }
12        }
13        System.out.println("Done! "+getName());
14    }
15 }
16 public class ExerciseCh11_6 {
17     public static void main(String[] args) {
18         new SimpleThread("Seoul").start();
19         new SimpleThread("Pusan").start();
20     }
21 }
```

<실행결과>



```
<terminated> ExerciseCh11_6 [Java Application] C:\Program Files\Java\jdk-15.0.1\bin\javaw.exe (2021. 5. 18. 오후 8:41:34 - 오후 8:41:37)
0 Pusan
0 Seoul
1 Seoul
1 Pusan
2 Seoul
2 Pusan
Done! Seoul
Done! Pusan
```

<설명>

SimpleThread 클래스에 Thread를 상속받아 스레드를 구현한다. Thread의 run() 메소드를 오버라이딩해 구현한다. 우선 매개변수 있는 생성자를 통해 Thread 클래스의 이름을 super(name);을 통해 저장한다. 그리고 run메소드를 오버라이딩해 구현한다. for반복문을 총 3번 반복한다. i=0부터 i=2까지 반복한다. 반복문에서는 i와 getName메소드를 통해 Thread의 이름을 같이 출력해준다. 그리고 try문에서 sleep메소드를 통해 0~999ms만큼 랜덤하게 해당스레드를 중지한다. Catch는 InterruptedException을 매개변수로 받고 빈 블록으로 남겨둔다. for반복문을 빠져나오면 "Done! "과 getName메소드를 통해 해당 스레드 이름을 같이 출력한다.

메인메소드에서 무명클래스로 SimpleThread 2개를 만들고, 각각 "Seoul", "Pusan"을 생성자 매개변수로 넘겨준다. 그리고 바로 start메소드를 호출해 스레드를 동작시킨다.

이때 for반복문을 통해 총 3번씩 반복이되는데, 이때 순서는 실행시킬 때마다 달라질 수 있다. 그리고 각각 for반복문을 빠져나와 "Done! getName()"을 출력해준다. 이때 getName()으로 "Seoul", "Pusan"이 나온다.

<문제>

11.7 다음 프로그램을 실제로 자바 시스템을 통하여 실행하시오. 그리고 실행 결과를 분석하여 프로그램의 기능을 말로 설명하시오.

<코드>

```
1 class MyThread extends Thread{
2     MyThread(String name){
3         super(name);
4     }
5     public void run() {
6         while(true) {
7             System.out.println(getName()+" is now running");
8             try {
9                 sleep(5*1000);
10            }
11            catch(InterruptedException e) {
12                System.out.println("Interrupted is received");
13                break;
14            }
15        }
16    }
17 }
18 public class ExerciseCh11_7 {
19     public static void main(String[] args) {
20         MyThread t1 = new MyThread("ThreadOne");
21         MyThread t2 = new MyThread("ThreadTwo");
22         t1.start();
23         t2.start();
24         try {
25             Thread.sleep(30*1000);
26         }
27         catch(Exception e) { }
28         t1.interrupt();
29         t2.interrupt();
30     }
31 }
```

<실행결과>

Problems Console

<terminated> ExerciseCh11_7 [Java Application] C:\Program Files\Java\jdk-15.0.1\bin\javaw.exe (2021. 5. 18. 오후 8:42:26 – 오후 8:42:58)

```
ThreadOne is now running
ThreadTwo is now running
ThreadOne is now running
ThreadTwo is now running
ThreadTwo is now running
ThreadOne is now running
ThreadTwo is now running
ThreadOne is now running
ThreadTwo is now running
ThreadOne is now running
ThreadTwo is now running
ThreadOne is now running
Interrupted is received
Interrupted is received
```

<설명>

MyThread클래스는 Thread를 상속받아 구현한다. 매개변수 있는 생성자를 만들어 Thread이름을 name으로 설정한다. 그리고 run메소드를 오버라이딩해 구현한다. while무한 반복문 안에 getName메소드로 스레드 이름과 "is now running"이 출력되게 한다. 그리고 try문 안에 sleep(5*1000);을 통해 5초동안 해당 스레드가 정지하고, catch는 InterruptedException매개변수로 받아서 예외처리를 해주는데 catch블록 안에 "Interrupted is received"를 출력하고 while반복문을 빠져나간다.

메인메소드에서 MyThread 객체 t1, t2를 만든다. 그리고 생성자 매개변수로 각각 "ThreadOne"과 "ThreadTwo"를 넘겨준다. 그리고 t1, t2모두 start메소드를 통해 스레드가 실행되고 try문에서 Thread.sleep(30*1000); 즉 30초 동안 메인문이 정지되게 된다. 이때 t1, t2의 스레드는 정상 작동된다. 그 후 30초 후에 메인메소드의 28,29번줄 코드 interrupt메소드가 실행되 스레드가 종료되게 된다. 그러면서 각 객체의 run메소드 내에 catch블록안 코드가 실행되고 프로그램은 모두 종료된다.

실행결과를 보면 ThreadOne, ThreadTwo를 한사이클로보면 이 사이클이 6번 반복되고 종료된다. 그 이유는 한번 사이클이 5초가 딜레이되므로 $6 \times 5 = 30$ 초이므로 메인메소드의 Thread.sleep이 30초가 딜레이되므로 "ThreadOne is now running", "ThreadTwo is now running"이 총 6번 출력되고 "Interrupted is received", "Interrupted is received"가 출력되고 종료된다.



<문제>



11.8 [예제11.5]의 Account.java를 이용하여 은행 계정 업무를 병행으로 수행하기 위한 예제 프로그램을 작성하여 보시오. 은행원에 대한 작업 순서는 다음과 같다.



<코드>

```
1 class Account{
2     private double balance;
3
4     public Account(double initialDeposit) {
5         balance = initialDeposit;
6     }
7     public synchronized double getBalance() {
8         return balance;
9     }
10    public synchronized void deposit(double amount) {
11        balance += amount;
12    }
13 }
14
15 class Teller extends Thread{
16     String name;
17     Account account;
18     double amount;
19     Teller(String name, Account account, double amount){
20         this.name = name;
21         this.account = account;
22         this.amount = amount;
23     }
24     public void run() {
25         account.deposit(amount);
26         System.out.println(name+" : "+account.getBalance());
27     }
28 }
29
30 public class ExerciseCh11_8 {
31     public static void main(String[] args) {
32         Account account = new Account(50000);
33
34         Teller teller1 = new Teller("A", account, 5000);
35         Teller teller2 = new Teller("B", account, 1000);
36         Teller teller3 = new Teller("C", account, 2000);
37
38         teller1.start();
39         teller2.start();
40         teller3.start();
41     }
42 }
```

<실행결과>

 Problems	 Console	
<terminated> ExerciseCh11_8 [Java Application] C:\Program Files\Java\jdk-15.0.1\bin\javaw.exe (2021. 5. 18. 오후 8:50:56 – 오후 8:50:57)		
A : 56000.0		
B : 58000.0		
C : 58000.0		

 Problems	 Console	
<terminated> ExerciseCh11_8 [Java Application] C:\Program Files\Java\jdk-15.0.1\bin\javaw.exe (2021. 5. 18. 오후 8:51:20 – 오후 8:51:20)		
A : 56000.0		
B : 56000.0		
C : 58000.0		

 Problems	 Console	
<terminated> ExerciseCh11_8 [Java Application] C:\Program Files\Java\jdk-15.0.1\bin\javaw.exe (2021. 5. 18. 오후 8:51:32 – 오후 8:51:32)		
A : 58000.0		
B : 58000.0		
C : 58000.0		

실행결과는 실행될때마다 달라지는데 달라지는 이유는 밑에 <설명>란에 설명되었다.

<설명>

클래스 Account를 만든다. 접근지정자 private인 double형 변수 balance를 선언해준다. 이 자료형은 계좌의 잔고를 의미한다. 그리고 매개변수 있는 생성자를 만들어준다. 매개변수로 받은 double형 자료형은 Account클래스의 balance 자료형을 초기화 해준다. 그리고 double형을 반환하는 getBalance메소드를 구현한다. 여기서 getBalance 메소드는 synchronized키워드를 사용해 getBalance메소드를 통째로 임계영역 지정한다. 그래서 스레드간에 동기화를 시켜준다. 그래서 다수의 스레드가 공유자원에 동시에 접근하는 문제점을 해결한다.

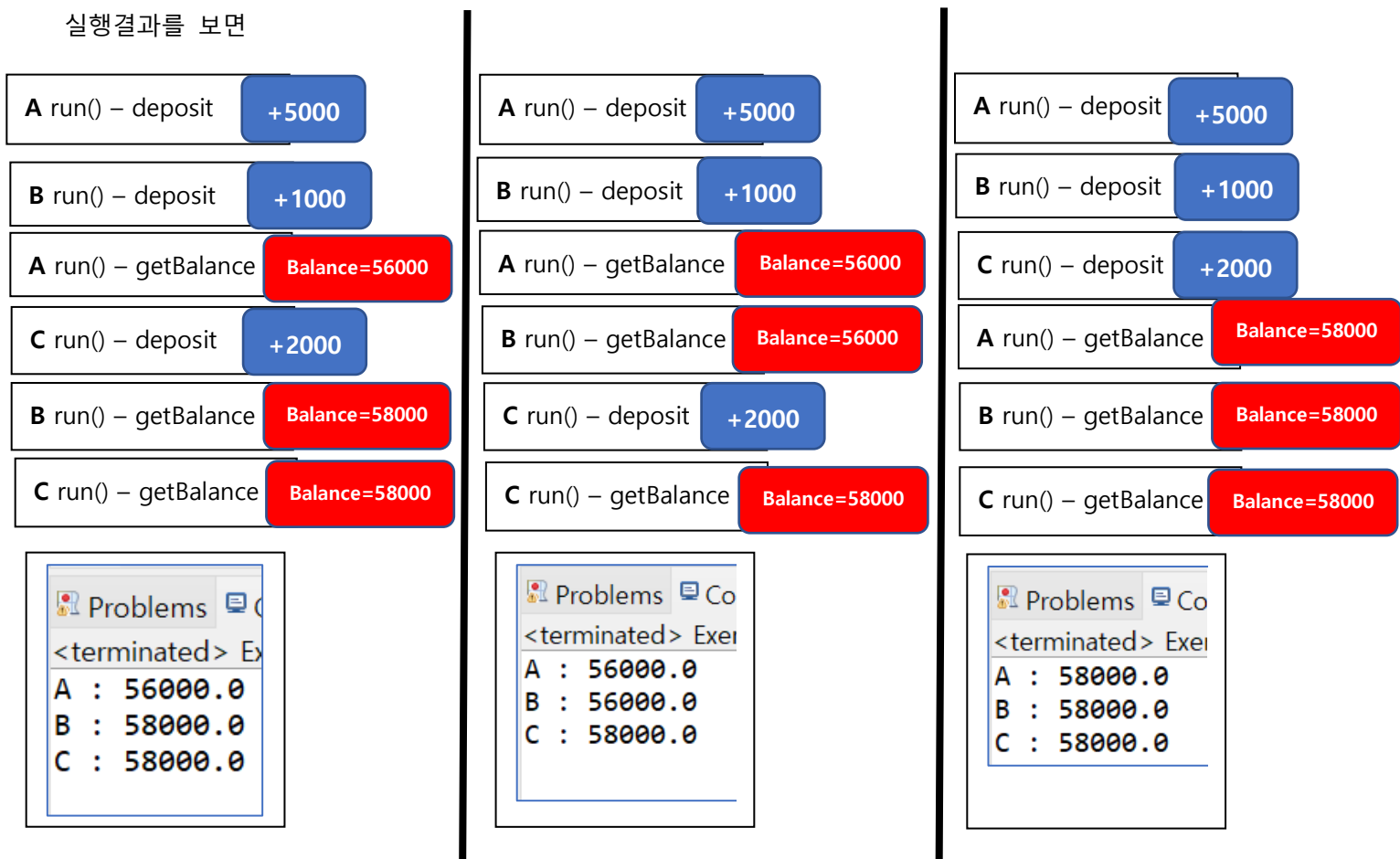
마찬가지로 amount매개변수를 받아 계좌잔고를 amount만큼 늘려주는 deposit메소드를 구현하는데 이 역시 synchronized키워드를 사용해 deposit메소드 전체를 임계영역으로 지정해 스레드간 동기화 해준다. 그래서 다수의 스레드가 공유자원에 동시에 접근하는 문제점을 해결한다.

클래스Teller는 Thread를 상속받는다. 그리고 teller의 이름을 의미하는 String형 변수 name, teller가 처리할 계좌를 의미하는 Account클래스 account와 처리할 금액을 의미하는 double형변수 amount 필드를 가진다.

Teller생성자를 만들어주는데 String형, Account클래스형, double형 매개변수 3개를 받아 3개의 필드를 각각 초기화 시킨다. 그리고 run메소드는 오버라이딩해 구현한다. run메소드 안에는 account의 메소드 deposit과 getBalance를 호출한다.

메인메소드안에는 Account클래스 객체 account를 만들고 account객체의 잔고 balance는 50000으로 초기화한다. 그리고 Teller클래스 객체 teller1, teller2, teller3를 만들고, teller1객체는 이름을 "A", 처리할 계좌는 account객체, 입금할 금액은 5000으로 초기화한다. teller2객체는 이름을 "B", 처리할 계좌는 account객체, 입금할 금액은 1000으로 초기화한다. Teller3객체는 이름을 "C", 처리할 계좌는 account객체, 입금할 금액은 2000으로 초기화한다. 그리고 teller1, teller2, teller3객체의 start메소드를 실행해 세개의 스레드가 실행되게 한다.

실행결과를 보면



위 그림과 같이 스레드 동작순서가 실행때마다 다르기 때문에 실행결과도 실행 때마다 다르다.

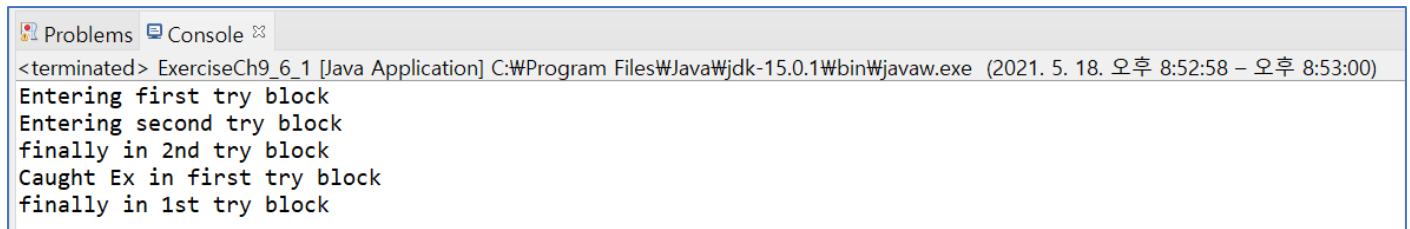
<문제>

9.6.(1) 다음 프로그램의 실행 결과를 보이시오.

<코드>

```
1 class Ex extends Exception{}
2 public class ExerciseCh9_6_1 {
3     public static void main(String[] args) {
4         System.out.println("Entering first try block");
5         try {
6             System.out.println("Entering second try block");
7             try {
8                 throw new Ex();
9             }
10            finally {
11                System.out.println("finally in 2nd try block");
12            }
13        }
14        catch (Ex e) {
15            System.out.println("Caught Ex in first try block");
16        }
17        finally {
18            System.out.println("finally in 1st try block");
19        }
20    }
21 }
```

<실행결과>



```
<terminated> ExerciseCh9_6_1 [Java Application] C:\Program Files\Java\jdk-15.0.1\bin\javaw.exe (2021. 5. 18. 오후 8:52:58 - 오후 8:53:00)
Entering first try block
Entering second try block
finally in 2nd try block
Caught Ex in first try block
finally in 1st try block
```

<설명>

클래스 Ex는 Exception을 상속받는다.

4번째 줄코드가 실행되 "Entering first try block"이 출력된다.

그리고 밑에 있는 try블록으로 진입해 "Entering second try block"이 출력된다. 그밑에 try블록이 또 있다. 여기에 있는 throw new Ex();를 통해 예외가 발생하게되어 첫번째 try블록의 catch블록이 실행된다.

하지만 그 이전에 두번째 try블록의 finally블록이 실행되게 된다. Finally는 예외발생 여부와 상관없이 무조건 실행된다는 특징이 있다. 그래서 "finally in 2nd try block"이 출력된다.

그리고 아까 두번째 try블록에 의해 예외가 발생해 첫번째 catch블록이 실행되는데 여기서 "Caught Ex in first try block"이 출력된다.

그리고 그밑에 있는 finally블록안에 있는 코드가 실행되 "finally in 1st try block"이 출력된다.

결과적으로 실행결과를 보면 프로그램 코드에 있는 모든 System.out.println이 실행되 모든 문장이 출력되게 된다.

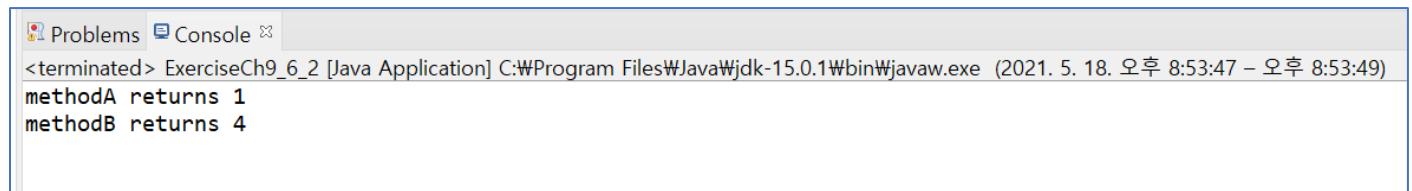
<문제>

9.6.(2) 다음 프로그램의 실행 결과를 보이시오.

<코드>

```
1 class FinallyClause{
2     public int methodA() {
3         try {
4             return 1;
5         }
6         catch(Exception e) { return 2; }
7     }
8     public int methodB() {
9         try {
10            return 3;
11        }
12        finally {
13            return 4;
14        }
15    }
16 }
17 public class ExerciseCh9_6_2 {
18     public static void main(String[] args) {
19         FinallyClause fc = new FinallyClause();
20         System.out.println("methodA returns "+fc.methodA());
21         System.out.println("methodB returns "+fc.methodB());
22     }
23 }
```

<실행결과>



```
<terminated> ExerciseCh9_6_2 [Java Application] C:\Program Files\Java\jdk-15.0.1\bin\javaw.exe (2021. 5. 18. 오후 8:53:47 - 오후 8:53:49)
methodA returns 1
methodB returns 4
```

<설명>

클래스 FinallyClause를 만들어 준다. 이 클래스 안에는 public이고 int형을 반환하는 methodA를 만들어준다. methodA메소드는 try블록안에 1을 반환해주고, catch블록은 Exception매개변수를 가지고있고 2를 반환한다. 그리고 public인 int형을 반환하는 methodB를 구현한다. methodB는 try블록안에서 3을 반환하고, finally블록에서는 4를 반환한다.

메인메소드에서 FinallyClause객체 fc를 만들고 fc.methodA()와 fc.methodB()를 각각 소환해 출력해준다.

출력결과는 methodA는 1, methodB는 4라고 출력되는데 그 이유는 methodA메소드는 예외가 발생하지 않는이상 1을 반환하게 되었기 때문에 1이 출력되고, methodB메소드는 try블록에 의해 3이반환되지만 예외 발생여부와 상관없이 finally블록은 무조건 실행되기 때문에 순서상 try블록보다 실행순서가 더 뒤에있는 finally블록에 있는 4가 반환되 4가 출력되게 된다.

<문제>

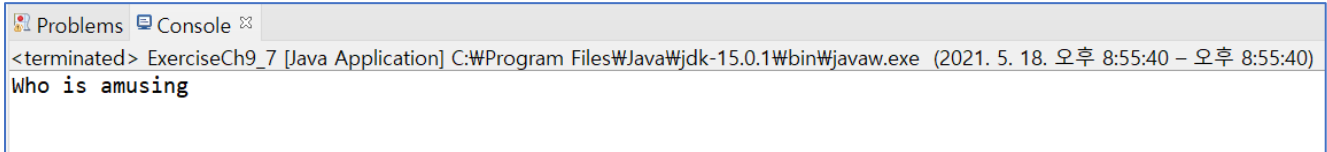
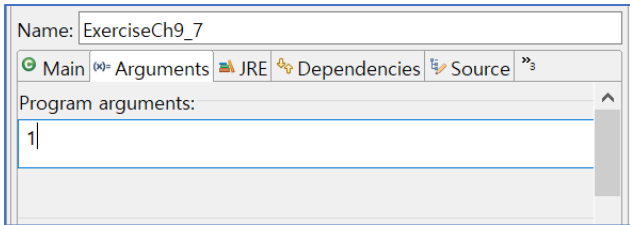
9.7 다음 프로그램은 명령어 라인 매개변수의 값에 따라 다른 형태의 결과를 갖는다. 명령어 라인 매개변수로 1 부터 5까지 각각의 값에 따른 결과를 예측하여 보시오.

<코드>

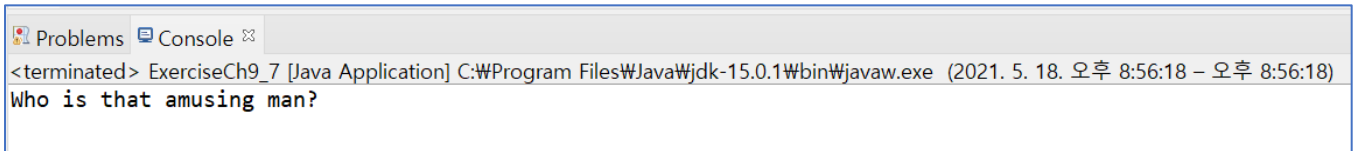
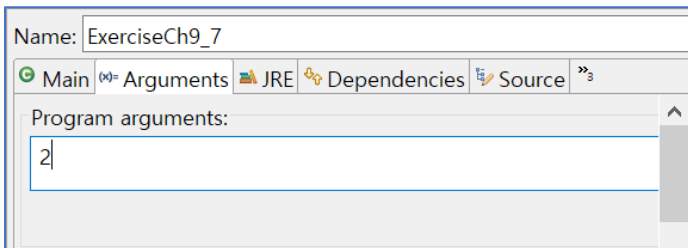
```
1 public class ExerciseCh9_7 {
2     public static void main(String[] args) {
3         int mysteriousState = Integer.parseInt(args[0]);
4
5         while(true) {
6             System.out.print("Who ");
7             try {
8                 System.out.print("is ");
9                 if(mysteriousState==1) return;
10                System.out.print("that ");
11                if(mysteriousState==2) break;
12                System.out.print("strange ");
13                if(mysteriousState==3) continue;
14                System.out.print("but kindly ");
15                if(mysteriousState==4)
16                    throw new Error();
17                System.out.print("now at all ");
18            }
19            finally {
20                System.out.print("amusing ");
21            }
22            System.out.print("yet compelling ");
23            break;
24        }
25        System.out.print("man?");
26    }
27 }
```

<실행결과><설명>

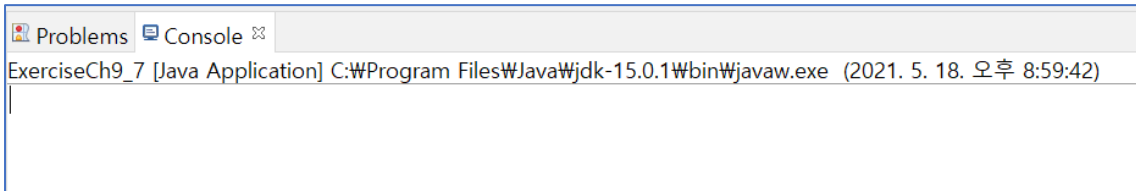
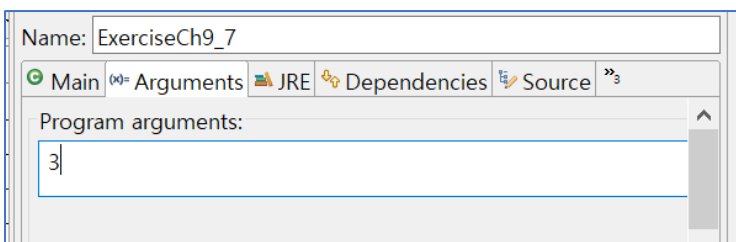
프로그램을 실행하면서 매개변수를 넘겨주는데 그 값을 Integer.parseInt를 통해 정수로 바꿔 mysteriousState 정수형 변수에 저장한다. 그리고 while 무한 반복문을 작성한다. while반복문 안에는 "Who"를 무조건 출력한다. 그리고 try블록과 finally블록이 작성된다. try블록 안에는 "is"를 무조건 출력되게 된다.



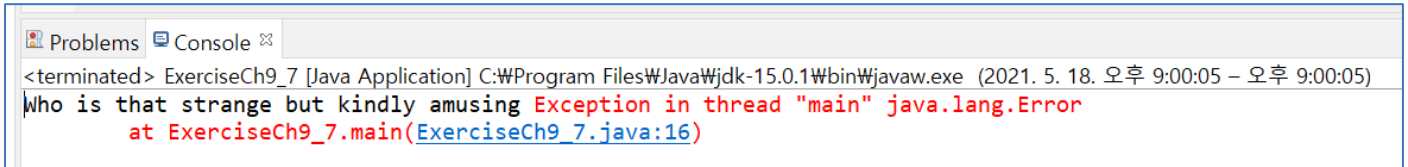
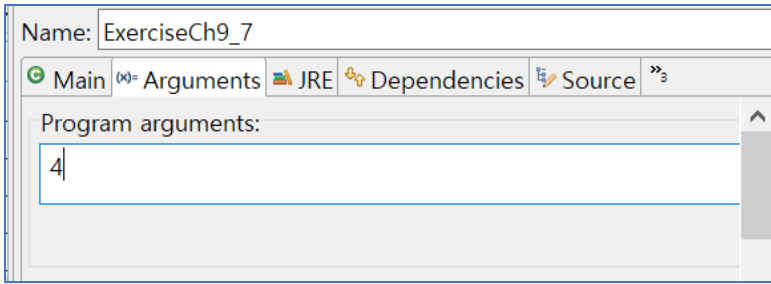
mysteriousState값이 1이면 return;이라서 메인메소드 즉 프로그램이 종료되는데 그럼에도 불구하고 조건문이 try 블록에서 실행된거라 finally블록이 실행되고 메인메소드가 종료되게 된다. 그래서 "Who is amusing"이 출력된다.



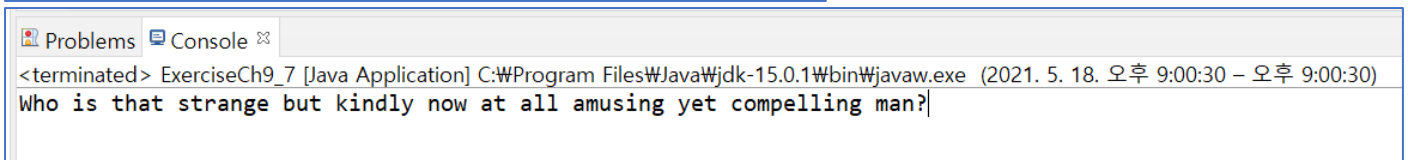
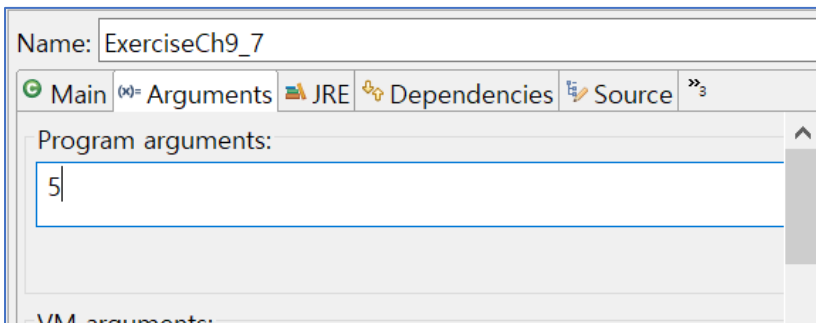
mysteriousState값이 2라서 try블록안 조건문에 의해 10번째줄 코드에 의해 "that"까지 출력되게 된다. mysteriousState값이 2라서 break;가 실행되 while반복문에서 빠져나오는데 이때 try블록안에서 실행된 조건문이라 finally블록안에 "amusing"이 출력되게 되고, while반복문을 빠져나오면 25번째 줄 코드가 실행되 "man?"이 출력된다. 그래서 "Who is that amusing man?"이 출력된다.



mysteriousState값이 3이면 조건문에 의해서 continue;가 실행되는데 continue는 그 밑에 있는 코드는 모두 무시되고 while반복문 블록의 맨 처음으로 다시 돌아간다. 문제는 mysteriousState값은 3으로 고정되어있어서 이 과정이 무한반복되 프로그램은 종료되지 않는다.



mysteriousState값이 4라면 "Who is that strange but kindly"까지 실행되고 if(mysteriousState==4)코드를 만나게 된다. 조건문이 만족되면 throw new Error(); 코드가 실행되는데 이 예외를 받아줄 catch블록이 없어서 예외처리를 못하고 비정상적인 종료가 일어난다. 다만 여기서 예외발생 여부와 상관없이 finally블록이 실행되 "amusing"이 출력되어 결과적으로 "Who is that strange but kindly amusing"까지 출력되고 오류가 발생해 종료된다.



mysteriousState값이 5라면 try블록안 조건문에 걸리지 않는다. 그래서 "Who is that strange but kindly now at all"이 출력되고 finally블록은 무조건 실행되므로 추가로 "amusing"이 출력된다. 그리고 22번째 줄 코드가 실행되 "yet compelling"이 출력되고 23번째줄 break;가 실행되 while반복문에서 빠져나와 25번째 줄 코드가 실행되 "man?"이 출력된다.

결과적으로 "Who is that strange but kindly now at all amusing yet compelling man?"이 출력된다.