객체지향언어와 실습

실습 13주차

엄진영 교수님

담당조교 박선희
동국대학교 컴퓨터공학과
seonhuibag1228@gmail.com
010-4065-3024

실습문제 1 타자게임 <산성비>

- 단일 쓰레드를 사용하여 타자게임 산성비 게임을 구현해보자. wordGenerator라는 쓰레드가 ArrayList에 2초마다 단어를 하나씩 추가한다. 그리고 사용자가 단어를 입력하면 ArrayList에서 일치하는 단어를 삭제하도록 작성하시오.

실습문제 1-1 Source1 클래스

- **Source1 클래스의 필드와 메인 메소드**

```
public class Source1
{
    ArrayList<String> words = new ArrayList<String>(); //게임 판에 보여져야할 단어들
    String[] data = {"자바","객체","자구","컴구","기초프로그래밍","이산구조","인컴","패턴"};
    wordGenerator wg = new wordGenerator();

    public static void main(String args[])
    {
        Source1 game = new Source1();
        빈칸을 채우시오  // 단어를 생성하는 쓰레드를 실행시킨다.

        ArrayList<String> gameBoard = game.words;
        while(true) {
            System.out.println(gameBoard);
            String prompt = ">>";
            System.out.print(prompt);

            Scanner s = new Scanner(System.in);
            String input = s.nextLine().trim();
            빈칸을 채우시오  // 입력받은 단어를 words에서 찾는다.
            if(index!=-1) { // 찾으면
                빈칸을 채우시오  // words에서 해당 단어를 제거한다.
            }
        }
    } // main
```

https://docs.oracle.com/javase/8/docs/api/java/util/ArrayList.html

Tip. ArrayList 의 메소드를 활용하시오.

실습문제 1-1 Source1 클래스

- **Source1 클래스의 wordGenerator Thread**

```
class 빈칸을 채우시오 {
    int interval = 2 * 1000; // 2초
    public void run() {
        while(true) {
            // 1. 배열 data의 값 중 하나를 임의로 선택해서
            빈칸을 채우시오

            // 2. words에 저장한다.
            빈칸을 채우시오

            // 3. 2초(interval) 동안 쉰다.
            빈칸을 채우시오
        }
    }
}
```

실습문제 1 문제 설명

- 이전의 코드의 빈칸을 완성하여 아래와 같은 결과화면을 출력하시오.
  - 결과 화면

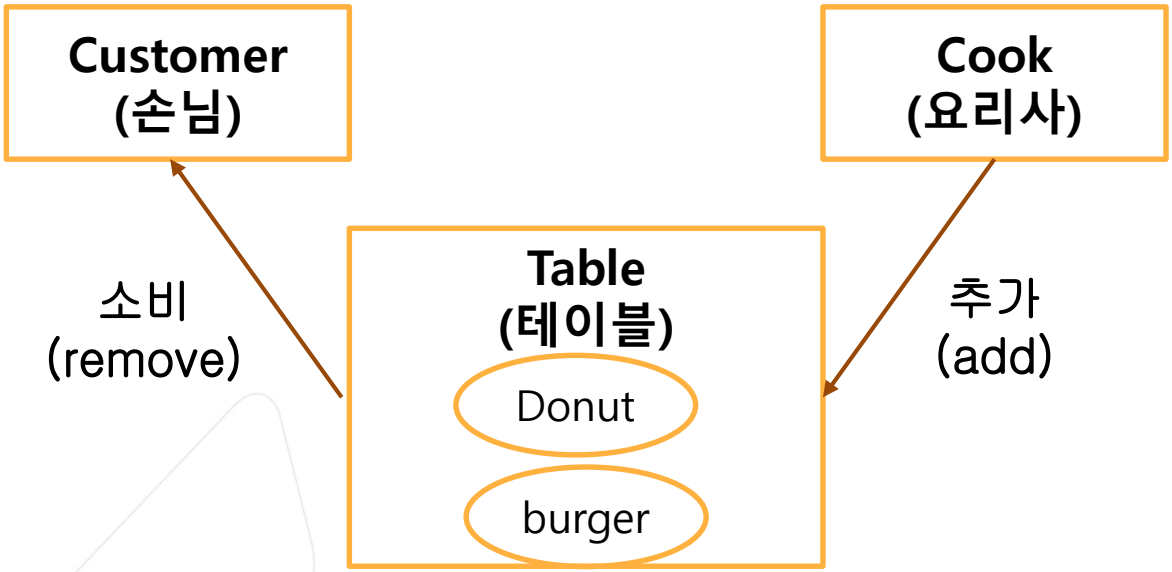
```
[]
>>
[자바]
>>자바
[패턴, 자구]
>>자구
[패턴, 인컴]
>>패턴
[인컴, 객체]
>>객체
[인컴, 기초프로그래밍]
>>인컴
[기초프로그래밍, 자바]
>>기초프로그래밍
[자바, 패턴, 객체, 이산구조, 객체, 객체]
>>
```

```
[]
>>
[객체]
>>객체
[기초프로그래밍, 이산구조]
>>기초프로그래밍
[이산구조, 컴구]
>>이산수학
[이산구조, 컴구, 이산구조, 자바]
>>자바
```

  - 산출 파일 이름 : Source1.java

실습문제 2 멀티 쓰레드

- 식당에서 음식을 만들어서 테이블에 추가하는 요리사와 테이블의 음식을 소비하는 손님을 쓰레드로 구현해보자.



- 산출 파일 이름 : Source2.java

실습문제 2-1 Table 클래스

- Table 클래스의 필드는 다음과 같다.

```
class Table{
    String[] dishNames= {"donut","donut","burger"}; //음식 이름
    final int MAX_FOOD=6; //테이블에 세팅될 최대 음식 수
    private ArrayList<String> dishes=new ArrayList<String>(); //테이블에 세팅된 음식접시
```

- Table 클래스는 다음과 같은 메소드를 지니고 있다.
  - add(String dish) : 테이블에 세팅 할 수 있는 최대 음식 수 보다 적게 세팅이 되어 있으면 dish('donut' or 'burger') 을 추가한다. 추가하게 되면 "Dishes : [<현재 세팅되어 있는 음식들>] " 을 출력한다.
    - » 출력화면 : `Dishes : [burger, burger, burger, donut]`
  - remove(String dishNames) : 테이블에 음식이 아무것도 없으면 "<해당손님이름> is waiting."을 출력하고 0.5초마다 음식이 추가되어있는지 확인하면서 기다린다. 만약 테이블에 음식이 있으면 원하는 음식(dishNames)과 일치하는 요리 한 접시를 테이블에서 제거한다.

실습문제 2-2 Customer 클래스

- Customer 클래스는 Runnable 인터페이스 구현을 통해 쓰레드 클래스로 구현한다. 이 클래스의 필드와 생성자는 다음과 같다. Food 는 고객이 주문한 음식이다.(음식 주문은 한 사람당 한 개만 한다고 가정 한다.)

```
private Table table;
private String food;

Customer(Table table, String food){
    this.table=table;
    this.food=food;
}
```

- Customer 클래스는 0.01초를 기다리고 주문한 음식(food)를 remove 하며, 만약에 제거가 되면 "<고객이름> ate a <주문한 음식 이름>"을 출력 하고, 만약 제거가 안되었다면 "<고객이름> failed to eat." 을 출력한다. 그리고 이를 무한 반복한다.
  - Tip. 고객 이름은 현재 쓰레드의 getName() 메소드로 얻을 수 있다.

실습문제 2-3 Cook 클래스

- Cook 클래스는 Runnable 인터페이스 구현을 통해 쓰레드 클래스로 구현한다. 이 클래스의 필드와 생성자는 다음과 같다.

```
private Table table;

Cook(Table table){
    this.table=table;
}
```

- Cook 클래스는 table의 dishNames에서 랜덤하게 한 음식을 add 한다. 그리고 이 행동을 0.1초 쉬고 다시 반복한다.

실습문제 2 멀티 쓰레드

- 여러 쓰레드가 하나의 테이블을 공유하고 있다. 따라서 요리사가 음식을 놓으려는 중에 손님이 음식을 가져가려 하면 안되고, 첫 번째 손님이 남은 음식을 가져가는 도중에 다른 손님이 먼저 음식을 가져가서 있지도 않은 테이블에서 제거하려고 하면 안되게끔(예외가 발생하면 안됨) 결과화면과 같은 결과를 출력 하시오.

- 메인 화면

```
public static void main(String[] args) {
    // TODO Auto-generated method stub
    Table table=new Table(); //여러 쓰레드가 공유하는 객체 테이블
    new Thread(new Cook(table),"Cooker").start();
    new Thread(new Customer(table,"donut"),"client1").start();
    new Thread(new Customer(table,"burger"),"client2").start();

    try{
        Thread.sleep(5000);
    }catch(InterruptedException e) {
        System.exit(0);
    }
}
```

- 결과 화면

```
Dishes : [donut]
client2 failed to eat.:(
client1 ate a donut
client2 is waiting.
client2 is waiting.
client2 is waiting.
client2 is waiting.
client2 is waiting.
client2 is waiting.
client2 is waiting.
```

- 산출 파일 이름 : Source2.java

실습문제 3 wait() & notify()

- 실습문제 2번에서는 여러 쓰레드에서 사용하는 공유 데이터를 보호하였다. 하지만 결과화면을 보면 새로운 문제가 보인다.

```
Dishes : [donut]
client2 failed to eat.:(
client1 ate a donut
client2 is waiting.
client2 is waiting.
client2 is waiting.
client2 is waiting.
client2 is waiting.
client2 is waiting.
client2 is waiting.
```

?

- 왜 요리사는 음식을 더 이상 추가하지 않고 손님을 계속 기다리게 하는 것일까?

실습문제 3 wait() & notify()

- 왜 요리사는 음식을 더 이상 추가하지 않고 손님을 계속 기다리게 하는 것일까?

```
Dishes : [donut]
client2 failed to eat.:(
client1 ate a donut
client2 is waiting.
client2 is waiting.
client2 is waiting.
client2 is waiting.
client2 is waiting.
client2 is waiting.
client2 is waiting.
```

- 손님 쓰레드가 테이블 객체의 lock을 쥐고 기다리기 때문이다. 요리사가 음식을 새로 추가하려 해도 테이블 객체의 lock을 얻을 수 없어서 추가할 수가 없다.

실습문제 3 add(), remove() 와 Customer 클래스

- 이전의 상황을 개선하기 위해 add(), remove()와 Customer 클래스를 다음과 같이 수정하시오.

- Customer 클래스는 0.01초를 기다리고 주문한 음식을(food)를 remove 한다. 제거가 되면 "<고객이름> ate a <주문한 음식 이름>" 을 출력한다. 그리고 이를 무한 반복한다.

- add(String dish) : 만약 테이블의 최대 세팅 그릇 수보다 크면 0.5초 동안 기다린다. 이를 반복하는데 만약 자리가 나면 해당 접시를 추가하고 "Dishes : [<현재 세팅되어 있는 음식들>] " 을 출력한다.

- remove(String dishName) : 만약 테이블에 세팅이 안 되어있다면 0.5초 기다린다. 이를 반복하는데 만약 세팅이 되어있다면 주문한 음식을 찾는다. 만약 주문한 음식이 없다면 또 0.5초 기다리며, 반복하고, 주문한 음식이 생기면 제거한다.

실습문제 3 wait() & notify()

- wait()와 notify()를 사용하여 문제를 해결하고 2번과 같은 main 함수 내용을 실행하였을 때 아래와 같은 결과화면을 얻으시오.
  - Tip. 손님이 원하는 음식이 없으면 wait()으로 lock 을 풀고 기다리다가 음식이 추가되면 notify()로 통보를 받고 다시 lock을 얻어서 나머지 작업을 진행할 수 있게 한다.
  - 결과 화면

```
Dishes : [donut]
client2 is waiting.     <- 원하는 음식이 없으면 손님이 기다린다.
client1 ate a donut     <- 원하는 음식이 있기에 먹는다.
client2 is waiting.     <- 원하는 음식이 없으면 기다린다.
Dishes : [burger]       <- 음식을 추가한다.
client1 is waiting.     <- 원하는 음식이 없으면 기다린다.
client2 ate a burger    <- 원하는 음식이 있기에 먹는다.
client1 is waiting.
client2 is waiting.
Dishes : [donut]
client1 ate a donut
```

  - 산출 파일 이름 : Source3.java

제출 시 유의사항

- 기한 : 2019년 6월 2일 (일) 23:59까지
- 제출 파일 형식
  - 제출 시, *.java파일과 보고서를 압축하여
    [n주차]_[학번]_[이름].zip 파일로 압축하여 제출
    ex > 1주차_2016xxxxxx_박선희.zip
  - 보고서는 소스코드와 실행화면을 캡쳐하고, 간단히 분석하여 제출
  - 코드 시작 부분에 주석을 이용하여 과, 학번, 이름 및 문제번호를 적을 것
- 기타 문의: seonhuibag1228@gmail.com 으로 문의