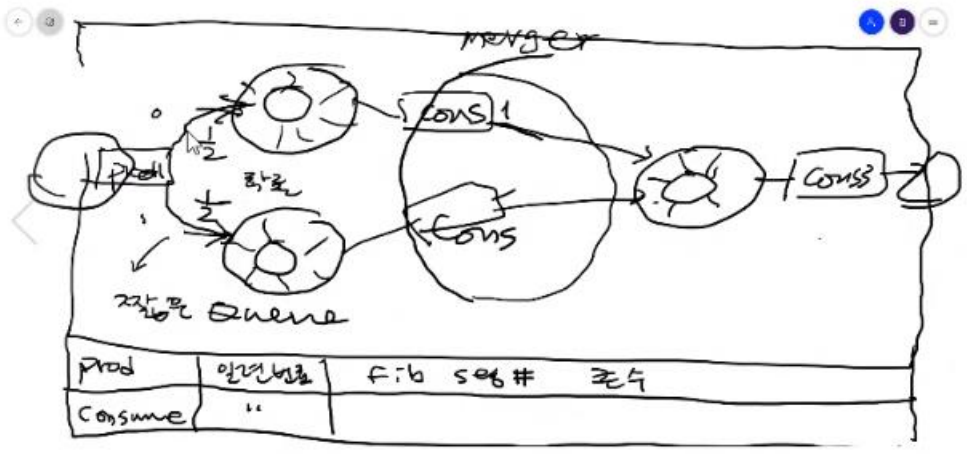
제목	3개의 Circular Buffer 만들기	과제번호	5
학과	독일어과	학번	201602173
이름	윤영택	제출일	20200611

1. 문제 정의

OOPY2020의 CircularBufferSimulationFX를 수정해서 주어진 조건을 충족하여 다음과 같은 형태의 3개의 Circular-Buffer를 구현한다.



2-1. 문제 해결 방법

여러 작은 문제들이 있지만, 위 프로그램을 구성하기 위해 해결해야 할 문제는 크게 3가지로 나눌 수 있다.

1) 3개의 큐를 만들어 알맞은 위치에 배치한다.

기존 CircularBuffer객체인 cirbuf를 3개 만든다. 이때 BufferShape클래스의 생성자를 수정하여 두 개의 매개변수를 설정한다(double x, double y). 이는 각각의 큐를 원하는 위치에 배치하기 위함이다.

2) 쓰레드가 작동하면서 큐에 데이터가 write/read되는 것을 구현한다. 이때 Monitor개념을 구현하여 동기화가 이루어지게 한다.

Producer 1개와 Consumer 3개가 필요하다. 그런데 기존 프로젝트(CircularBufferSimulationFX)처럼 Consumer를 만들면 Consumer3이 데이터를 읽지 못한다. 따라서 Consumer1과 Consumer2에서 데이터를 read하고 읽은 데이터를 3번 큐에 write하게 만든다. 이때 .read()와 .write()에 Synchronized 키워드가 사용되었음에 유의하여 임계구역 설정을 위한 임의의 전역변수를 생성하지 않도록 한다.

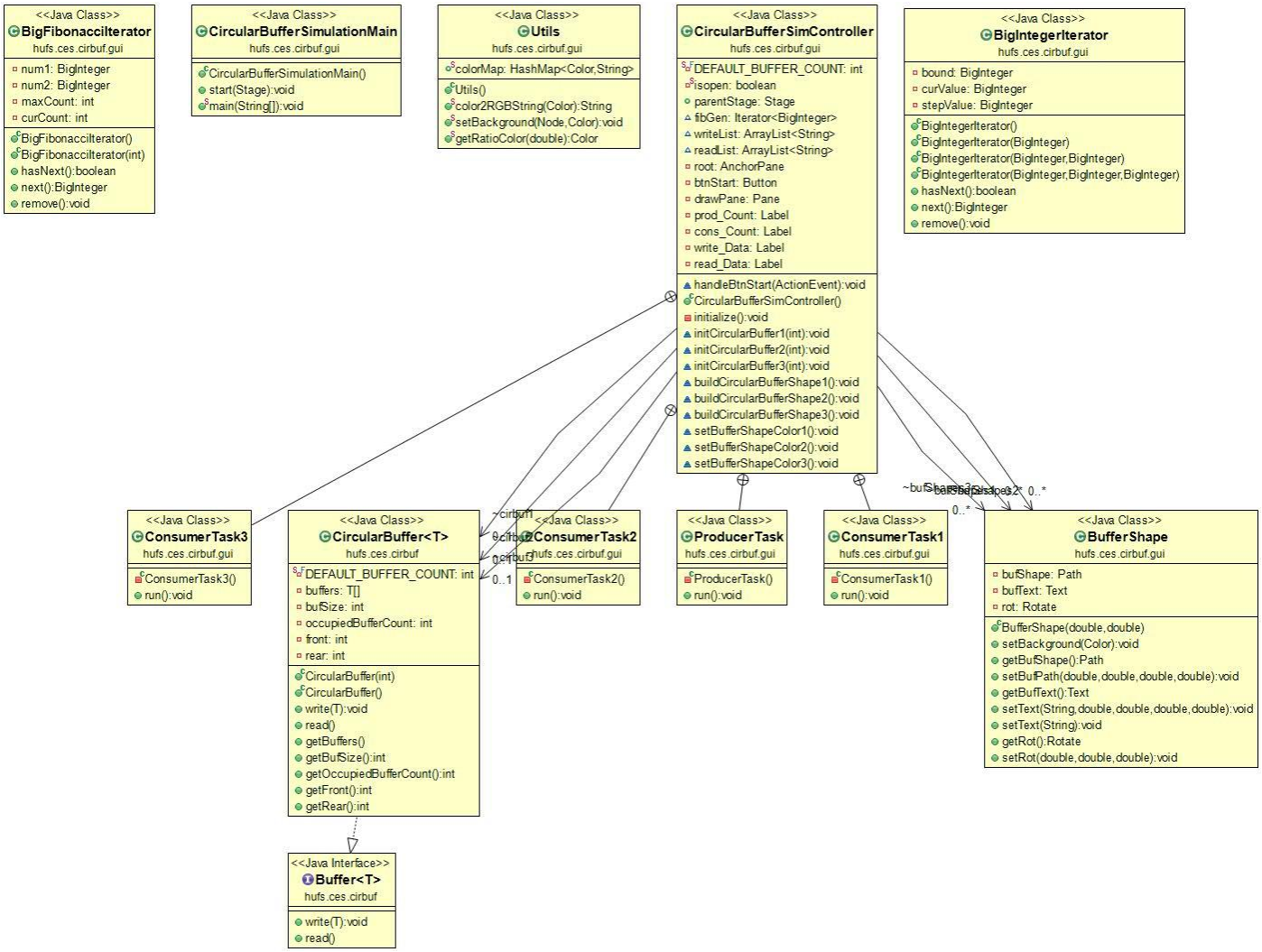
3) 요구사항에 따라 하단의 HBOX에 데이터가 보이게 한다.

Scene Builder를 이용해서 HBOX 안에 또 2개의 HBOX를 삽입하고 그 안에 필요한만큼 Label을 삽입한다. 하나의 Label엔 하나의 String만 Set할 수 있으므로, 10개의 데이터를 보여주기 위해 ArrayList를 이용해서 하나의 String을 만들어 Label에 set한다.

2-2. 설계 결과(프로그램 구성도, 필요 함수, 클래스 설명)

1) 프로그램 구성도

- 이미지 크기의 문제로 다음 페이지에



2) 필요함수, 클래스 설명

1) initCircularBuffer()
큐 객체를 생성하고 큐의 모양을 잡아주는 **buildCircularBufferShape1()**메소드를 호출한다.

2) buildCircularBufferShape()
원형 큐의 모양을 직접적으로 만들어준다. 원래는 입력한 사이즈에 따라 큐의 모양이 상이하지만 여기선 사이즈를 10으로 고정했다.

3) setBufferShapeColor()
큐에 색을 칠해준다.

4) public class CircularBuffer<T> implements Buffer<T>
큐를 구현한 클래스이다. Synchronized키워드를 사용하여 동기화가 이루어지게 했다.

4) public class BigIntegerIterator implements Iterator<BigInteger>
Iterator를 구현하여 만든 Generator이다. 정수를 생성하는 역할을 한다.

5) private class ProducerTask implement Runnable
Generator 객체의 .next()메소드를 호출해서 데이터를 생성하고 1/2 의 확률로 1 번 큐 또는 2 번 큐에 write 한다. 데이터를 write 할 때, Producer 의 생성카운트를 1 씩 증가하고 int 형의 데이터를 String 으로 형 변환하여 ArrayList 에 담아준다. 이때 ArrayList 의 size 가 10 이 넘어가면 0 번 인덱스의 String 은 사라지고 새로운 String 이 10 번 인덱스에 담긴다. 그 후에 ArrayList 를 다시 String 으로 변환하여 intGen(Sequential)값을 보여주는 Label 에 set 해준다.
6) private class ConsumerTask1 implements Runnable, private class ConsumerTask2 implements Runnable
1 번 큐와 2 번 큐에서 데이터를 읽고 3 번 큐에 write 한다.

7) private class ConsumerTask3 implements Runnable
최종 Producer이다. write동작 없이 3번 큐를 read만 한다. 5번과 비슷하게 read한 데이터를 String으로 변환하여 Label에 set해준다.

3. 결론 및 소감

3.1 과제 해결 중 생긴 문제와 해결 과정

문제 1: 1번 큐와 2번 큐가 작동을 하지 않고 3번 큐만 작동했다.

-> 큐를 생성할 때 메소드 내부에서 변수명을 잘못 선언했기 때문에 발생했다. 즉 1번 큐와 2번 큐는 형태는 만들어졌으나, 1번 큐와 2번 큐에서 일어나야 할 동작(Producer의 write, Consumer1과 Consumer2의 read/write)이 모두 3번 큐에서 일어났다.

문제 2: 모든 큐가 모두 하나의 Producer에서 데이터를 받아오는 문제가 발생했다.

-> 발생한 문제의 예시는 다음과 같다.
<console>
con1 reads 0
con2 reads 1
con2 reads 2

con3 reads 3 ...

이런 식으로 Producer에서 생성한 데이터를 Consumer1과 Consumer2, 그리고 Consumer3이 나눠서 read하는 문제가 발생했다. 물론 눈에 보이는 큐의 동작도 엉망이었다. Consumer1과 Consuemr2과 데이터를 read만 하고 3번 큐에 write하지 않았기 때문에 3번 큐가 제대로 데이터를 read하지 못했던 것이었다. 물론 Consumer3에서도 3번 큐가 아닌 1번 큐와 2번 큐에서 데이터를 read하도록 코드를 짜서 복합적으로 문제가 발생했다. 코드를 수정하니 모두 해결되었다.

문제 3: 상호 배제를 구현하기 위해 전역 변수를 사용했다가 read/write동작이 제대로 이루어지지 않았다.

-> 이 경우 Synchronized키워드에 대한 이해가 부족했기 때문에 발생했다. 자바엔 쓰레드 간의 동기화를 지원하는 Synchronized 키워드가 존재하는데, CircularBuffer 클래스에서 이것이 구현되어 있는 줄 모르고 열심히 Controller에서 전역변수를 사용해 동기화를 하려고 했다. 그냥 애초에 그럴 필요가 없던 것이었다.

문제 4: Producer에서 생성된 데이터를 어떻게 1/2확률로 1번 큐와 2번 큐에 write해줄 것인가?
-> 최대한 간단한 식으로 1/2확률로 분배하고자 했다. 처음엔 java.util.Random()을 import한 후 배열에 임의의 값을 뽑아내는 식으로 구현했다. 하지만 그렇게 하니 조건문의 길이가 너무 길어졌다. 구글링을 좀 더 해서 다른 방법을 찾아봤다. 결국 if(new java.util.Random().nextInt(2)==0)와 같이 Random객체를 생성한 후 .nextInt()메소드를 호출하여 코드의 길이를 줄였다.

3.2 배운 점 & 소감

현재 운영체제 과목을 수강하고 있는데, 해당 과목에서 배운 Thread synchronization, 즉 Mutual Exclusion, Critical Section, Progress의 개념을 이해하는 데 큰 도움이 되었다. 다만, Synchronized라는 파워풀한 키워드를 넘겨짚고 여러 알고리즘을 적용하려 했다가 실패했다. 사실 해당 과목에서 배운 것보다 이번 과제를 하면서 좀 더 위 개념들에 대해 더 깊게 이해할 수 있었다. 하지만 아쉬운 점도 있었다. 자바의 객체지향적인 특징은 잘 살리지 못했다. 인스턴스 메소드를 좀 더 잘 다듬었다면 쓸 데 없이 중복으로 메소드를 만들 일이 없었는데, 내 실력이 부족한 탓에 그러지 못했다. 다시 말해, '우아함'이 부족했다. 이것은 코드를 짜는 내내 마주칠 문제이기에 좀 더 연습하고 공부하여 실력을 향상시키는 수밖에 없을 것 같다.