제목	2 개를 동시에 보여주는 Digital Clock	과제번호	1
학과	독일어과	학번	201602173
이름	윤영택	제출일	20200425

1. 문제 정의

TwoDIgitalClock Project 를 참고로 하되, 출력 화면에 날짜 두 개와 시, 분, 초도 각각 2 개씩 표시되도록 한다. 이때, 각각의 Timing 은 TimeLine 과 TimerThread 를 사용하여 구현한다.

2-1. 문제 해결 방법

1) 타이머 메소드를 추가로 만든다.

기존에 구현되어 있던 DigitalClock Project 의 DigitalClockPane.java 와 DigitalClockPaneThread.java 를 Compare with 기능을 이용해 비교하여 각 방식의 Timing 이 어떻게 이루어지는지 확인한다. Thread 방식의 경우엔, 각 쓰레드가 독립적으로 작동하게 하기 위해서 Thread 를 상속받는 또 하나의 클래스를 만든다. 그 후, format 메소드의 파라미터를 변경하여 시간을 표시하도록 한다.

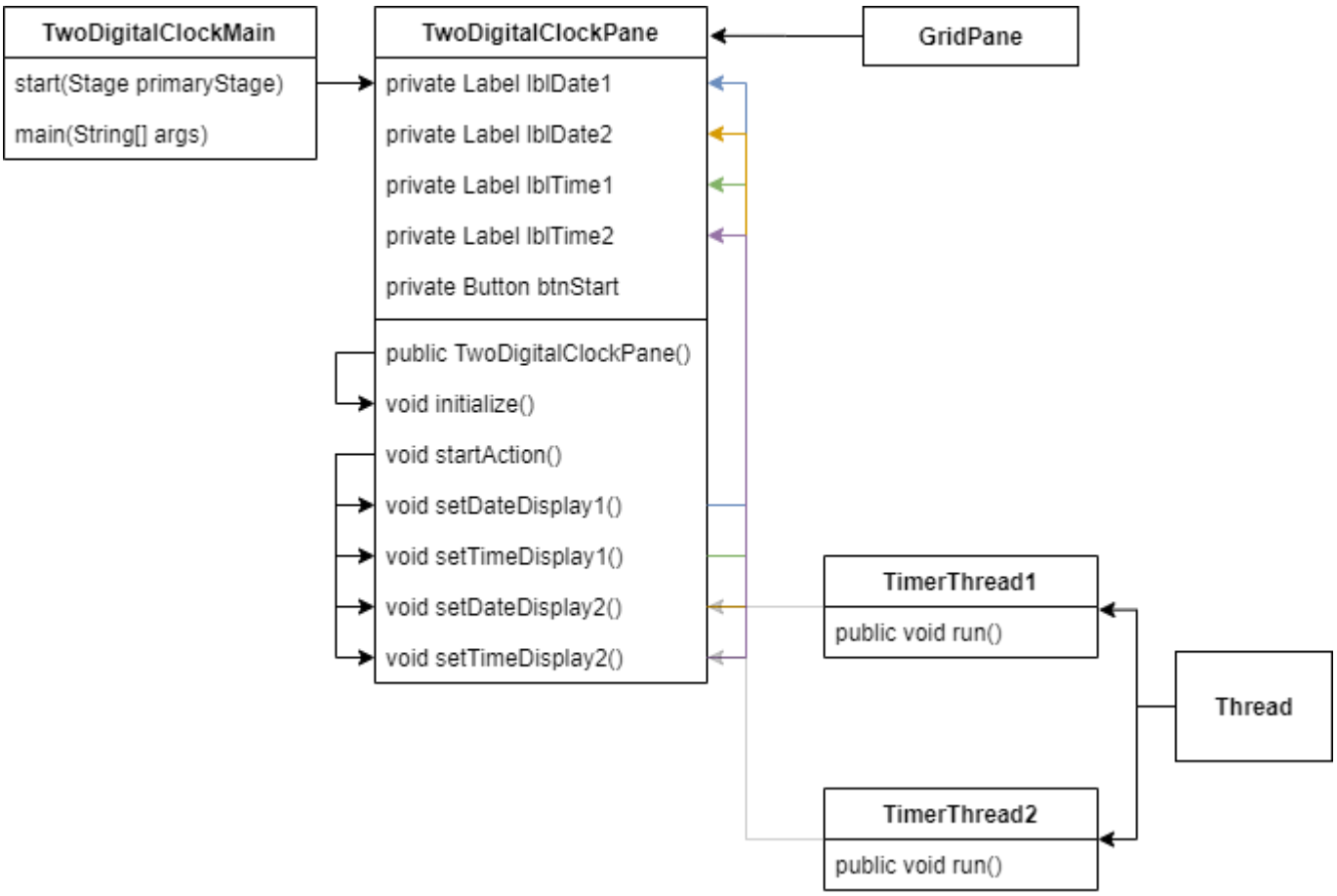
2) 각 아이템(label, node)의 디자인과 시계(pane)의 레이아웃을 변경한다.

BorderPane클래스는 아이템을 고정적인 위치(Top, Left, Center, Right, Bottom)에 배치하기 때문에, 만들고자 하는 시계의 레이아웃에는 어울리지 않는다. 물론 또 다른 Boderpane객체를 생성해서 BorderPane 안에 넣을 수 있지만, 코드가 너무 복잡해지기 때문에 BorderPane 대신 GridPane클래스를 상속받도록 한다. 이때 TwoDigitalClock-Main의 코드도 수정해야 한다. root의 타입을 BorderPane에서 GridPane으로 변경하고 필요한 클래스를 import한다. 다만 GridPane은 스크린의 사이즈에 상관없이 아이템의 크기가 고정적이기 때문에, 반응형 디자인을 만들기 위해 이벤트를 처리하는 메소드를 추가했다. 또한, 각 아이템의 속성값을 변경함으로써 디자인을 변경했다.

3) 1)과 2)를 종합하여 문제를 해결한다.

2-2. 설계 결과(프로그램 구성도, 필요 함수, 클래스 설명)

1) 프로그램 구성도



2) 필요함수, 클래스 설명

1) initialize

시계의 레이아웃과 디자인이 결정된다. 생성자가 호출되면서 이 메소드를 호출한다.
이때, 이벤트가 발생(스크린 크기 변경)하면 heightProperty.addListener 와 .widthProperty.addListener 를 호출하여 Height 값과 Width 값을 변경시킨다. 이때 파라미터로 주어지는 ChangeListener 라는 익명클래스를 람다식으로 간단하게 작성했다.

2) startAction

내부적으로 시계의 동작이 이루어지는 메소드이다. Timing 을 구현하는 방식에 따라 상이한 타입의 객체를 생성한다.

3) setDateDisplay1, setDateDisplay2, setTimeDisplay1, setTimeDisplay2

시간을 보여주는 메소드들이다. setDateDisplay1 과 setTimeDisplay1 은 TimeLine 방식을 이용하며, setDateDisplay2 와 setTimeDisplay2 는 TimerThread 을 사용한다.

4) TimerThread1, TimerThread2

Thread 클래스를 상속받아 스레드를 생성한다. 이때 Thread 클래스에서 제공되는 run 메소드를 오버라이딩해서 사용한다. 스레드가 실행되면서 각각 setDateDisplay2 와 setTimeDisplay2 를 호출하고 0.1 초의 대기시간을 가진다.

3. 결론 및 소감

3.1 과제 해결 중 생긴 문제와 해결 과정

문제: Pane의 레이아웃을 변경하기 어려웠다.

→ 처음에는 계속 BorderPane을 사용해서 레이아웃을 바꿔보려 했지만, 계속 실패했다. 또 다른 BorderPane을 두 개 더 생성해서 메인 BorderPane의 Left와 Right에 넣는 방법도 있었지만, 에러를 줄이기 위해 최대한 코드를 적게 수정하는 방향으로 가고 싶었다. 그런 이유로 GridPane을 이용하기로 했다. GridPane을 이용해서 코드를 작성하니 간결하게 코드를 작성할 수 있었지만 또 다른 문제가 발생했다.

BorderPane의 경우 위치가 고정적이기 때문에 각 아이템(label)의 속성값만 변경하면 반응형 디자인을 만들 수 있었지만, GridPane은 스크린이 늘어나도 아이템이 크기가 그대로였다. 이 문제를 해결하기 위해 많은 외국사이트를 찾아봤지만, 대부분 FXML을 사용한 코드였거나 아직 내 수준에서 이해하기 어려운 코드들뿐이었다. 그래서 이벤트 처리 메소드를 통해 해결하기로 했다. 계속 찾아보다가 마침내 관련된 영상이 있었다. 영상을 참조하여 문제를 해결했다.

3.2 배운 점 & 소감

다른 사소한 문제들의 경우엔 한국어로 설명되어 있는 페이지들이 많아서 괜찮았지만, Pane과 관련된 문제들은 한국어는 물론 영어로 작성된 페이지들도 많지 않았다(내가 해결하고자 하는 문제에 한해선). 한국어로 구글링해선 앞으로의 과제들을 해결하기엔 역부족이라는 것을 뼈저리게 느꼈다. API도 온통 영어라 읽기가 버거워 제대로 못 읽은 부분도 많았다. 이번 기회에 그동안 등한시했던 영어를 더 열심히 공부하겠다는 다짐을 했다. 또한, 컴퓨터 구조 수업시간에 배웠던 쓰레드에 대해서 더욱 깊게 이해하고 직접 구현해볼 수 있는 기회가 생겨서 좋았다. 그리고 디자인&레이아웃의 경우에 CSS와 흡사한 점이 많아서 흥미롭게 과제를 했다.