

Homework #6

Homework Title: Thread Synchronization and Cooperation

Student ID: 201724542 Name: 이준영

(Problem 1) Use lock/unlock and signal/await for thread's synchronization and cooperation.

1) Describe the program source with detailed comments.

```
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;
import java.util.concurrent.locks.Condition;
import java.util.concurrent.locks.Lock;
import java.util.concurrent.locks.ReentrantLock;

public class HW6 { // 메인 클래스
    private static Pot honeyPot = new Pot(); // Pot 클래스 변수 생성

    public static void main(String[] args) { // 메인 함수
        System.out.println("Bee Thread\t\tBear Thread\t\tHoney Amount");
        // 출력 포맷을 위한 문장 출력

        ExecutorService executor = Executors.newFixedThreadPool(21);
        // Bee 개수 + Bear 개수 == 21 이므로 그에 맞는 쓰레드 풀 생성

        for (int i = 1; i <= 20; i++) { // 반복문으로 Bee 쓰레드를 쓰레드 풀에 추가
            executor.execute(new Bee(i));
        }

        executor.execute(new Bear()); // Bear 쓰레드를 쓰레드 풀에 추가
    }

    private static class Bee implements Runnable { // Bee 클래스
        int beeNum; // 몇번째로 생성된 벌인지를 나타내는 int 변수

        Bee(int value) { // 생성자 재선언
            beeNum = value; // 입력값을 beeNum 에 저장
        }

        @Override
        public void run() { // run() 함수를 오버라이딩
            try {
                int honey = 1; // 저장할 꿀의 양을 1로 설정
                while(true) {
                    honeyPot.add(honey, beeNum); // 꿀단지에 꿀을 추가하는 메소드 실행
                    Thread.sleep(((int) Math.random() * 10000) % 1900 + 101);
                    // Bee 쓰레드를 100 ~ 2000ms 동안 중지
                }
            } catch (InterruptedException e){
                e.printStackTrace();
            }
        }
    }

    private static class Bear implements Runnable { // Bear 클래스

        @Override
        public void run() { // run() 함수를 오버라이딩
            while(true) {
```

```

        honeyPot.eat(); // 꿀을 다 먹어버리는 메소드 실행
    }
}

private static class Pot { // Pot 클래스
    private static final int H = 10000; // 꿀단지의 용량을 나타내는 int 변수
    private int honeyAmount = 0; // 저장된 꿀의 양을 나타내는 int 변수
    private int lastBee; // 마지막으로 꿀을 넣은 벌을 나타내는 int 변수

    private static Lock lock = new ReentrantLock(); // Lock 변수 생성
    private static Condition Empty = lock.newCondition();
    // 꿀단지가 비었다는 상태를 나타낼 Condition 변수 생성

    private static Condition Full = lock.newCondition();
    // 꿀단지가 다 찼다는 상태를 나타낼 Condition 변수 생성

    public void add(int honey, int beeNum) { // 꿀단지에 꿀을 추가하는 메소드
        lock.lock(); // Lock 을 잠금
        try {
            lastBee = beeNum; // 마지막으로 꿀을 넣은 벌을 저장
            if(honeyAmount == H) { // 만약 현재 꿀의 양이 10000 과 같으면
                Full.signal();
                // 다 찼다고 신호를 보냄
                Empty.await();
                // 곰이 다 먹고 단지가 비었다는 신호가 올 때까지 대기
            }
            honeyAmount += honey; // 꿀단지에 꿀을 추가
            System.out.println(String.format("%12s", beeNum + "th Bee: " + honey) +
                "\t\t\t\t\t" + honeyAmount);
            // 몇번째 벌이 얼마만큼 꿀을 추가했고, 그 후 잔여량이 얼마인지를 모두 출력
        } catch (InterruptedException e){
            e.printStackTrace();
        }
        finally {
            lock.unlock(); // Lock 을 다시 해제
        }
    }

    public void eat() { // 꿀단지에서 곰이 꿀을 먹는 메소드
        lock.lock(); // Lock 을 잠금
        try {
            while(honeyAmount != 0) { // 만약 현재 꿀의 양이 10000 이 아니면
                Full.await(); // 꿀단지에 꿀이 다 찼 때까지 대기
            }
            System.out.println(String.format("%20s", lastBee + "th Bee: Wake UP!!") +
                " Bear: Yum.. Delicious Honey!!");
            // 꿀을 다 먹었다고 출력
            honeyAmount = 0; // 꿀단지를 모두 비움
            Empty.signal(); // 꿀단지가 비었다고 신호를 줌
        } catch (InterruptedException e){
            e.printStackTrace();
        }
        finally {
            lock.unlock(); // Lock 을 해제
        }
    }
}
}

```

2) Capture the outcome generated by your program on the screen.

```
<terminated> HW6 [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_121.jdk/Contents/Home/bin/java (Dec 3, 2018, 11:06:54 AM)
Bee Thread      Bear Thread      Honey Amount
1th Bee: 1      1
3th Bee: 1      2
2th Bee: 1      3
4th Bee: 1      4
5th Bee: 1      5
6th Bee: 1      6
7th Bee: 1      7
8th Bee: 1      8
9th Bee: 1      9
10th Bee: 1     10
11th Bee: 1     11
12th Bee: 1     12
13th Bee: 1     13
14th Bee: 1     14
15th Bee: 1     15
16th Bee: 1     16
17th Bee: 1     17
18th Bee: 1     18
19th Bee: 1     19
20th Bee: 1     20
1th Bee: 1     21
6th Bee: 1     22
3th Bee: 1     23
5th Bee: 1     24
4th Bee: 1     25
8th Bee: 1     26
2th Bee: 1     27
7th Bee: 1     28
11th Bee: 1    29
9th Bee: 1    30
12th Bee: 1    31
13th Bee: 1    32
10th Bee: 1    33
14th Bee: 1    34
<terminated> HW6 [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_121.jdk/Contents/Home/bin/java (Dec 3, 2018, 11:05:47 AM)
10th Bee: 1    9981
10th Bee: 1    9982
4th Bee: 1    9983
3th Bee: 1    9984
9th Bee: 1    9985
19th Bee: 1   9986
1th Bee: 1    9987
15th Bee: 1   9988
12th Bee: 1   9989
7th Bee: 1    9990
13th Bee: 1   9991
17th Bee: 1   9992
6th Bee: 1    9993
16th Bee: 1   9994
20th Bee: 1   9995
2th Bee: 1    9996
11th Bee: 1   9997
8th Bee: 1    9998
5th Bee: 1    9999
14th Bee: 1  10000
3th Bee: Wake UP!! Bear: Yum.. Delicious Honey!!
19th Bee: 1    1
18th Bee: 1    2
1th Bee: 1     3
17th Bee: 1    4
6th Bee: 1     5
15th Bee: 1    6
12th Bee: 1    7
7th Bee: 1     8
13th Bee: 1    9
16th Bee: 1   10
20th Bee: 1   11
11th Bee: 1   12
2th Bee: 1    13
8th Bee: 1    14
```

(Problem 2) Use lock/unlock and notify/wait for thread's synchronization and cooperation.

1) Describe the program source with detailed comments.

```
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;
```

```
public class HW6_2 { // 메인 클래스
    private static Pot honeyPot = new Pot(); // Pot 클래스 변수 생성

    public static void main(String[] args) { // 메인 함수
        System.out.println("Bee Thread\t\tBear Thread\t\tHoney Amount");
        // 포맷에 맞춘 출력을 위한 문장 출력

        ExecutorService executor = Executors.newFixedThreadPool(21);
        // 쓰레드 풀 생성

        for (int i = 1; i <= 20; i++) { // 반복문으로 쓰레드 풀에 Bee 쓰레드 추가
            executor.execute(new Bee(i));
        }

        executor.execute(new Bear()); // 쓰레드 풀에 Bear 쓰레드 추가
        executor.shutdown();
    }

    public static class Bee implements Runnable { // Bee 클래스
        private int beeNum; // 벌의 번호를 저장할 int 변수

        Bee(int value) { // 생성자 재선언
            beeNum = value; // 입력값을 beeNum에 저장
        }

        @Override
        public void run() { // run() 함수를 오버라이딩
            try {
                while(true) {
                    synchronized(honeyPot) { // Pot 쓰레드를 동기화
                        Pot.lastBee = beeNum; // 가장 마지막에 넣은 벌의 번호를 저장
                        if (honeyPot.getHoneyAmount() != Pot.H) { // 만약 현재 꿀의
양이 10000이 아니면
                                honeyPot.add(1, beeNum); // 꿀을 1만큼 더하는 메소드를
실행
                                honeyPot.notify(); // Pot 쓰레드에 알림을 준다
                            }
                        }
                    }
                    Thread.sleep(((int) Math.random() * 10000) % 1900 + 101); // 100 -
2000ms 동안 쓰레드를 재운다
                }
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }

    public static class Bear implements Runnable { // Bear 클래스
        @Override
        public void run() { // run() 함수를 오버라이딩
            try {
                while (true) {
                    synchronized(honeyPot) { // Pot 쓰레드를 동기화
```

않았을 동안

```
while(honeyPot.getHoneyAmount() < Pot.H) // 꿀단지가 가득 차지

        honeyPot.wait(); // Pot 쓰레드의 notify를 기다림
        honeyPot.eat(); // 꿀을 먹는 메소드를 실행
    }
} catch (InterruptedException e) {
    e.printStackTrace();
}
}

private static class Pot { // Pot 클래스

    private static final int H = 10000; // 꿀단지의 용량을 나타내는 int 변수
    private int honeyAmount = 0; // 현재 담긴 꿀의 양을 나타내는 int 변수
    private static int lastBee; // 마지막에 꿀을 넣은 벌을 나타내는 int 변수

    public int getHoneyAmount() { // 현재 담긴 꿀의 양을 반환하는 함수
        return honeyAmount; // honeyAmount를 반환
    }

    public void eat() { // 꿀을 먹는 메소드
        honeyAmount = 0; // 꿀의 양을 0으로 바꿈
        System.out.println(String.format("%20s",lastBee + "th Bee: Wake UP!!") +
            " Bear: Yum.. Delicious Honey!!");
        // 포맷에 맞게 어떤 벌이 꿀을 깨웠는지와 꿀이 꿀을 먹었다는 내용을 출력
    }

    public void add(int honey, int beeNum) { // 꿀을 추가하는 메소드
        if (honey+honeyAmount <= H) honeyAmount += honey; // 꿀을 추가
        System.out.println(String.format("%12s",beeNum + "th Bee: " + honey) +
            "\t\t\t\t\t" + getHoneyAmount());
        // 포맷에 맞게 어떤 벌이 꿀을 얼마만큼 추가했는지를 출력
    }
}
}
```

2) Capture the outcome generated by your program on the screen.

```
<terminated> HW6_2 [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_121.jdk/Contents/Home/bin/java (Dec 3, 2018, 11:10:01 AM)
Bee Thread      Bear Thread      Honey Amount
1th Bee: 1      1
20th Bee: 1     2
19th Bee: 1     3
18th Bee: 1     4
17th Bee: 1     5
16th Bee: 1     6
15th Bee: 1     7
14th Bee: 1     8
13th Bee: 1     9
12th Bee: 1    10
11th Bee: 1    11
10th Bee: 1    12
9th Bee: 1     13
8th Bee: 1     14
7th Bee: 1     15
6th Bee: 1     16
5th Bee: 1     17
4th Bee: 1     18
3th Bee: 1     19
2th Bee: 1     20
1th Bee: 1     21
14th Bee: 1    22
11th Bee: 1    23
15th Bee: 1    24
16th Bee: 1    25
18th Bee: 1    26
17th Bee: 1    27
20th Bee: 1    28
19th Bee: 1    29
2th Bee: 1     30
3th Bee: 1     31
5th Bee: 1     32
4th Bee: 1     33
6th Bee: 1     34
15th Bee: 1    9989
20th Bee: 1    9990
6th Bee: 1     9991
5th Bee: 1     9992
2th Bee: 1     9993
3th Bee: 1     9994
10th Bee: 1    9995
17th Bee: 1    9996
9th Bee: 1     9997
14th Bee: 1    9998
7th Bee: 1     9999
11th Bee: 1   10000
3th Bee: Wake UP!! Bear: Yum.. Delicious Honey!!
16th Bee: 1     1
12th Bee: 1     2
18th Bee: 1     3
13th Bee: 1     4
4th Bee: 1     5
19th Bee: 1     6
10th Bee: 1     7
14th Bee: 1     8
9th Bee: 1     9
17th Bee: 1    10
7th Bee: 1    11
8th Bee: 1    12
15th Bee: 1    13
5th Bee: 1    14
16th Bee: 1    15
13th Bee: 1    16
19th Bee: 1    17
4th Bee: 1    18
18th Bee: 1    19
12th Bee: 1    20
3th Bee: 1     21
2th Bee: 1     22
```