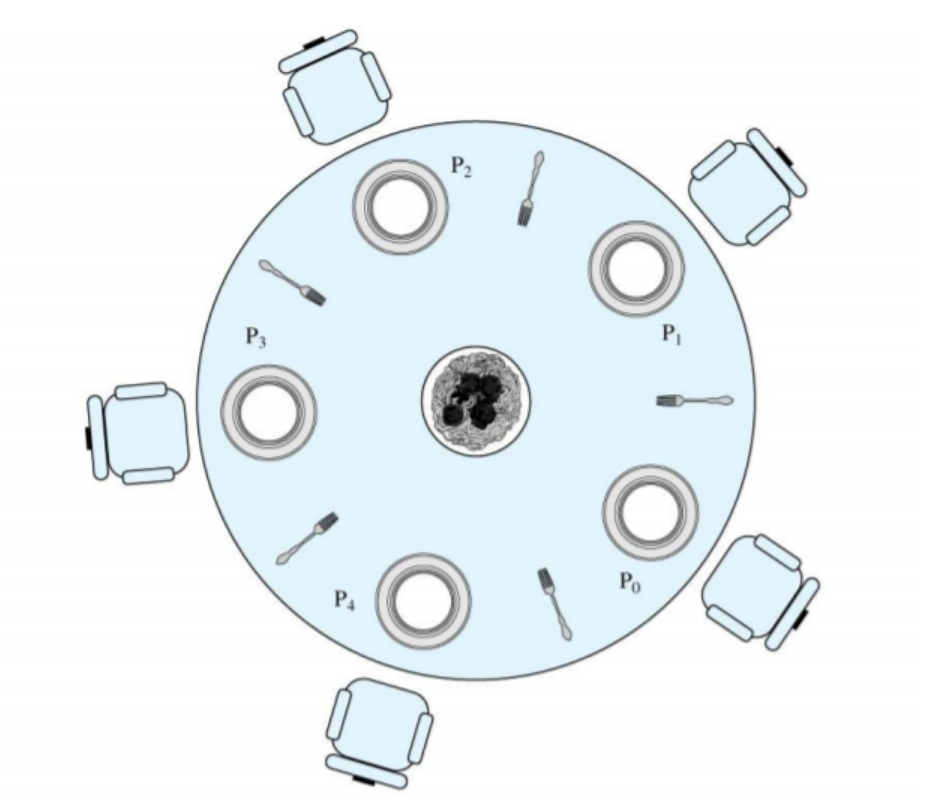
**과제2 : 식사하는 철학자 문제**



**식사하는 철학자 문제**

철학자 다섯이서 원형 식탁에 둘러앉아 생각에 빠지다가, 배고플 땐 밥을 먹는다. 그들의 양쪽엔 각각 젓가락 한 짠씩 놓여있고, 밥을 먹으려 할 때 다음의 과정을 따른다.

1. 왼쪽 젓가락부터 집어든다. 다른 철학자가 이미 왼쪽 젓가락을 쓰고 있다면 그가 내려놓을 때까지 생각하며 대기한다.
2. 왼쪽을 들었으면 오른쪽 젓가락을 든다. 들 수 없다면 1번과 마찬가지로 들 수 있을 때까지 생각하며 대기한다.
3. 두 젓가락을 모두 들었다면 일정 시간동안 식사를 한다.
4. 식사를 마쳤으면 오른쪽 젓가락을 내려놓고, 그 다음 왼쪽 젓가락을 내려놓는다.
5. 다시 생각하다가 배고프면 1번으로 돌아간다.

**발생할 수 있는 문제**

식사하는 철학자 문제는 교착상태의 대표적인 예제. 위의 상황을 프로그램으로 만들어 실행하면 잘 돌아가다가 어느순간 멈춰버림. 만약 모든 철학자가 동시에 배가 고파서 왼쪽 젓가락을 집어든다면 오른쪽 젓가락은 이미 자신의 우측에 앉은 철학자가 집어들었을 것이므로, 모두가 영원히 오른쪽 젓가락을 들지 못하게 된다. 그렇게 상단 과정의 2번에서 더이상 진행하지 못하고 철학자들은 영원히 생각에 빠지게 되는데, 이런 현상을 컴퓨터과학에선 교착상태(Deadlock)라고 한다.

또 포크를 집는 동작 혹은 놓는 동작이 여러 스레드가 동시에 접근하게 되면 공유자원을 의도치 않게 잘못된 동작을 하게 되는 경우가 있어서 스레드가 특정 동작에 있어서 특정 시간에 한 개의 스레드만 접근가능하게 만들어야 한다. 즉 임계영역을 만들어야 한다. 즉 동기화문제가 있다.

포크 = **공유자원**

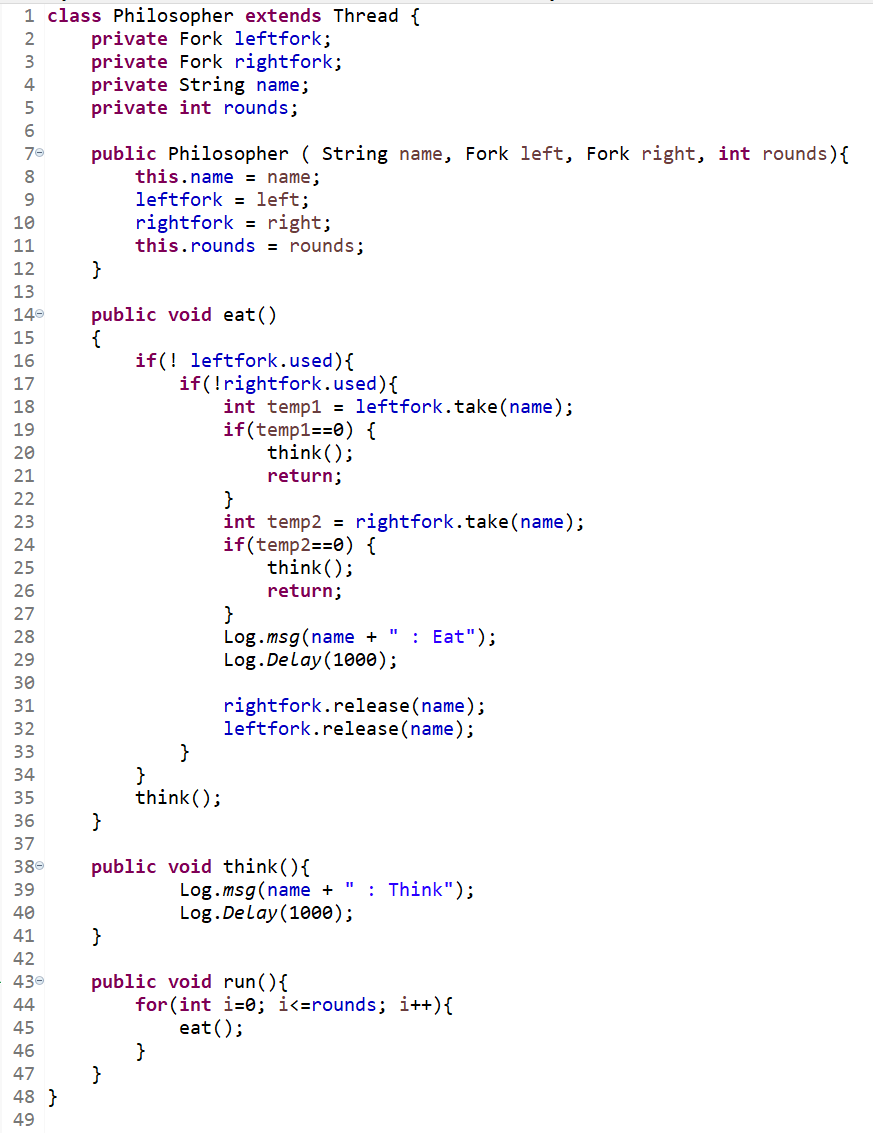
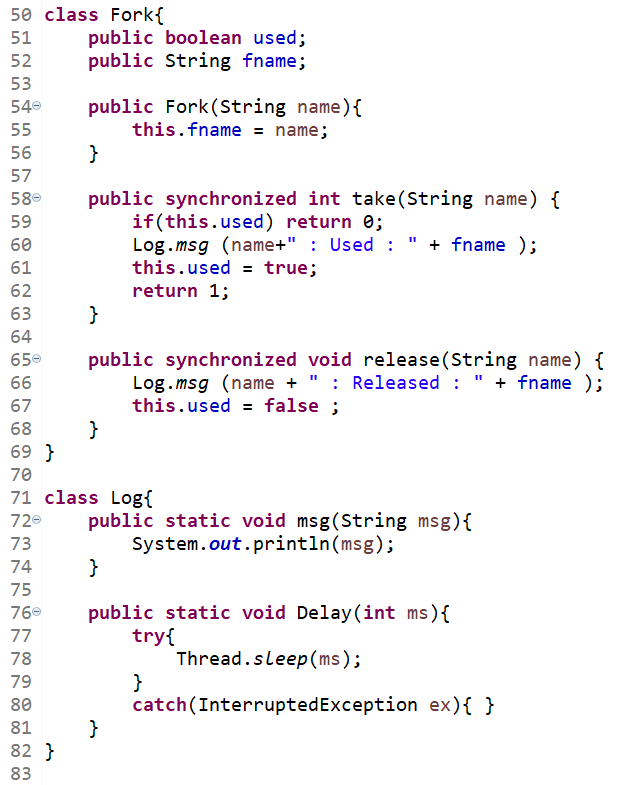
철학자 = **프로세스**

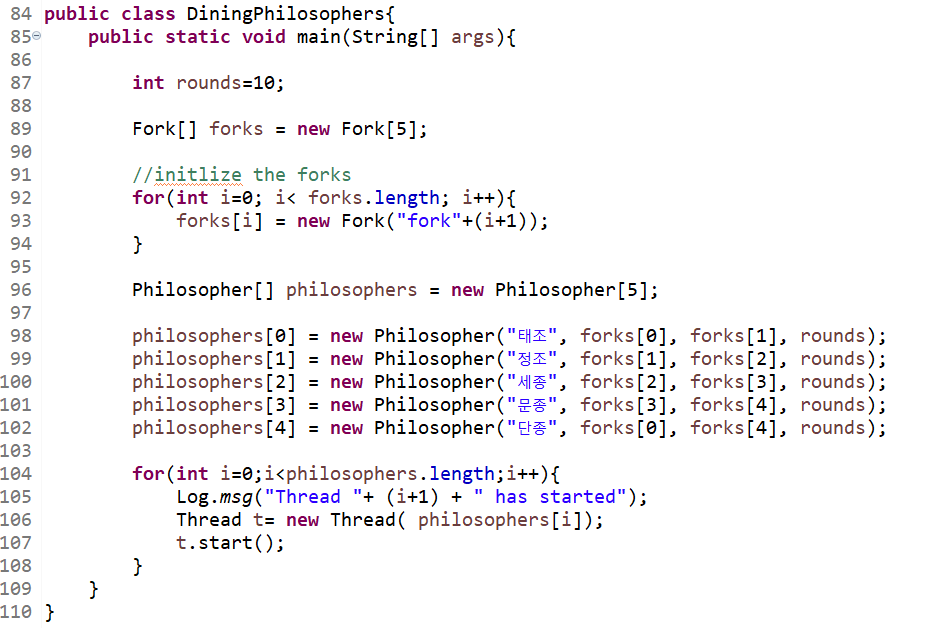
**해결방법**

철학자 문제에서 젓가락에 1부터 5번까지 번호를 부여하고, 철학자들은 가장 낮은 번호를 우선 취하도록 강요하는 것이다. 1과 2번 젓가락 사이에 있는 철학자와 1과 5번 사이에 있는 철학자는 그의 오른쪽에 있는 젓가락을 잡는 것이 아니라 같은 젓가락 1번을 우선 잡아야 한다. 젓가락 1을 잡지 못한 철학자는 먼저 잡은 철학자가 젓가락을 놓을 때까지 기다려야 한다. 이러한 상황에서 데드락은 발생하지 않게 된다.

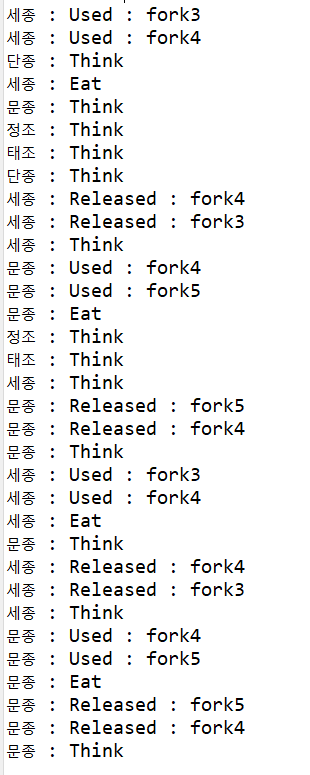
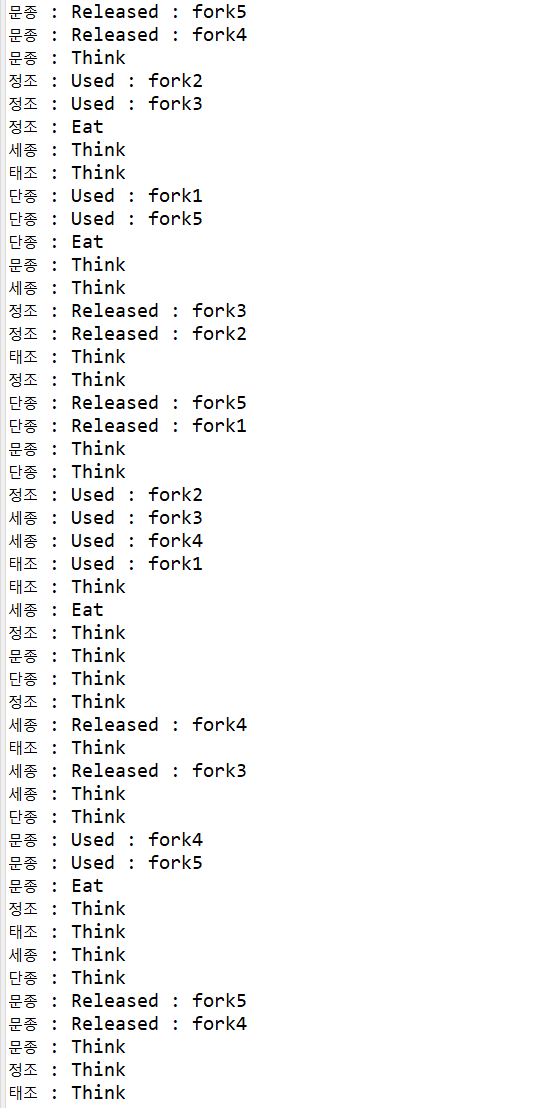
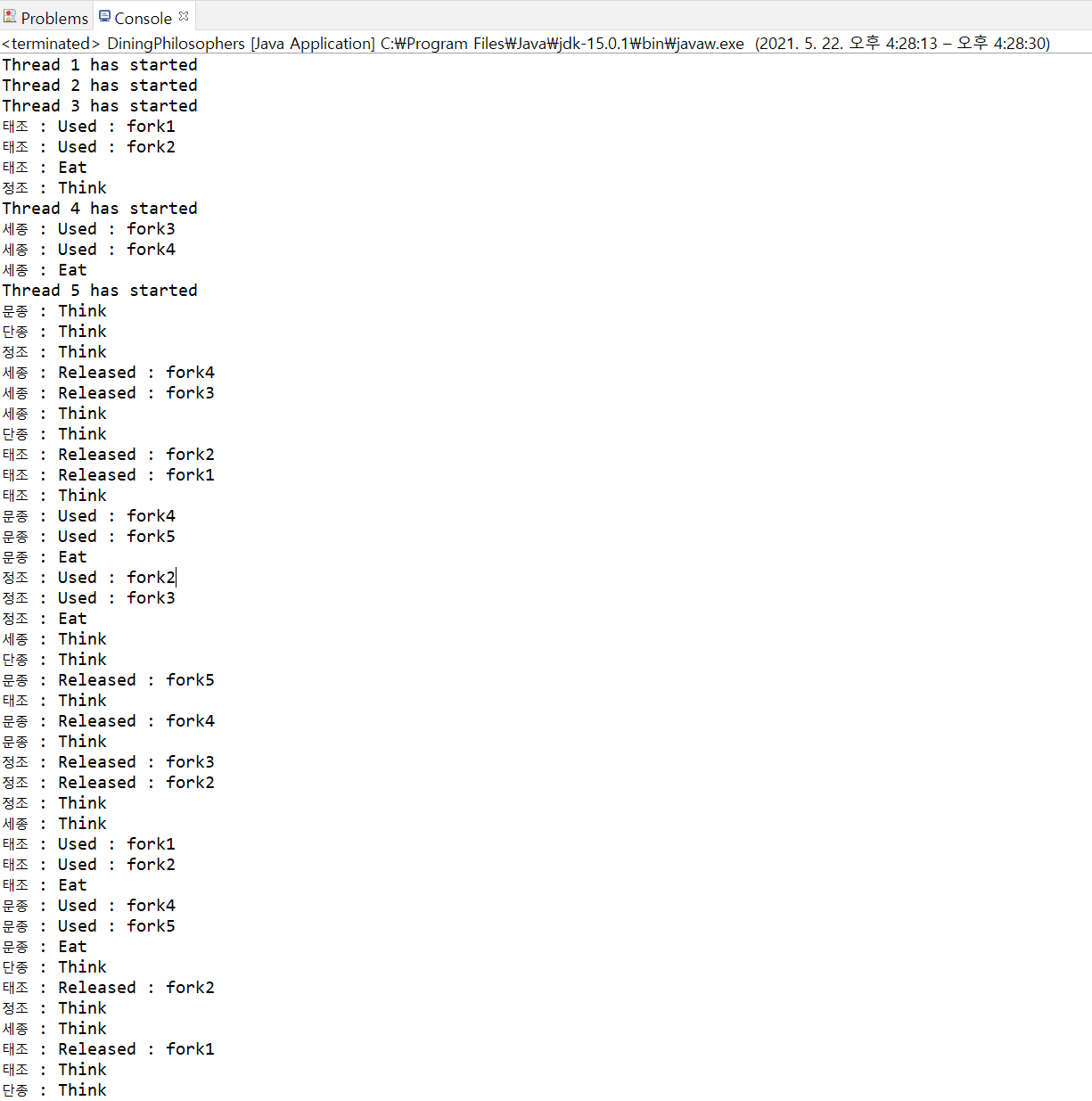
동기화 문제를 해결하기 위해서 **synchronized** 키워드를 사용해서 해결한다.

**<코드>**



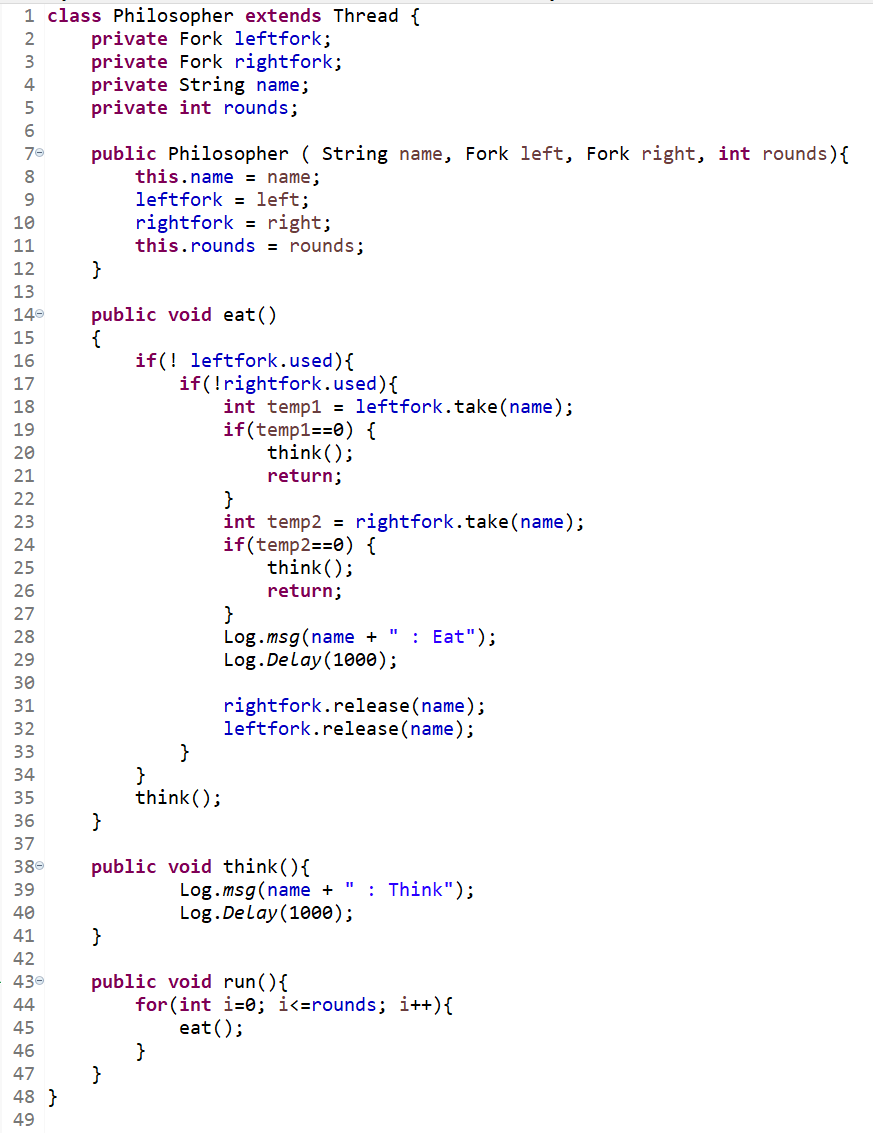
**<실행결과>**



Eat 순서 : 태조, 세종, 문종, 정조, 태조, 문종, 정조, 단종, 세종, 문종, 세종, 문종, 세종, 문종

실행결과 태조, 정조, 세종, 문종, 단종 모두 최소 한번씩 식사를 함.

**<설명>**



철학자를 의미하는 Philosopher 클래스를 만든다. 이 클래스는 Thread를 상속받아 스레드를 구현해주게 된다.

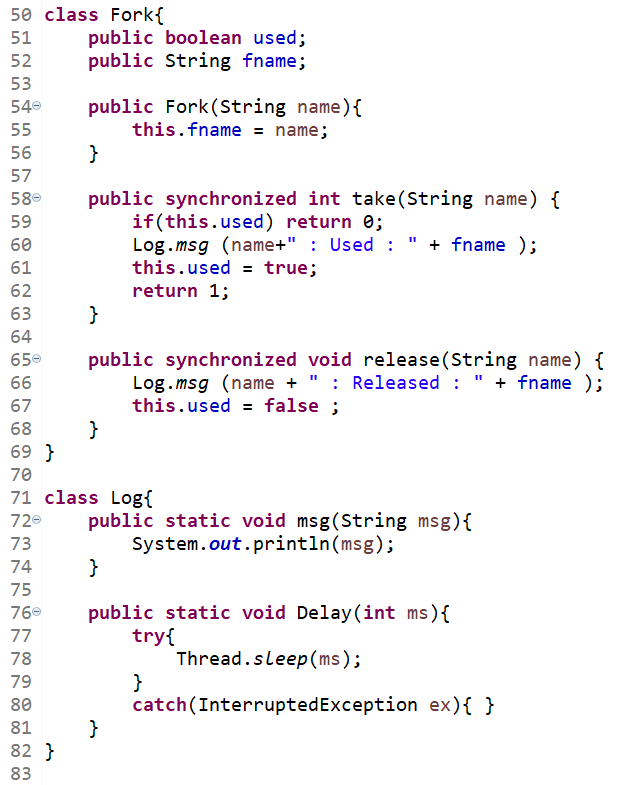
Philosopher클래스는 총 4개의 필드를 가진다. Fork클래스 leftfork, rightfork를 필드로 가진다. 각각 철학자 기준 왼쪽, 오른쪽에 놓여져 있는 포크를 의미한다. String형 name필드는 철학자의 이름을 의미한다. 정수형 변수 rounds는 eat을 시도하는 횟수라고 보면된다.

(코드7~12) Philosopher의 매개변수 있는 생성자를 만든다. 그래서 4개의 필드를 모두 초기화 하는 작업을 한다.

(코드14~36) eat메소드에서는 이중 조건문을 통해 철학자의 왼쪽포크와 오른쪽포크가 사용중인지 체크한다. 만약 한 개라도 사용중 이라면 조건문 조건이 만족을 못해 think메소드가 실행된다. 만약 철학자가 식사하기 위한 왼쪽 포크와 오른쪽 포크 두개 모두 사용중이 아니라면 이중 조건문 조건이 만족이되 조건문 안의 코드가 실행된다. 왼쪽포크와 오른쪽포크 순서대로 take메소드를 통해 사용하고 1초동안 delay후 오른쪽, 왼쪽 포크 순서대로 release하게 된다. 근데 여기에 문제가 발생하게 된다. 스레드가 동시에 실행되기 때문에 식사를 하기위에 필요한 포크가 겹치는 철학자 둘이 동시에 이 조건문이 실행되 둘다 조건문의 조건이 만족이되 이중조건문 안의 코드를 동시에 실행된다고 했을 때 두개의 철학자가 동시에 같은 포크를 사용하는 오류가 발생하게 된다. 그래서 이때 take를 할 때 temp1, temp2 정수형 변수에 값을 반환받아 반약 변수에 0이 반환되면 누군가가 이미 이포크를 선점했다는 것을 의미해 think메소드를 호출하고 return;해 함수를 종료하게 된다.

(코드38~41) 철학자가 eat을 하지 못한경우 think를 하게된다. Log클래스의 msg메소드를 통해 관련 내용을 알려주고 Delay메소드를 통해 1초동안 해당 스레드를 sleep한다. 이것의 의미는 1초 후에 먹는걸 다시 시도한다는 의미이다.

(코드43~47) thread를 상속받았다. 그래서 run메소드를 오버라이딩해 구현해준다. for반복문을 통해 rounds+1만큼 eat메소드를 호출한다. 즉 rounds+1만큼 eat을 시도하는 것이다.

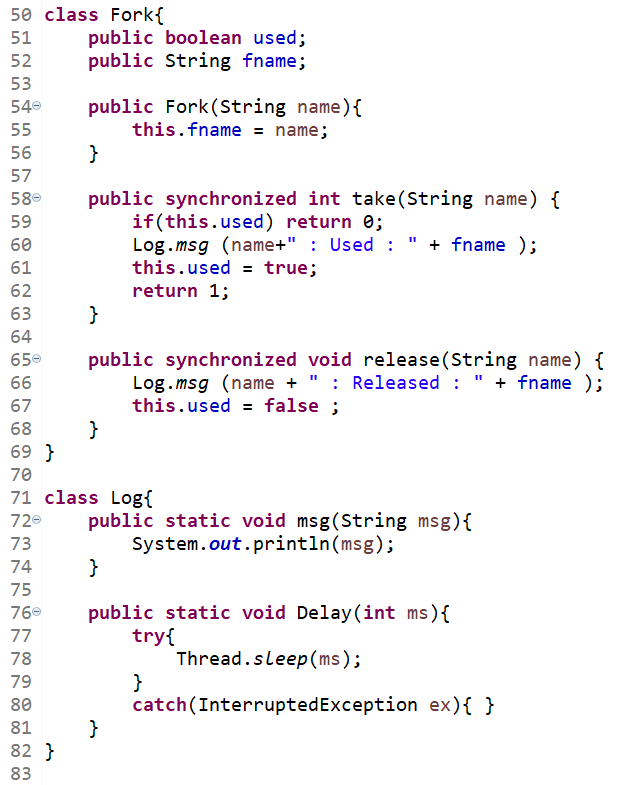


Fork클래스는 포크를 의미한다. 여기서는 두개의 필드를 가진다. boolean자료형의 used 필드다. used는 포크가 철학자에 의해 사용되고 있냐 즉 사용자가 이 포크를 먹는 행위를 하기위해 들고있어나 들고 음식을 먹느냐의 의미이다. 그래서 그럴경우 true이고 그 상태가 아니면 false상태가 된다. String형 fname필드는 포크의 이름을 의미한다.

(코드54~56) Fork클래스의 매개변수 있는 생성자를 구현한다. String형 매개변수를 받아 fname필드를 초기화한다.

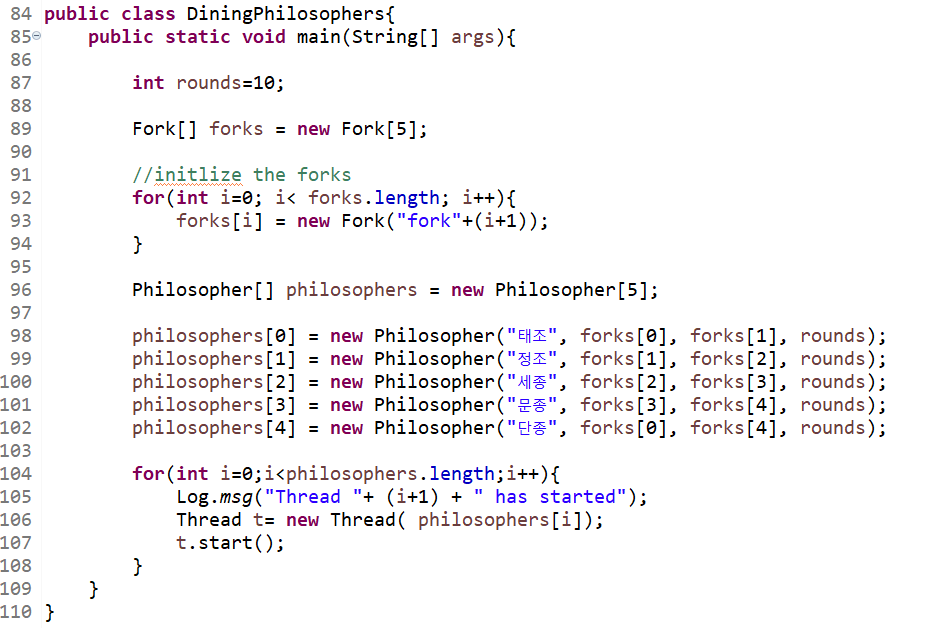
(코드58~63) public이고 int형을 받환하는 take메소드를 구현한다. 이 메소드는 String형 변수를 매개변수로 받는다. take함수는 이 포크를 매개변수로 받은 name이름을 가진 철학자가 사용을 한다는 동작을 의미한다. 그래서 메소드 안에서 used가 true인 경우 0을 리턴한다. 즉 이 메소드에 들어와서 포크가 사용중일경우 실패했다는 의미로 0을 리턴하고 그것이 아니면 used를 true로 만들고 성공의 의미로 1을 리턴한다. 이때 Log클래스에 있는 msg메소드를 사용해 name이름을 가진 철학자가 fname을 가진 포크를 take한다는 것을 출력해준다. 이때 여러쓰레드 중 단 한개의 스레드만 이 함수를 실행할 수 있다는 동기화를 보장하기 위해서 synchronized 키워드를 사용해준다.

(코드65~68) release함수 역시 동기화를 보장하기 위해서 synchronized 키워드를 사용한다. 이함수는 public으로 take메소드와 다르게 void 즉 아무것도 반환하지 않고, String형 변수를 통해 행위를 하는 철학자 이름을 매개변수로 넘겨 받는다. 역시 Log클래스의 msg메소드를 사용해 한 행위를 출력해주고 used필드를 false로 바꾼다.



클래스 Log는 철학자, 포크 클래스에서 일어나는 일들을 출력하는 역할을 한다. 그래서 Log클래스는 두개의 메소드가 구현되있다. msg메소드와 Delay메소드이다. msg메소드는 매개변수로 받은 String형 msg를 출력해주는 역할을 하고 Delay메소드는 매개변수로 받은 정수형변수 ms만큼 Thread.sleep를 통해 sleep해준다. 이때 try블록에 구현이 되있고 예외를 처리하기 위해 catch블록 역시 구현되있다.

여기서 두개의 메소드 모두 public static 으로 구현되있다. 그래서 Log클래스 내에 있는 두개의 메소드를 사용할때 Log클래스의 객체를 생성하지 않고 Log.메소드이름(…)형태로 사용하게 된다.



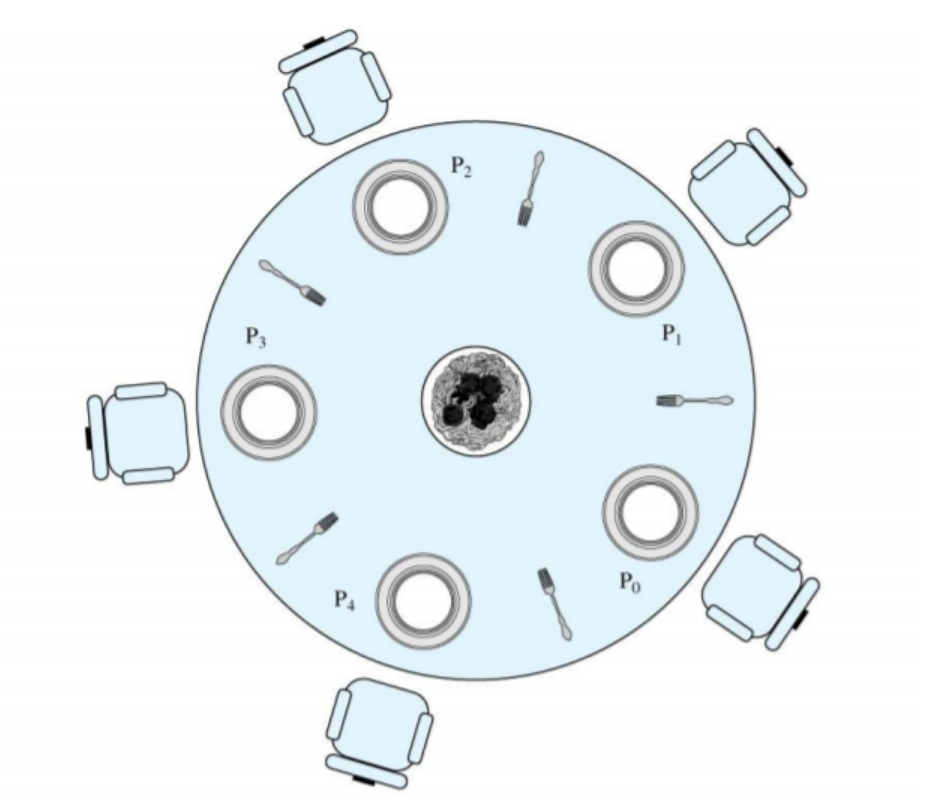
main함수에 rounds를 정해준다. 여기서는 10으로 해준다.

(코드89) 포크 5개가 필요하다. 그래서 Fork클래스 객체 5개를 배열로 만들어준다.

(코드92~94) 포크를 초기화하는 for반복문을 만든다. Forks를 인덱스로 접근해 5개의 객체에 대해서 포크이름

(코드96) 철학자를 의미하는 Philosopher객체 5개를 배열로 만들어야 한다.

(코드98~102) 5개의 철학자 객체를 실제로 생성한다. 객체를 생성할 때 매개변수로 넘겨줘야 할 것들은 철학자의 이름과 왼쪽 포트, 오른쪽포크의 각각의 객체, rounds 수이다. 여기서 철학자의 이름은 조선시대 왕이름으로 대신했다. 그리고 왼쪽, 오른쪽 포트는 아래 그림과 같이 설정해 주었다. 다만 마지막 철학자에게는 왼쪽포크와 오른쪽 포크의 위치를 바꾸어 주었다. 그 이유는 교착생태를 막기위함이다.



**fork5**

**fork4**

**fork3**

**fork2**

**fork1**

(코드104~108) for반복문을 통해 5개의 철학자 객체의 쓰레드를 실행시킨다. 이때 Log클래스의 msg메소드를 활용해 스레드가 실행된 다는 것을 알려준다.