**2020 Spring OOP Assignment Report**

과제 번호 : 5

학번 : 20190084

이름 : 권민재

Povis ID : mzg00

**명예서약 (Honor Code)**

나는 이 프로그래밍 과제를 다른 사람의 부적절한 도움 없이 완수하였습니다.

I completed this programming task without the improper help of others.

1. **프로그램 개요**
   * 이 프로그램은 C++와 QT Framework를 이용해 간단한 시계 어플리케이션을 구현한 것이다.
   * 빌드된 파일을 실행하고, 버튼을 눌러서 프로그램을 이용할 수 있다.
   * mainwindow.cpp, mainwindow.cpp에 시계, 스톱워치, 타이머가 구현되어 있고, main.cpp에 main()이 존재한다.
   * mainwindow.h에는 디자인, audio.qrc에는 외부 오디오 리소스 파일의 정보가 저장되어 있다. 별도로, ASSN5.pro가 qt creator 프로젝트 파일로 존재한다.
2. **프로그램의 구조 및 알고리즘**

* Class[[1]](#footnote-1)

|  |  |
| --- | --- |
| **MainWindow** | |
| 메인 윈도우 클래스 | |
| **private slots** | |
| **멤버 함수** | |
| **Clock Mode** | |
| void event\_clock() | 시계 이벤트 시 실행되는 함수 |
| **Stopwatch Mode** | |
| void event\_stopwatch() | 스탑워치 이벤트 시 실행되는 함수 |
| void on\_stopwatch\_btn\_start\_clicked() | 스탑워치 시작 버튼이 클릭될 때 실행되는 함수 |
| void on\_stopwatch\_btn\_stop\_clicked() | 스탑워치 중지 버튼이 클릭될 때 실행되는 함수 |
| void on\_stopwatch\_btn\_reset\_clicked() | 스탑워치 리셋 버튼이 클릭될 때 실행되는 함수 |
| **Timer Mode** | |
| void event\_timer() | 타이머 이벤트 시 실행되는 함수 |
| void on\_timer\_btn\_hour\_up\_clicked() | 타이머 시간 증가 버튼이 클릭될 때 실행되는 함수 |
| void on\_timer\_btn\_minute\_up\_clicked() | 타이머 분 증가 버튼이 클릭될 때 실행되는 함수 |
| void on\_timer\_btn\_sec\_up\_clicked() | 타이머 초 증가 버튼이 클릭될 때 실행되는 함수 |
| void on\_timer\_btn\_hour\_down\_clicked() | 타이머 시간 감소 버튼이 클릭될 때 실행되는 함수 |
| void on\_timer\_btn\_minute\_down\_clicked() | 타이머 분 감소 버튼이 클릭될 때 실행되는 함수 |
| void on\_timer\_btn\_sec\_down\_clicked() | 타이머 초 감소 버튼이 클릭될 때 실행되는 함수 |
| void on\_timer\_start\_clicked() | 타이머 시작 버튼이 클릭될 때 실행되는 함수 |
| void on\_timer\_stop\_clicked() | 타이머 중지 버튼이 클릭될 때 실행되는 함수 |
| void on\_timer\_reset\_clicked() | 타이머 리셋 버튼이 클릭될 때 실행되는 함수 |
| **private** | |
| **멤버 변수** | |
| Ui::MainWindow \*ui | 메인 윈도우 ui |
| QTimer s\_timer | 초단위 (Second) 타이머 |
| QTimer m\_timer | 밀리초단위 (Millisecond) 타이머 |
| **Stopwatch Mode** | |
| QTime stopwatch\_time{0,0,0,0} | 스탑워치 시간 |
| bool stopwatch\_is\_started | 스탑워치의 시작 여부 |
| **Timer Mode** | |
| QTime timer\_time{0,0,0,0} | 타이머 시간 |
| bool timer\_is\_started | 타이머의 시작 여부 |
| QSoundEffect timer\_alram | 타이머 종료 시 효과음 |
| **멤버 함수** | |
| **Timer Mode** | |
| QString timer\_label\_update(QString q, int d, int min, int max) | 최대, 최솟값을 반영하여 d 만큼 더한 숫자 문자열을 반환하는 함수. |
| void set\_timer\_label(QString h, QString m, QString s) | 타이머 라벨에 표시되는 시간을 임의의 문자열로 설정하는 함수 |
| void timer\_done() | 타이머 종료시 트리거되는 함수 |

* 알고리즘
  + Watch mode
    - Watch mode에서, event\_clock이 s\_timer에 의해 1초마다 실행되게 된다. 실행되었을 경우에, event\_clock은 현재 시간을 가져오고, 시간과 날짜를 라벨에 출력한다. 이를 통해 1초에 한번씩 시간과 날짜를 출력함으로써 시계의 기능을 수행한다.
  + Stopwatch mode
    - Stopwatch mode에서는 event\_stopwatch가 m\_timer에 의해 10 밀리초마다 실행되게 된다.
    - 실행되었을 경우, event\_stopwatch는 우선 스탑워치가 실행중인지 그 여부를 우선 체크한다. 만약 실행 중인 상태가 아니라면 즉시 종료하고, 아니라면 이후 과정을 수행한다.
    - 스톱워치는 QTime 오브젝트 stopwatch\_time으로 시간을 잰다. 만약, 현재까지 잰 stopwatch\_time의 밀리초가 60 \* 60000 밀리초, 즉 1시간이 되었다면 실행을 중지하고 초기화시킨다. 현재까지 잰 시간은 msecsTo 메서드를 통해 알 수 있다.
    - 아니라면, stopwatch\_time을 10 밀리초 증가시키고, 라벨에 표시하여 스톱워치를 실행시킨다.
    - 시작 버튼을 눌렀을 경우, 시작 여부를 표시해주고, 현재 문자열에서 fromString 메서드를 이용해 시간을 추출하여 stopwatch\_time을 그 시간으로 초기화한다.
    - 중지 버튼을 눌렀을 경우, 시작 여부를 false로 마킹해준다.
    - 리셋 버튼을 눌렀을 경우, stopwatch\_time을 0밀리초로 초기화하고, 시작 여부를 false로 마킹하고, 표시되는 라벨 또한 초기화시킨다.
  + Timer mode
    - Timer mode에서는 event\_timer가 s\_timer에 의해 1초에 한번씩 실행되게 된다.
    - 실행되었을 경우, event\_timer는 우선 타이머가 실행중인지 그 여부를 체크하여 실행중인 상태가 아니라면 즉시 event\_timer를 즉시 종료한다.
    - 타이머는 QTime 오브젝트, timer\_time으로 시간을 측정한다.
    - 만약 timer\_time의 누적 시간이 0초라면, timer\_done을 실행시키고 종료한다. 타이머가 종료되지 않은 상태라면, event\_timer에서 timer\_time을 1초 감소시키고, 그 시간으로 라벨을 업데이트 하여 타이머의 역할을 수행한다.
    - timer\_done에서는 효과음을 재생시키고, 중지 버튼을 눌렀을 때 실행되는 함수와 리셋 버튼을 눌렀을 때 실행하는 함수를 호출하여 타이머를 중지 및 리셋한다.
    - **화면에 0이 표시된 직후는, 엄연히 0.99999…초 정도 타이머가 남은 상황이기 때문에, 화면에 0이 표시되고 1초 후에 소리가 재생된다.**
    - 각 항목의 증가/감소 버튼을 눌렀을 경우, 함수 timer\_label\_update가 각 최소/최댓값에 맞는 문자열을 반환하고, 이 문자열로 라벨을 업데이트 한다.

1. **토론 및 개선**
   * + Qt를 이용하여 c++에서 라이브러리를 이용하는 방법에 대해 알아보고, event-driven 방식으로 프로그래밍 할 수 있음을 알았다.
     + 간단한 프로그램이기 때문에 별도의 파일 분리를 하지 않았지만, 차후 유지 보수를 생각해서 스톱워치, 타이머, 시계 별로 ui 파일을 분리하고 class를 분리하는 방향으로 개선할 필요가 있다.
     + Slot에서, 타임아웃이 일어날 때 마다 항상 시계, 스톱워치, 타이머의 이벤트 함수가 실행되도록 구현하였는데, 필요에 따라서 connect/disconnect를 동적으로 하게 하여 필요할때만 connect하여 실행되도록 하면 성능을 더 개선할 수 있을 것이다.
2. **참고 문헌**
   * + Qt Docs (Qt Core)
   * https://doc.qt.io/qt-5/qtcore-module.html

1. 중점적인 부분만 기록되어 있다. [↑](#footnote-ref-1)