**2018 산학협력프로젝트2 일반반(2037)**

**<치킨 타이머 for 편돌이>**

**7조 최종 설계 문서**



**2018.12.06**

**201611180 강지율**

**201611202 문혜린**

**201611244 한상규**

**목차**

1. 동작환경
2. 개발도구
3. 구성 파일 설명
   1. Java 파일
      1. MainActivity.java
      2. FF.java
      3. FFAdapter.java
      4. QueueAdapter.java
      5. GDialog.java
      6. GAdapter.java
      7. MsDialog.java
      8. MsAdpater.java
   2. Xml 파일및 Dialog
      1. activity\_main.xml
      2. row.xml
      3. queue\_row.xml
      4. AlertDialog.xml
      5. css\_dialog\_layou.xml t
      6. css\_row.xml
      7. ms\_dialog\_layou.xml t
      8. ms\_row.xml
4. 전체순서도
5. 타이머 등록 순서도
   1. 스피너에서 등록
   2. 대기큐에서 등록
   3. 중복체크
   4. 최대 타이머가 가득 찼는지
   5. 최대 타이머 개수 입력
   6. 대기큐에 등록
   7. 대기큐에서 삭제
   8. 타이머가 한 개일 경우 뷰 설정
6. 타이머 실행 순서도
   1. 타이머 시작버튼
   2. 타이머 쓰레드로 동작시키기
   3. 알람음 울리기
7. 타이머 삭제 순서도
8. 편의점 / 제품 변경 순서도
   1. 편의점 등록 순서도
   2. 편의점 삭제 순서도
   3. 제품 등록 순서도
   4. 제품 삭제 순서도
9. 기능에 대한 추가설명
   1. 데이터 초기화
   2. 등록 가능한 타이머 개수, 등록시 중복체크, 대기큐 관리
   3. 리스트뷰 화면에 타이머 보여주기
   4. 타이머 시작
   5. 타이머 삭제
10. **동작환경 : 안드로이드OS (마시멜로 이상)**
11. **개발도구**

****

안드로이드 스튜디오 3.1.4

1. **구성 파일 설명**
   1. **Java 파일**
      1. **MainActivity.java**

|  |  |
| --- | --- |
| Spinner spinner1 | 편의점을 선택하는 스피너 |
| Spinner spinner2 | ff 제품을 선택하는 스피너 |
| static HashMap<String, HashMap<String, ArrayList<Integer>>> c | ff 제품에 대한 정보를 저장하고 있는 변수 |
| ListView listView | 타이머를 표시해줄 리스트뷰 |
| static ArrayList<FF> registedList | 현재 등록된 타이머 정보들 |
| static ArrayList<FF> queue | 대기 큐에 등록된 타이머 리스트 |
| String conName | 타이머 등록시 FF클래스 객체 생성자에 넘겨줄 편의점 이름 변수 |
| String FFName | 타이머 등록시 FF클래스 객체 생성자에 넘겨줄 FF 제품 이름 변수 |
| ArrayList<Integer> times | 타이머 등록시 FF클래스 객체 생성자에 넘겨줄 전자레인지, 튀김기 시간 변수 |
| static FFAdapter arrayAdapter | 타이머 등록시 FF클래스 객체 생성자에 넘겨줄 listView의 어댑터 |
| static int max | 최대 등록 가능한 타이머 개수 |
| static ArrayList<FF> queue | 대기 큐에 등록된 타이머 리스트 |
| QueueAdapter queueadaper | 대기큐에 들어간 리스트를 출력해주는 recyclerView의 어댑터 |
| RecyclerView recyclerView | 대기큐에 들어간 리스트를 출력해주는 recyclerView |
| LinearLayoutManger layoutManager | recyclerView을 수평으로 출력해주는데 사용하기 위한 LinearLayoutManger |
| AlertDialog.Builder builder | 최대 등록 타이머를 설정할 다이얼로그 |
| boolean isLock | 연속해서 등록되는 상황을 막기 위한 변수 |
| ArrayList<FF> registedList4pause | 현재 타이머의 상태를 일시적으로 저장할 수 있는 변수 |
| static MsDialog msDialog; | 제품을 변경하는 기능을 수행하는 Dialog |
| static HashMap<String, ArrayList<Integer>> msList; | MsDialog에 보여줄 제품에 대한 정보를 가지고 있는 변수 |
| static MsAdapter msAdapter; | MsDialog에 Listview에 데이터를 연결시켜줄 어댑터 |
| private Button startCssChange; | GDialog를 호출시키는 버튼 |
| static ArrayList<String> cssList; | GDialog에 보여줄 편의점 이름을 저장하는 변수 |
| static GDialog gDialog; | 편의점을 변경하는 기능을 수행하는 Dialog |
| static GAdapter gAdapter; | GDialog에 그리드뷰와 데이터를 연결시켜주기 위한 어댑터 |
| static ArrayAdapter arrayAdapter1; | 편의점 선택 spinner를 위한 어댑터 |
| static ArrayAdapter arrayAdapter2; | 제품 선택 spinner를 위한 어댑터 |

* + 1. **FF.java**

|  |  |
| --- | --- |
| String company | 편의점 이름 |
| String name | 제품 이름 |
| ArrayList<Integer> time | 타이머가 돌아야할 시간을 저장하는 변수 |
| ArrayList<Integer> time\_p | 타이머가 흐른 시간을 저장하는 변수 |
| boolean isActive | 현재 타이머가 동작하고 있는지 알려주는 변수 |
| public MediaPlayer mediaPlayer | 알람음을 울리게 해주는 필드 |

* + 1. **FFAdapter.java**

|  |  |
| --- | --- |
| List<FF> m\_FF | registedList의 아이템들을 그대로 받아오기 위한 변수 |
| Context context | MainActivity에 대한 정보를 가져오기 위한 context |
| TextView fName | ff 제품 이름이 표시되는 텍스트뷰 |
| Progressbar mTime | 전자레인지 시간 프로그래스바 |
| Progressbar fTime | 튀김기 시간 프로그래스바 |
| Button startBtn | 타이머 시작 버튼 |
| Button stopbtn | 타이머 삭제 버튼 |
| TimeTask tt | 프로그래스바의 애니메이션을 보여주기 위한 객체 |
| final boolean[] chkTimer | 현재 타이머가 동작하고 있는지 나타내는 변수 |
| int num | 프로그레스바 몇 초마다 업데이트 할 것인지 저장하는 변수 |
| final ProgressBar[] progressBarArrayList | 여러개의 프로그래스바를 한번에 저장하고 있는 배열 |
| final int[] count | 현재 어떤 프로그래스바가 동작하고 있는지 알려주는 변수 |
| ActiveTimeNum | 완료된 타이머가 몇개인지 저장하는 변수 |

* + 1. **QueueAdapter.java**

|  |  |
| --- | --- |
| List<FF> m\_FF | queueList의 아이템들을 그대로 받아오기 위한 변수 |
| Context context | MainActivity에 대한 정보를 가져오기 위한 context |
| TextView fName | ff 제품 이름이 표시되는 텍스트뷰 |

* + 1. **GDialog**

|  |  |
| --- | --- |
| private EditText ed; | 추가할 편의점 이름을 입력하는 editText |
| private Button addCS; | 편의점 추가버튼 |
| private Button closeBt; | 다이얼로그 종료버튼 |
| private GridView gView; | 편의점 이름이 출력될 그리드뷰 |

* + 1. **GAdpater**

|  |  |
| --- | --- |
| Button delCssBtn; | 편의점 삭제하기 버튼 |
| TextView csname; | 편의점 이름을 출력하기 위한 TextView |
| ImageView csBack; | 제품을 수정할 편의점을 선택하기 위한 버튼 |
|  |  |

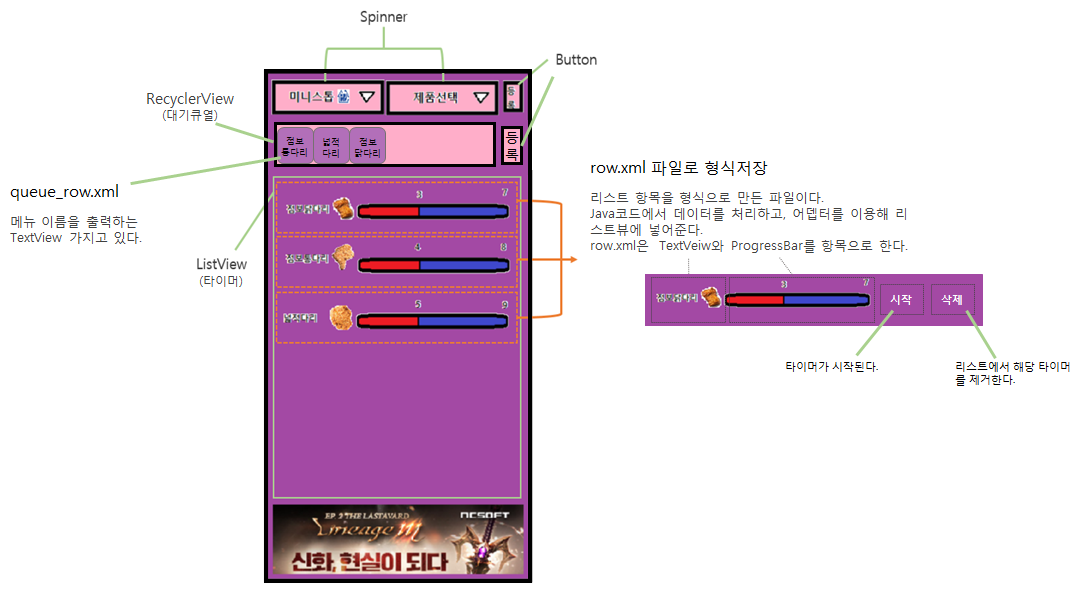
* + 1. **MsDialog**

|  |  |
| --- | --- |
| private TextView title; | 어떤 편의점 제품을 수정할 것인지 알려주는 Textview |
| private EditText m\_name; | 제품 이름을 입력하는 EditText |
| private EditText m1; | 제품의 첫번째 조리시간을 입력하는 EditText |
| private EditText m2; | 제품의 두번째 조리시간을 입력하는 EditText |
| private Button addBtn; | 제품 추가하기 버튼 |
| private ListView listView; | 제품을 정보를 보여줄 ListView |
| private String csName; | 현재 수정하고 있는 편의점 이름을 저장하는 변수 |

* + 1. **MsAdapter**

|  |  |
| --- | --- |
| Button delMenuBtn; | 메뉴를 삭제하는 버튼 |
| String csName; | 현재 수정중인 편의점 이름을 저장하는 변수 |
| HashMap<String, ArrayList<Integer>> msList; | 편의점의 제품정보를 저장하고 있는 변수 |
| ArrayList<String> keyset; | 제품정보를 ArrayList로 변환하기 위한 변수 |

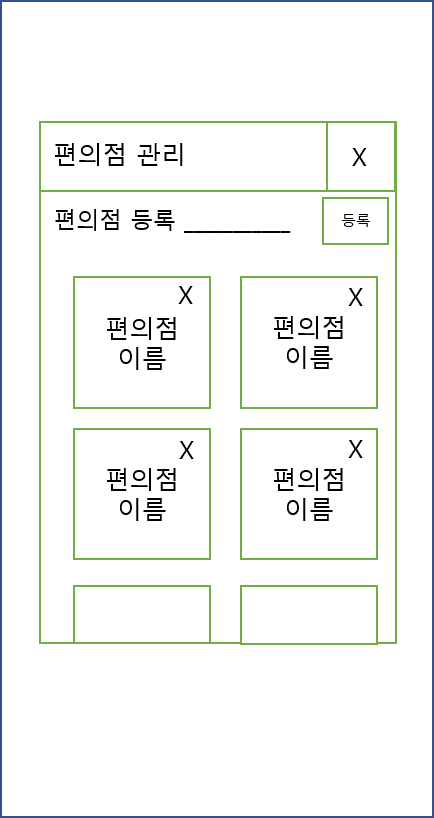
* 1. **Xml 파일 및 Dialog**
     1. **activity\_main.xml**
     2. **row.xml**
     3. **queue\_row.xml**



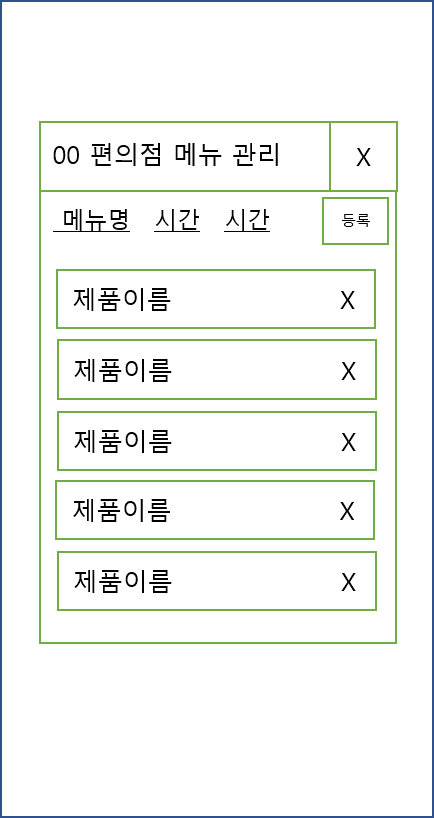
* + 1. **AlertDialog**



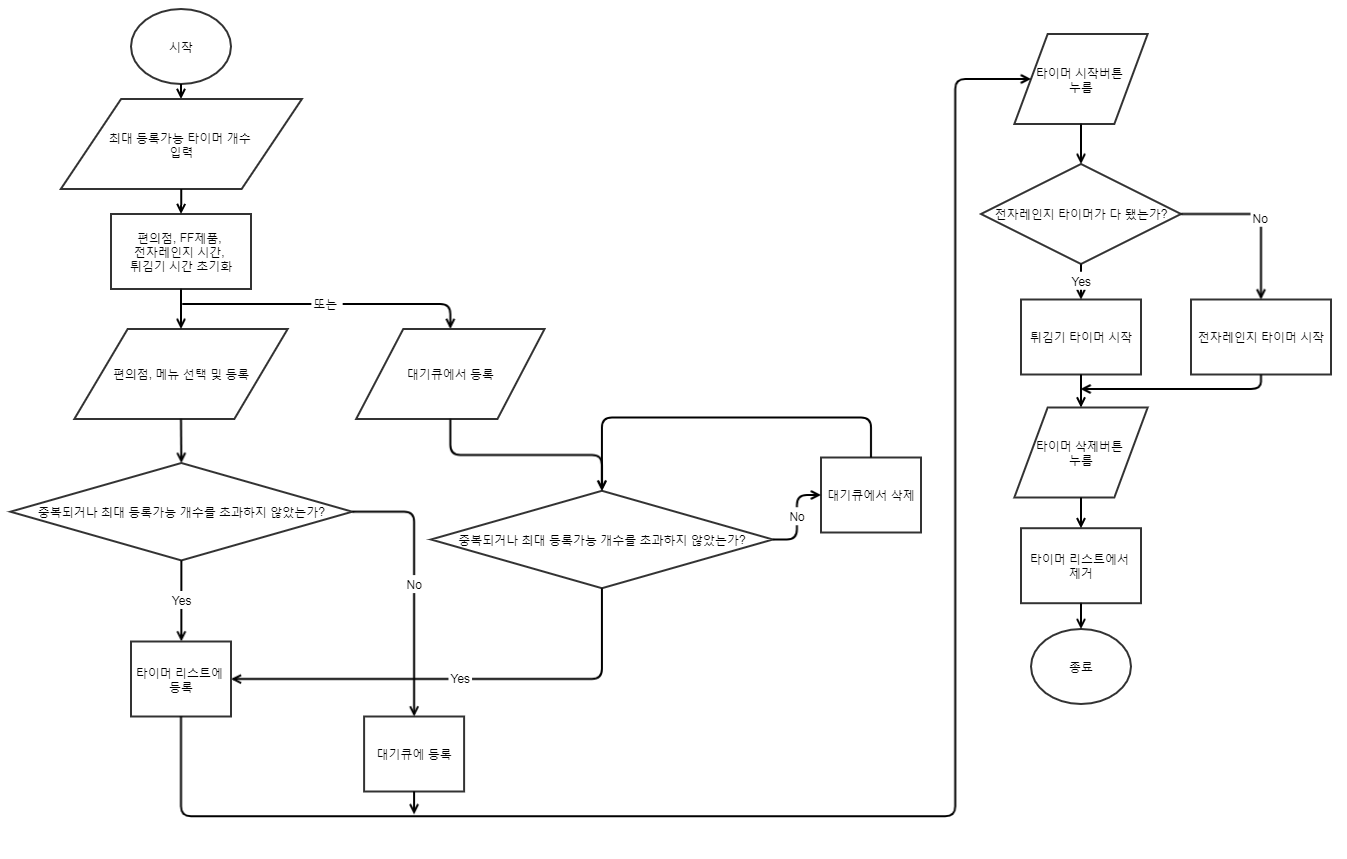
* + 1. **css\_dialog\_layout.xml**
    2. **css\_row.xml**

****

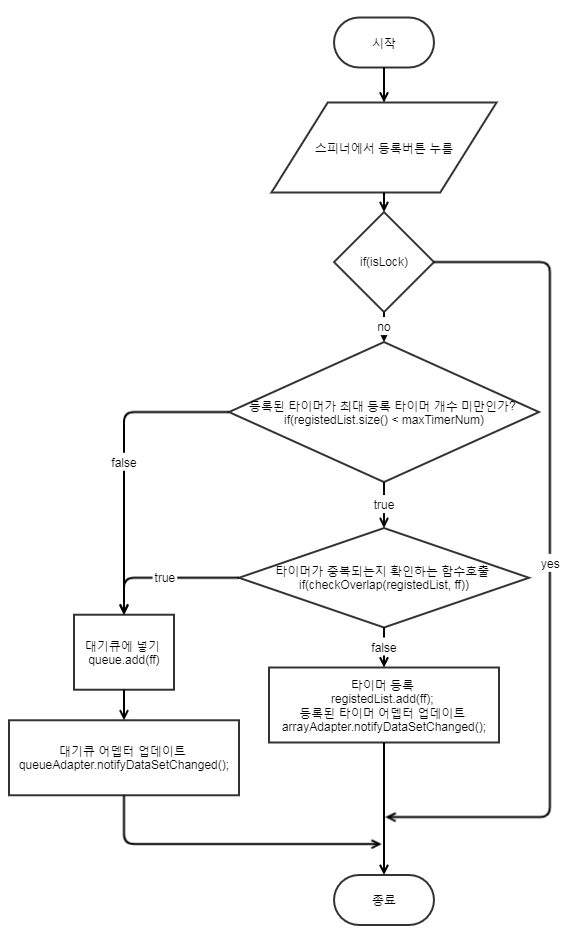
* + 1. **ms\_dialog\_layout.xml**
    2. **ms\_row.xml**

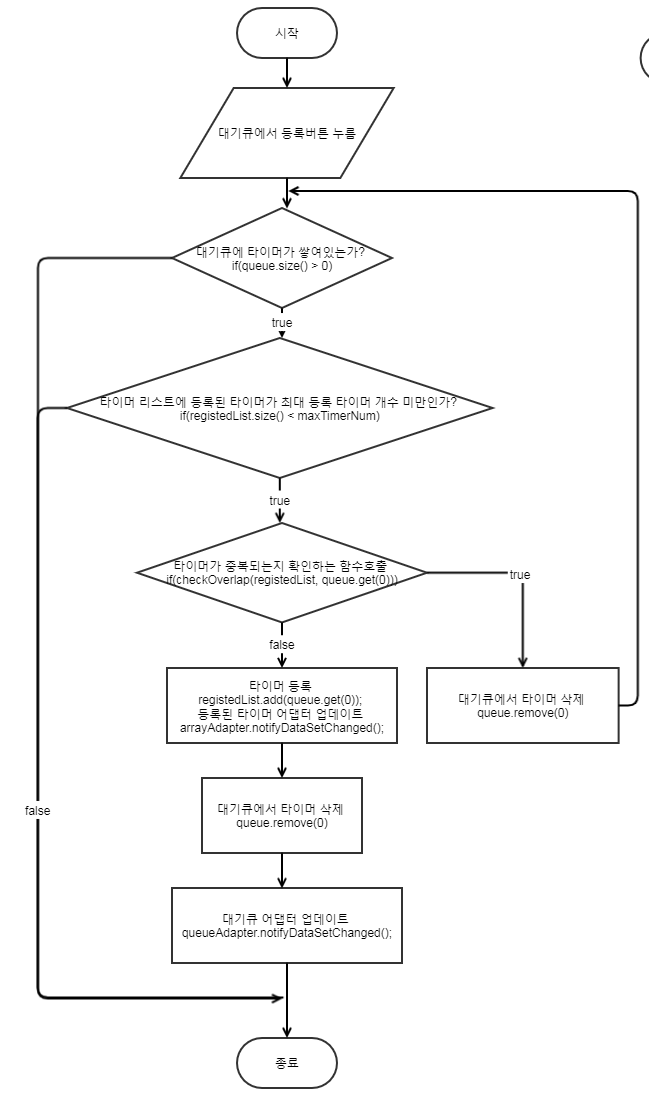
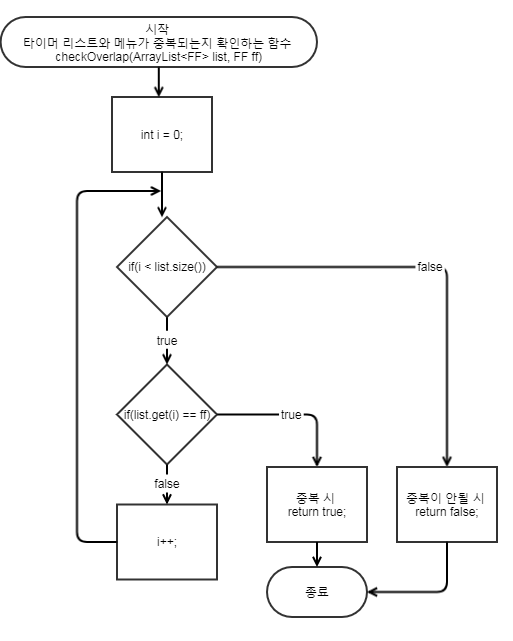


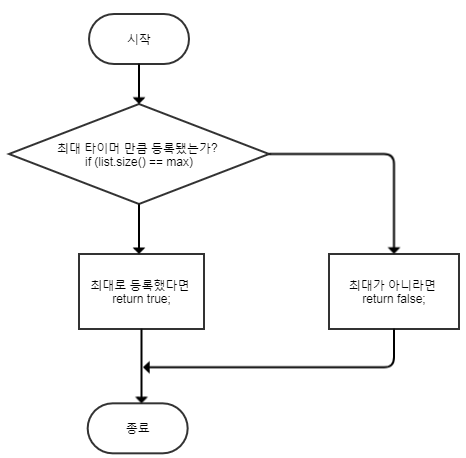
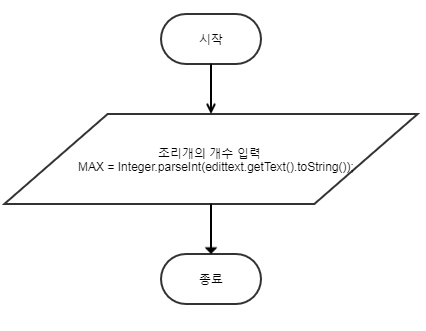
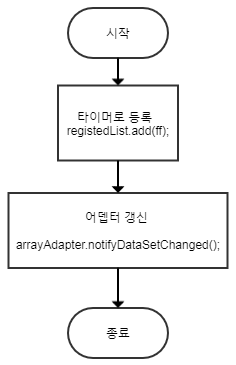
1. **전체순서도**

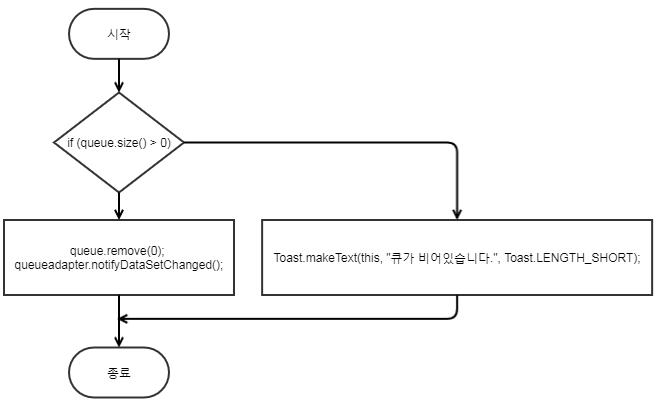
****

1. **타이머 등록 순서도(timercheck)** 
   1. **스피너에서 등록 전체흐름 - MainActivity.java에서 호출**

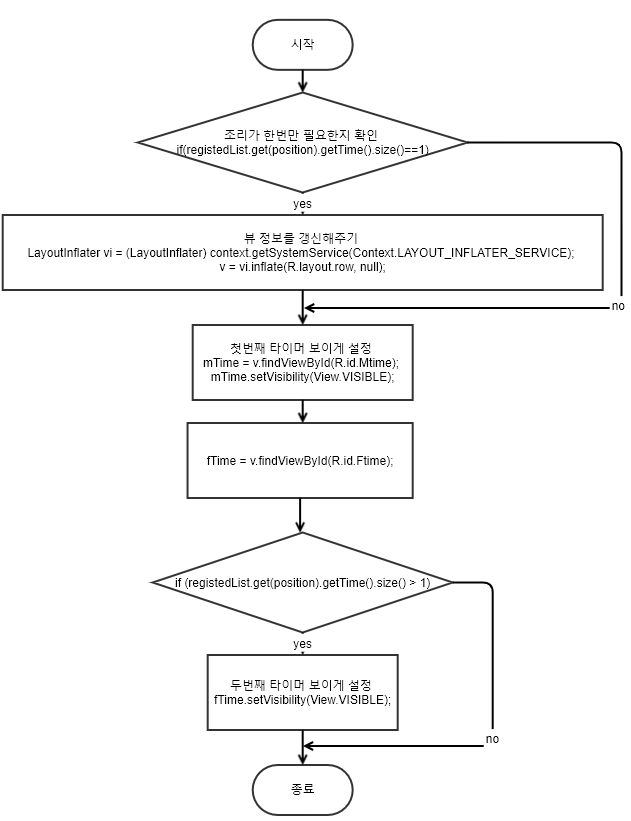
****

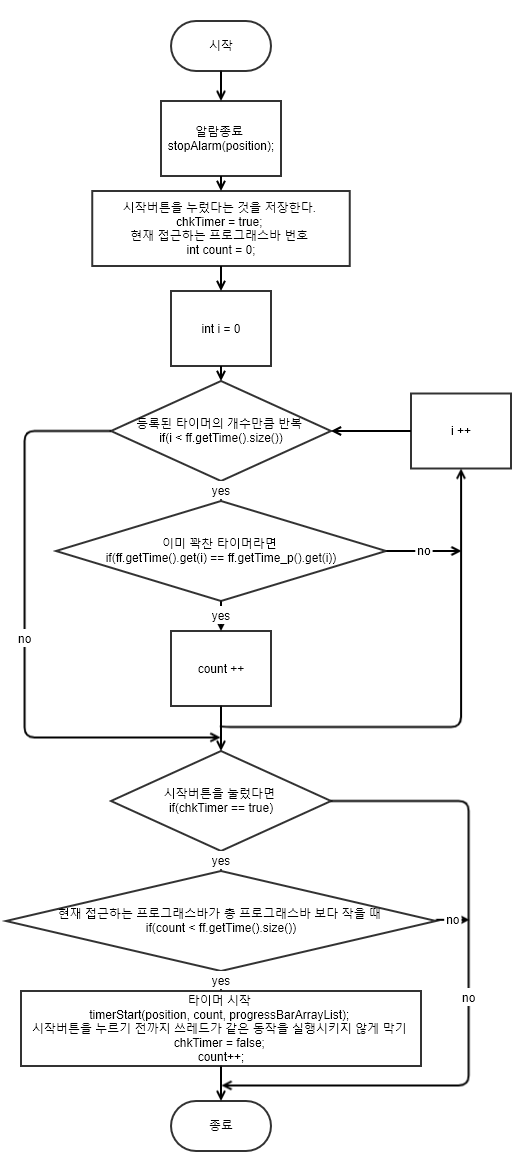
* 1. **대기큐에서 등록 전체흐름 - MainActivity.java에서 호출**
  2. **중복체크(checkOverlap) - MainActivity.java에서 호출**

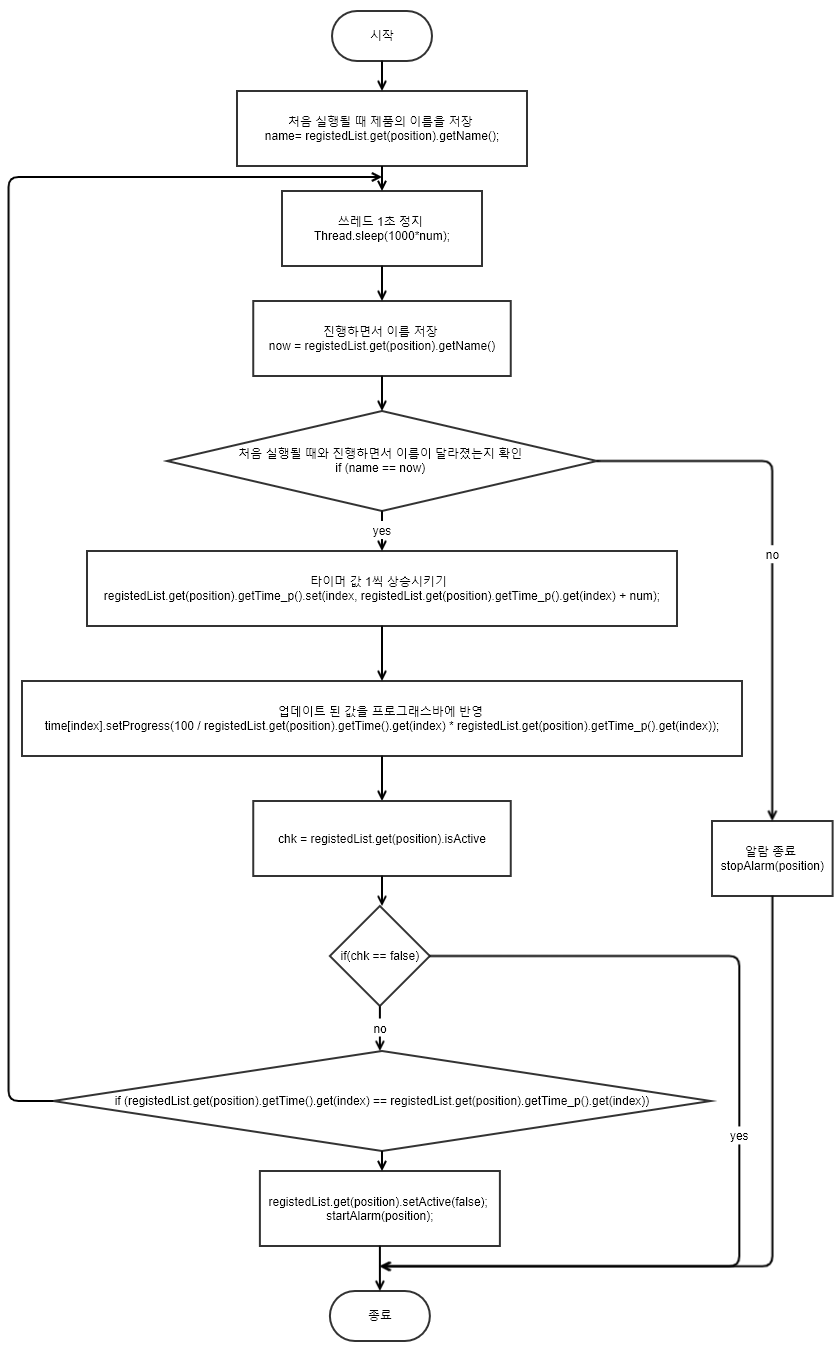
* 1. **최대 타이머가 가득 찼는지(isMax) - MainActivity.java에서 호출  
     **
  2. **최대 타이머 개수 입력(setMax) - MainActivity.java에서 호출  
     **
  3. **타이머에 등록(registOnRegistedList) - MainActivity.java에서 호출  
     **
  4. **대기큐에서 삭제(onQueueDeleteClicked) - MainActivity.java에서 호출**

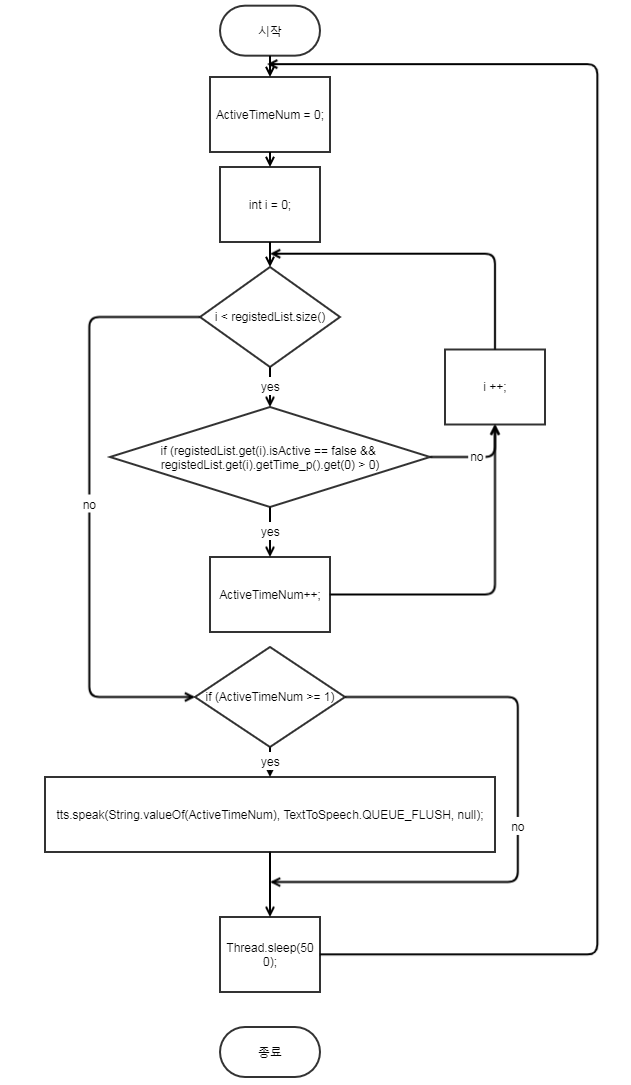
****

* 1. **타이머가 한개일 경우 뷰(View)설정 – FFAdapter.java에서 호출**

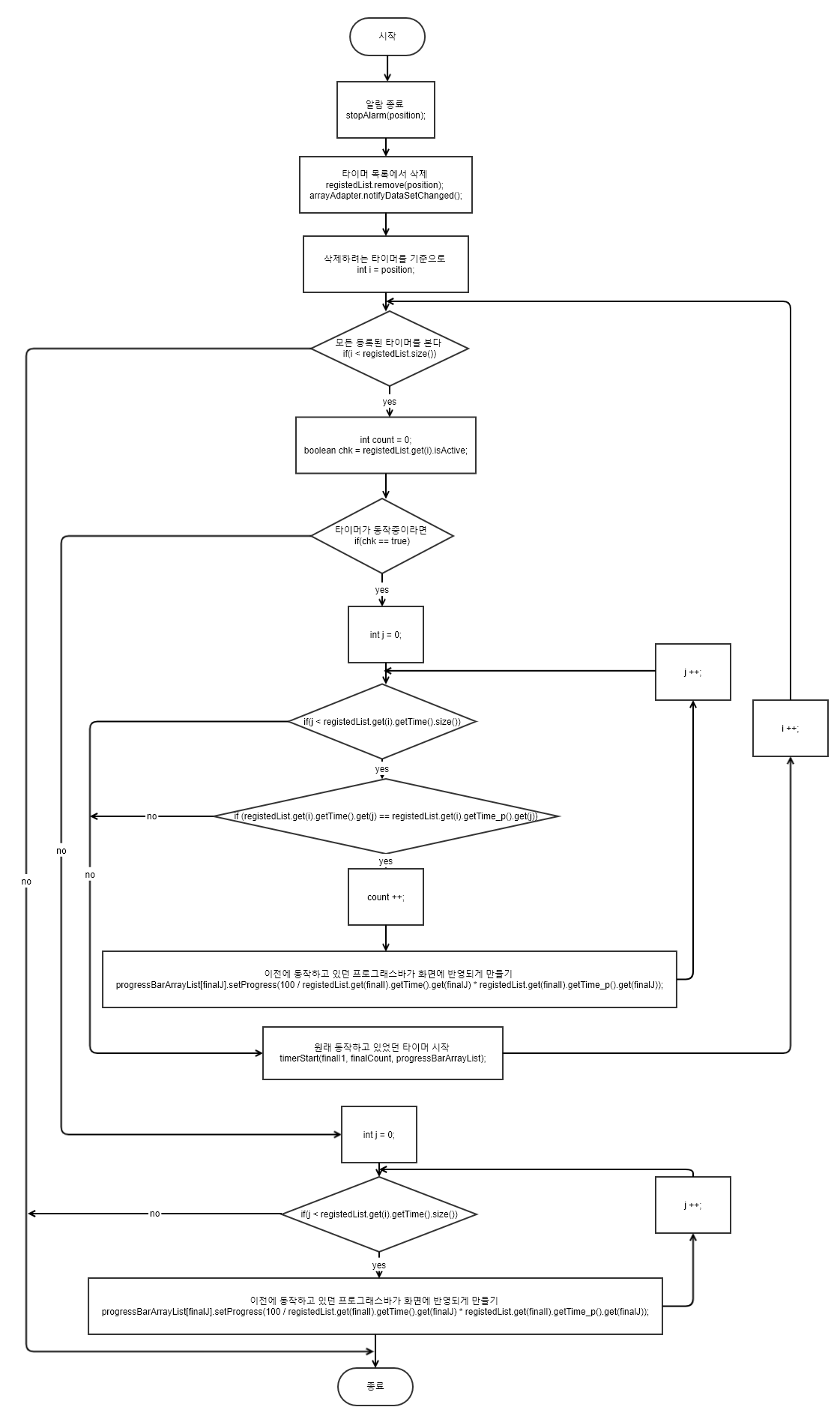
****

1. **타이머 실행 순서도(timerplay)**
   1. **타이머 시작버튼 – FFAdapter.java에서 호출  
      **
   2. **타이머 쓰레드로 동작시키기(timerStart) – FFAdapter.java에서 호출**

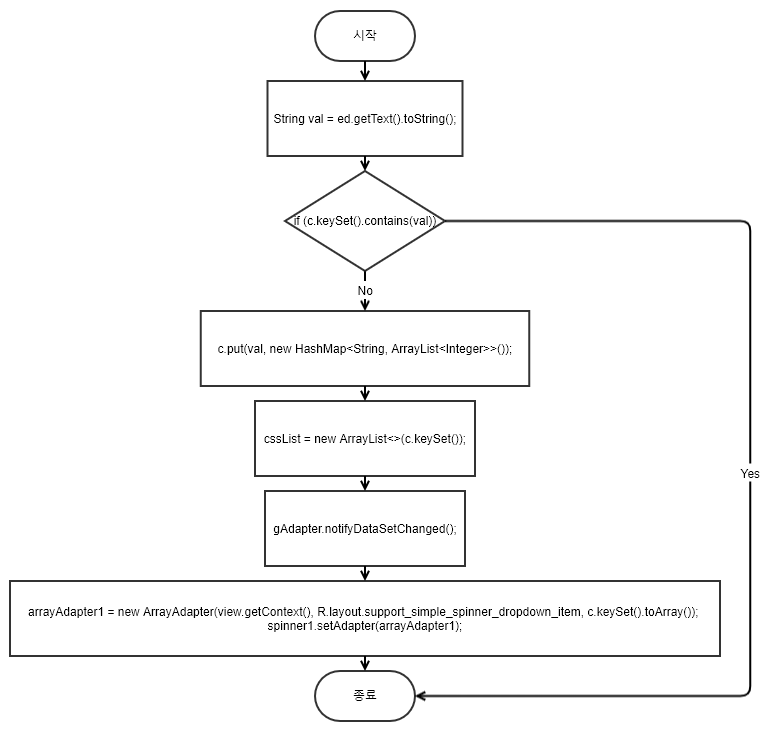
****

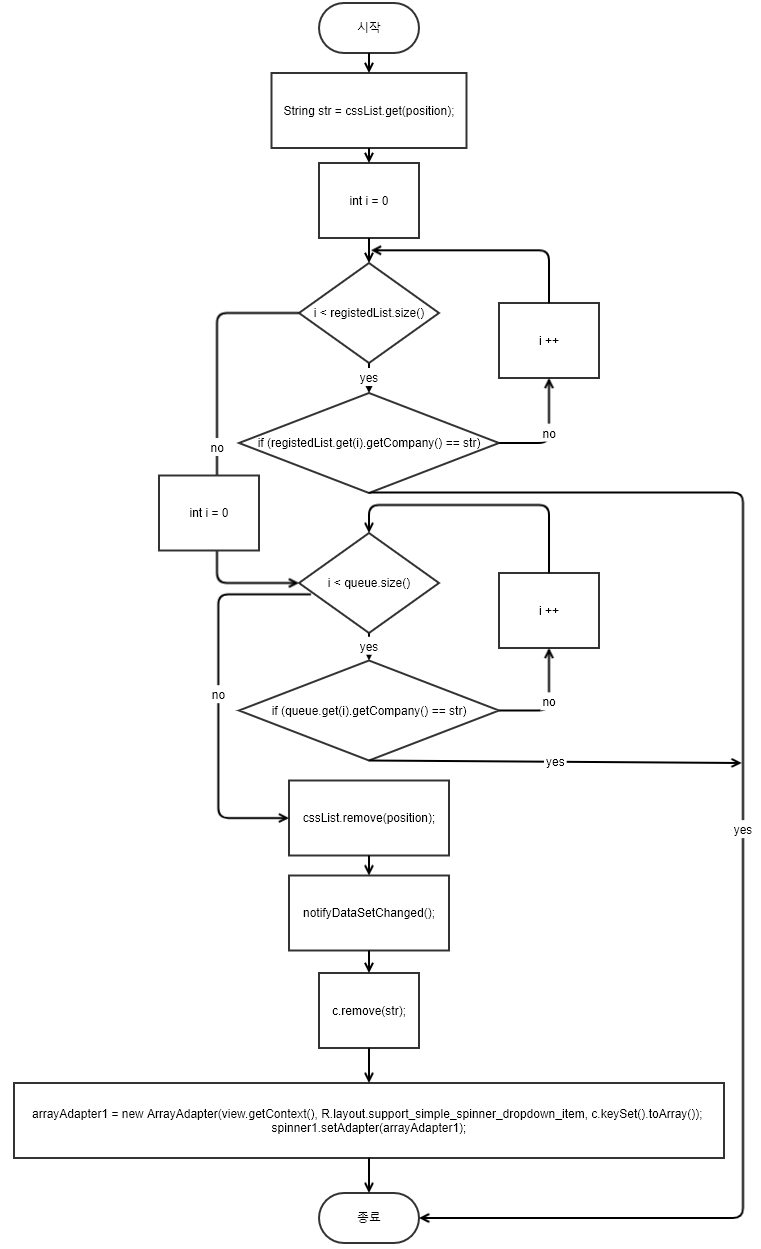
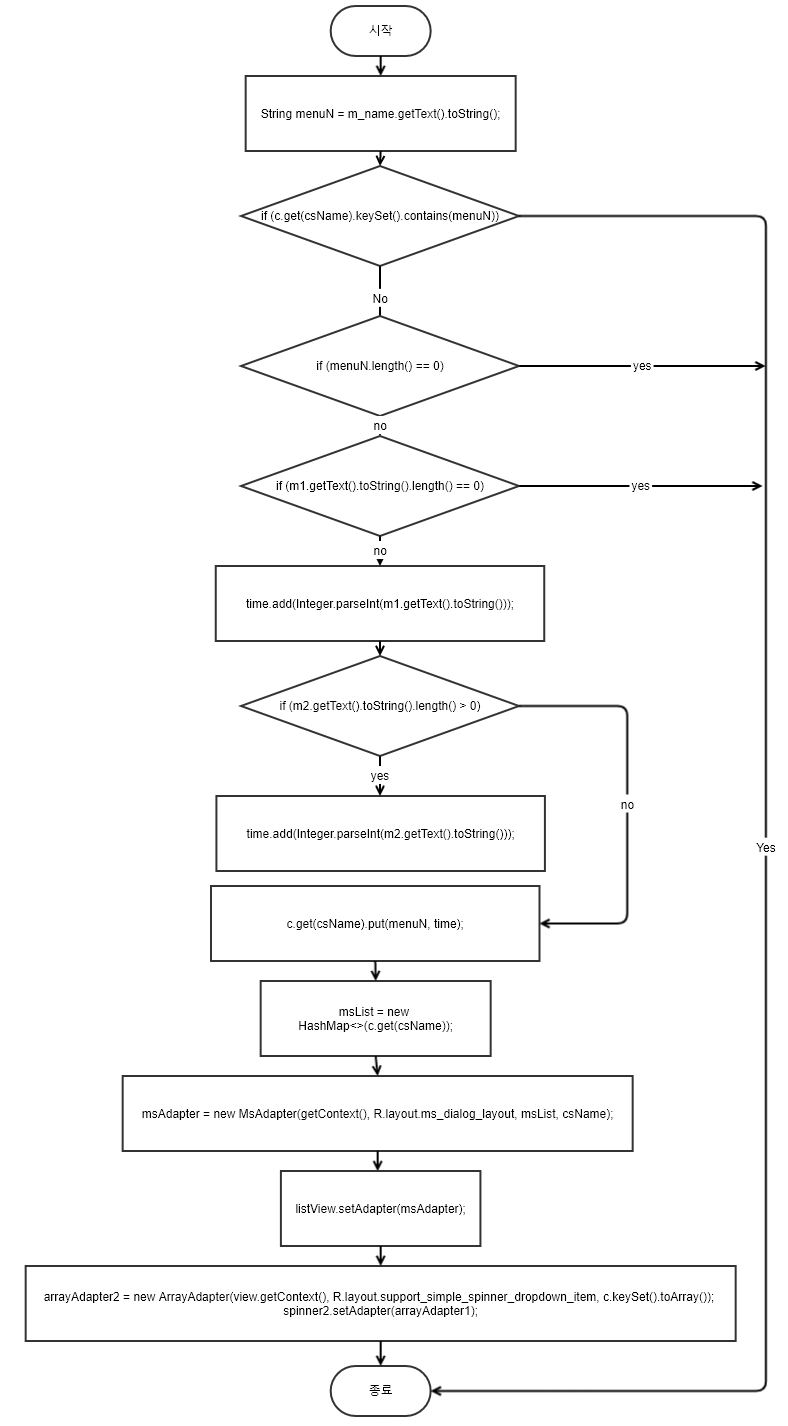
* 1. **알람음 울리기(AlertTimerEnd) – FFAdapter.java에서 호출  
     **

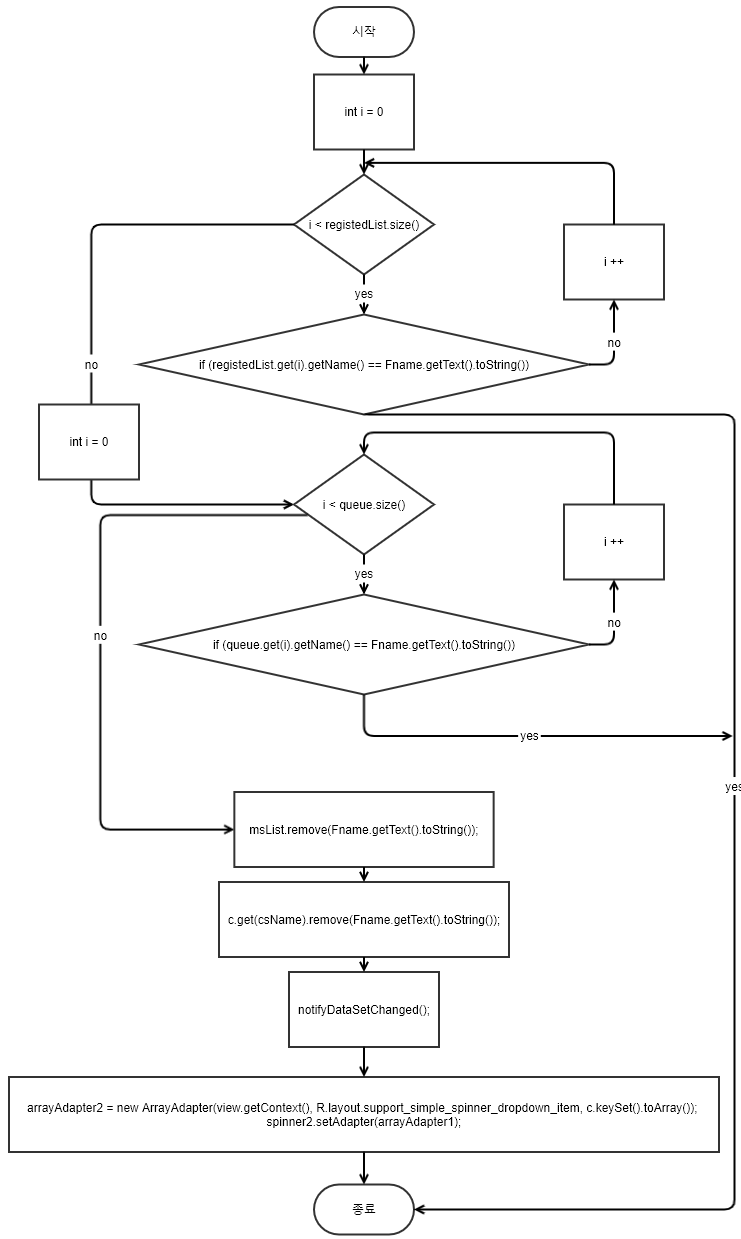
1. **타이머 삭제 순서도(timerdelete) – FFAdapter.java에서 호출**

****

1. **편의점 / 제품 변경 순서도**
   1. **편의점 등록 순서도 - GDialog.java 에서 호출**

****

* 1. **편의점 삭제 순서도 – GAdapter.java에서 호출  
     **
  2. **제품 등록 순서도 - MsDialog.java에서 호출  
     **
  3. **제품 삭제 순서도 - MsAdapter.java에서 호출**

****

1. **기능에 대한 추가설명**
   1. **데이터 초기화**

ff메뉴에 대한 메뉴 정보를 넣어주고 등록을 하는 과정이다.

본격적으로 실행하기에 앞서, 실행에 필요한 데이터들을 넣어준다. 대표적인 데이터로는, 편의점 이름, 메뉴 이름, 조리 시간(전자레인지 조리시간, 튀김기 조리 시간)이 있다. 이 데이터들은 HashMap<string,HashMap<string, int[] >>타입에 담겨 있다. 첫번째 key값에 해당하는 string에는 편의점 매장 이름이 들어가고, value값에 해당하는 hashmap의 key값은 ff제품 이름 그에 해당하는 key값은 전자레인지 조리 시간과 튀김기 조리 시간을 함께 담고 있는 int배열이다.

* 1. **등록 가능한 타이머 개수, 등록시 중복체크, 대기큐 관리**

등록 가능한 타이머 개수 : 초기에 프로그램이 실행되고 사용자로부터 최대 등록 가능한 타이머(조리기구) 개수를 입력받는다.

스피너에서 등록 : 스피너에서 등록은 스피너 옆에 있는 버튼을 눌러서 타이머를 등록하는 것을 말한다. 스피너 등록을 하면 등록된 타이머 개수가 몇개 인지 확인한다. 만약 이미 등록된 타이머 개수가 최대 등록 가능한 개수와 같다면 대기큐에 등록된다. 타이머리스트에 빈자리가 있다고 하더라도, 이미 등록된 타이머리스트에 현재 등록하려는 메뉴와 동일한 메뉴가 등록되어있다면 대기큐에 등록한다.

대기큐에서 등록 : 대기큐에서 등록은 대기큐 옆에 있는 버튼을 눌러서 타이머를 등록하는 것을 말한다. 여기에서는, 대기큐에 먼저 등록된 타이머를 먼저 등록하는 FIFO(First In First Out)방식을 채택한다. 등록 버튼을 눌렀을 때, 타이머리스트에 빈 자리가 없는 경우에는, 어떠한 이벤트도 발생하지 않는다. 만약 타이머 리스트에 빈자리가 있다면, 리스트에 현재 대기큐에서 등록하려는 메뉴와 중복된 메뉴가 있는지 확인한다. 만약 중복메뉴가 존재하면 해당 메뉴를 대기큐에서 삭제하고 대기큐에있는 다음 메뉴가 중복되는지 확인한다. 이런 식으로 중복되지 않은 타이머가 나올 때까지 중복 메뉴를 삭제시키고, 중복되지 않은 타이머를 찾은 경우에 이를 등록한다. 대기큐를 전체 탐색 했는데도 중복되지 않은 타이머를 찾이 못한경우에는 대기큐열이 모두 비워지고, 아무것도 타이머리스트에 추가 등록되지 않는다.

* 1. **리스트뷰 화면에 타이머 보여주기**

fName.setText(ff.getName)

mTime.setProgress(100/ff.getMtime()\*ff.getMtime\_p)

mTime.setProgress(100/ff.getFtime()\*ff.getFtime\_p)

100/ff.getMtime()\*ff.getMtime\_p 가 인자로 들어가는 이유 : 프로그래스바의 최댓값은 100이다. getMtime\_p가 1씩 증가한다. 전자레인지 시간을 가지고 있는 mtime정보를 이용해서 1초마다 100/ff.getMtime() 만큼 프로그래스바가 채워지게 만들었다.

* 1. **타이머 시작**

타이머 시작을 눌렀을 때 실행되는 루틴이다. isActive를 통해 현재 타이머의 상태를 판별한다. 타이머가 진행중이라면 시작버튼을 눌러도 아무런 변화 없이, 기존에 실행시키던 타이머를 계속 실행시킨다. 만약 타이머가 현재 진행되고 있지 않다면, 튀김기 타이머와 전자렌지 타이머 중 무엇을 시작할 지 if(Mtime == Mtime\_p)를 이용해서 판별한다. 전자레인지 시간이 완료되지 않은 경우, 즉, 전자레인지 타이머가 실행되지 않은 경우에는 전자레인지 타이머를 시작한다. 현재 전자레인지 시간이 다 된 경우, 시작 버튼을 누르면 튀김기 타이머를 시작한다. 멀티 타이머가 가능하게 하기 위해, 쓰레드를 이용한다. 매 초마다, Ftime\_p 혹은 Mtime\_p 값을 1씩 증가시키고, FF 제품 정보에 저장된 튀김기시간 혹은 전자레인지 시간과 같아지면 해당 타이머의 쓰레드를 종료한다.

* 1. **타이머 삭제**

타이머 삭제 버튼을 누르면 registedList에서 해당 타이머 정보를 삭제하고 adapter를 업데이트 한다.

* 1. **편의점 등록, 삭제**

편의점을 등록할 때는 기존에 존재하는 편의점을 등록할 수 없고, 이미 타이머로 등록된 제품이 삭제하려는 편의점 제품일 때 그 편의점은 삭제할 수 없습니다.

* 1. **제품 등록, 삭제**

제품을 등록할 때는 기존에 존재하는 제품을 등록할 수 없고, 이미 타이머로 등록된 제품이라면 그 제품은 삭제할 수 없습니다.