Bread bake machine Simulator

```
>> SetMenu: Bread
>> SetTime: 4:50
timer set to: 300000ms (5 min)


Menu
====
(1) Menu button
(2) Menu button long press
(3) Timer up button
(4) Timer down button
(5) Start button
(6) Set oven temperature
----------------------
(q) QUIT
Enter: repeat previous choice
?: show menu again

Choice :
```

Introduction

The breadbake project directory contains quite a bit of code. It contains:

- a product directory with a complete simulation of the hardware layer
- a test directory with all mocks and an example unit test file
- a Makefile that is prepared for building the simulator (`make`) and for building + running your tests (`make test`)

The code compiles and runs properly. If you run the tests you will get the following results:

```
student@fontys:/mnt/hgfs/t-sem3/eds/4_Dynamic_Behaviour/BreadBake$ make test
Running main() from gmock_main.cc
[==========] Running 1 test from 1 test case.
[----------] Global test environment set-up.
[----------] 1 test from StateTest
[ RUN      ] StateTest.bogustest
[       OK ] StateTest.bogustest (1 ms)
[----------] 1 test from StateTest (1 ms total)

[----------] Global test environment tear-down
[==========] 1 test from 1 test case ran. (1 ms total)
[  PASSED  ] 1 test.
student@fontys:/mnt/hgfs/t-sem3/eds/4_Dynamic_Behaviour/BreadBake$
```

If you run the simulator (`./breadbake`) you will get a running project:

```
$ ./breadbake


Menu
====
(1) Menu button
(2) Menu button long press
(3) Timer up button
(4) Timer down button
(5) Start button
(6) Set oven temperature
---------------------
(q) QUIT
Enter: repeat previous choice
?: show menu again

Choice :
```

The Simulator

The simulator works pretty basic. The real system has some buttons:

The simulator has the same "buttons" (i.e.: menu choices):

```
$ ./breadbake


Menu
====
(1) Menu button
(2) Menu button long press
(3) Timer up button
(4) Timer down button
(5) Start button
```
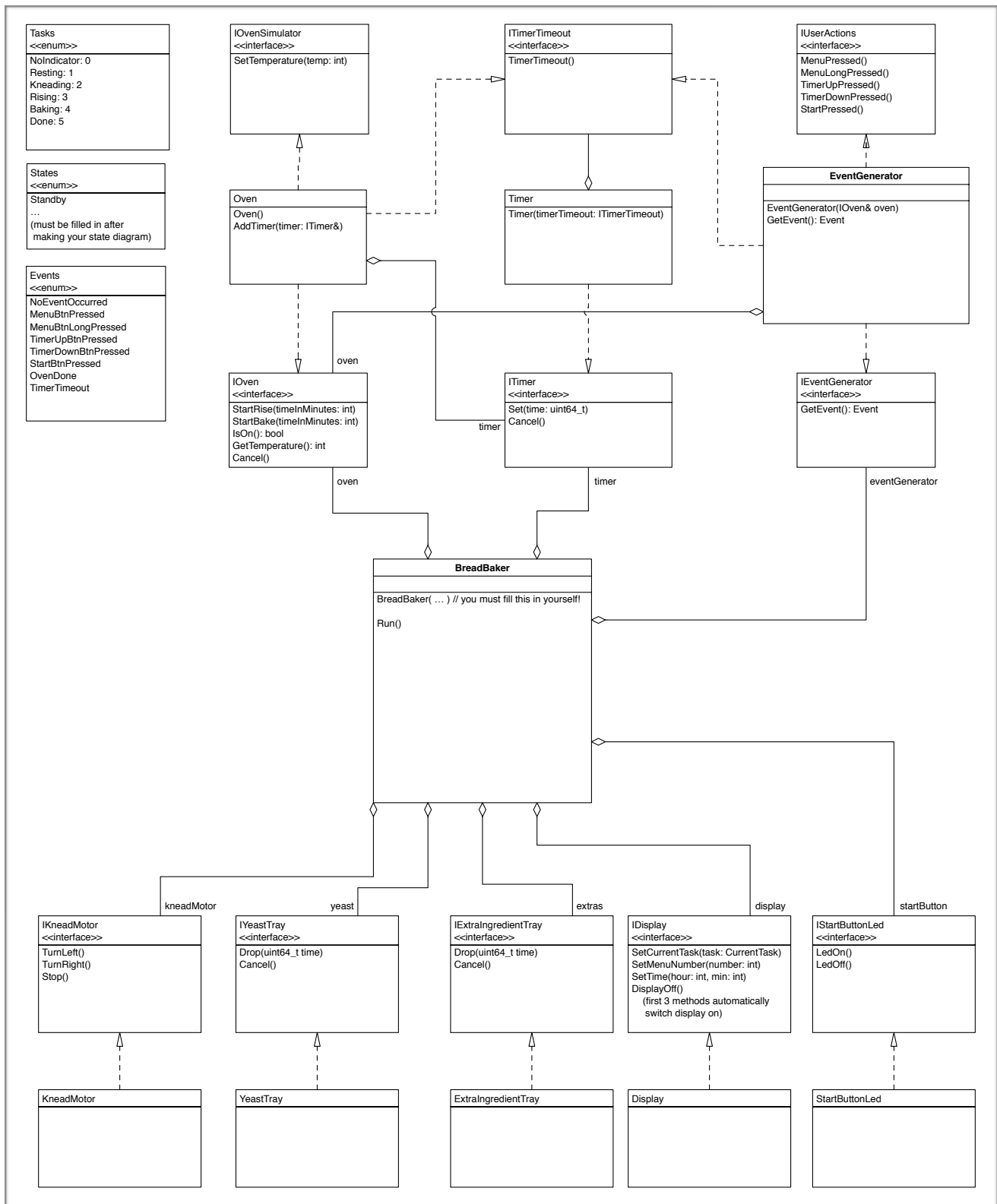
In `main.cpp` you'll find some hints in comment that BreadBaker::Run must be started in a separate thread. After you've done that you'll see events being logged on the display:

```
Choice : 1
MenuBtnPressed event occurred
```

Menu option 6 is implemented so you can control the oven's temperature to test if the machine starts when cold and doesn't start when hot.

The Simulator: software design
CLASS DIAGRAM

Tasks
<<enum>>
NoIndicator: 0
Resting: 1
Kneading: 2
Rising: 3
Baking: 4
Done: 5

States
<<enum>>
Standby
…
(must be filled in after
 making your state diagram)

Events
<<enum>>
NoEventOccurred
MenuBtnPressed
MenuBtnLongPressed
TimerUpBtnPressed
TimerDownBtnPressed
StartBtnPressed
OvenDone
TimerTimeout

IOvenSimulator
<<interface>>
SetTemperature(temp: int)

ITimerTimeout
<<interface>>
TimerTimeout()

IUserActions
<<interface>>
MenuPressed()
MenuLongPressed()
TimerUpPressed()
TimerDownPressed()
StartPressed()

Oven
Oven()
AddTimer(timer: ITimer&)

Timer
Timer(timerTimeout: ITimerTimeout)

EventGenerator
EventGenerator(IOven& oven)
GetEvent(): Event

IOven
<<interface>>
StartRise(timeInMinutes: int)
StartBake(timeInMinutes: int)
IsOn(): bool
GetTemperature(): int
Cancel()

ITimer
<<interface>>
Set(time: uint64_t)
Cancel()

IEventGenerator
<<interface>>
GetEvent(): Event

oven

timer

eventGenerator

oven

timer

BreadBaker
BreadBaker(…) // you must fill this in yourself!

Run()

kneadMotor

yeast

extras

display

startButton

IKneadMotor
<<interface>>
TurnLeft()
TurnRight()
Stop()

IYeastTray
<<interface>>
Drop(uint64_t time)
Cancel()

IExtraIngredientTray
<<interface>>
Drop(uint64_t time)
Cancel()

IDisplay
<<interface>>
SetCurrentTask(task: CurrentTask)
SetMenuNumber(number: int)
SetTime(hour: int, min: int)
DisplayOff()
   (first 3 methods automatically
    switch display on)

IStartButtonLed
<<interface>>
LedOn()
LedOff()

KneadMotor

YeastTray

ExtraIngredientTray

Display

StartButtonLed

Please note: the Log class is not mentioned in this diagram as it would only give extra clutter. All classes in the diagram use the Log class to log debug and trace information.

CLASS RESPONSIBILITIES

Class	Responsibility
BreadBaker	Implements the bread baker's state machine. `Run()` is already implemented and should be started in a separate thread. It keeps reading the latest event and feeds that to the state machine. This method stops when `quit` becomes true (same quit as in user interface). **TODO: implement state behaviour**
Oven	Simulates the bread baker's oven. It implements 2 interfaces: - IOven, which is the interface for the 'real' system - IOvenSimulator, which is only there so you can control the oven temperature from the user interface. Oven uses a Timer object for its timing.
Timer	Provides timer features to the bread bake machine. This class is not yet fully implemented but should work as follows: - Set should start a new thread that counts down the number of mili seconds (each time sleeping for 1 ms). After counting down, the TimerTimeout method is called on the provided ITimerTimeout interface. - Cancel should stop the timer. After calling Cancel, the TimerTimeout method will obviously not be called. - As both Oven and your state machine will use a different Timer object, you must make sure these 2 timers do not influence each other! *Please note that the private method GetSimulatedTime is implemented to make testing less annoying (it limits the number of ticks quite dramatically).* **TODO: implement the timer class**
EventGenerator	Queues system events. Events that are pushed (i.e. timer and UI events) are directly written into a queue. Events that need to be polled (oven) will be polled each time GetEvent is called.
KneadMotor YeastTray ExtraIngredientTray Display StartButtonLed	In a real bread baker these classes would talk to hardware, in the simulator all these classes do is write to the debug or trace log.
Log	Writes all log messages to std::cout. **TODO: Log will be called from multiple threads, make sure messages do not interfere each other**

An extra note on output to standard output: the user interface in main.cpp is written in such a way that it is more likely for you to see the synchronisation problem that occurs with writing to standard output from multiple threads.

The menu will be printed each second and it will wait 2 ms between printing each line. This makes sure you will be more likely to see problems like these:

Important: you are of course not allowed to change this. You must make sure all output to the terminal are properly synchronised.

```
Menu
====
== GetEvent returns: StartBtnPressed
>> GetTemperature: 0

timer set to: 3600000 ms (60 min  ==> simulated time: 4000 ms)
>> SetCurrentTask: Waiting
(1) Menu button
(2) Menu button long press
(3) Timer up button
(4) Timer down button
(5) Start button
(6) Set oven temperature
--------------------
(q) QUIT
Enter: repeat previous choice
```

If you have any questions about the simulator, please ask your teacher!