2020

Pomodoro Technique

JORGE SANCHEZ
LILIAN MARA ONYAMBU
GREGORY GIVENS JR

INTRODUCTION

A pomodoro technique is a tool where you use a timer to break down your work in time intervals with short breaks in between each interval. One can set study period for a certain amount of time say, 20 to 30 minutes, and break period for another amount of time say, 5 to 10 minutes for breaks. This sort of application would definitely be of use to students working on big assignments or doing a lot of writing and giving them the quick break that they need to refocus. One of the main features we will need to develop is setting up the timer and a GUI that is easy to use

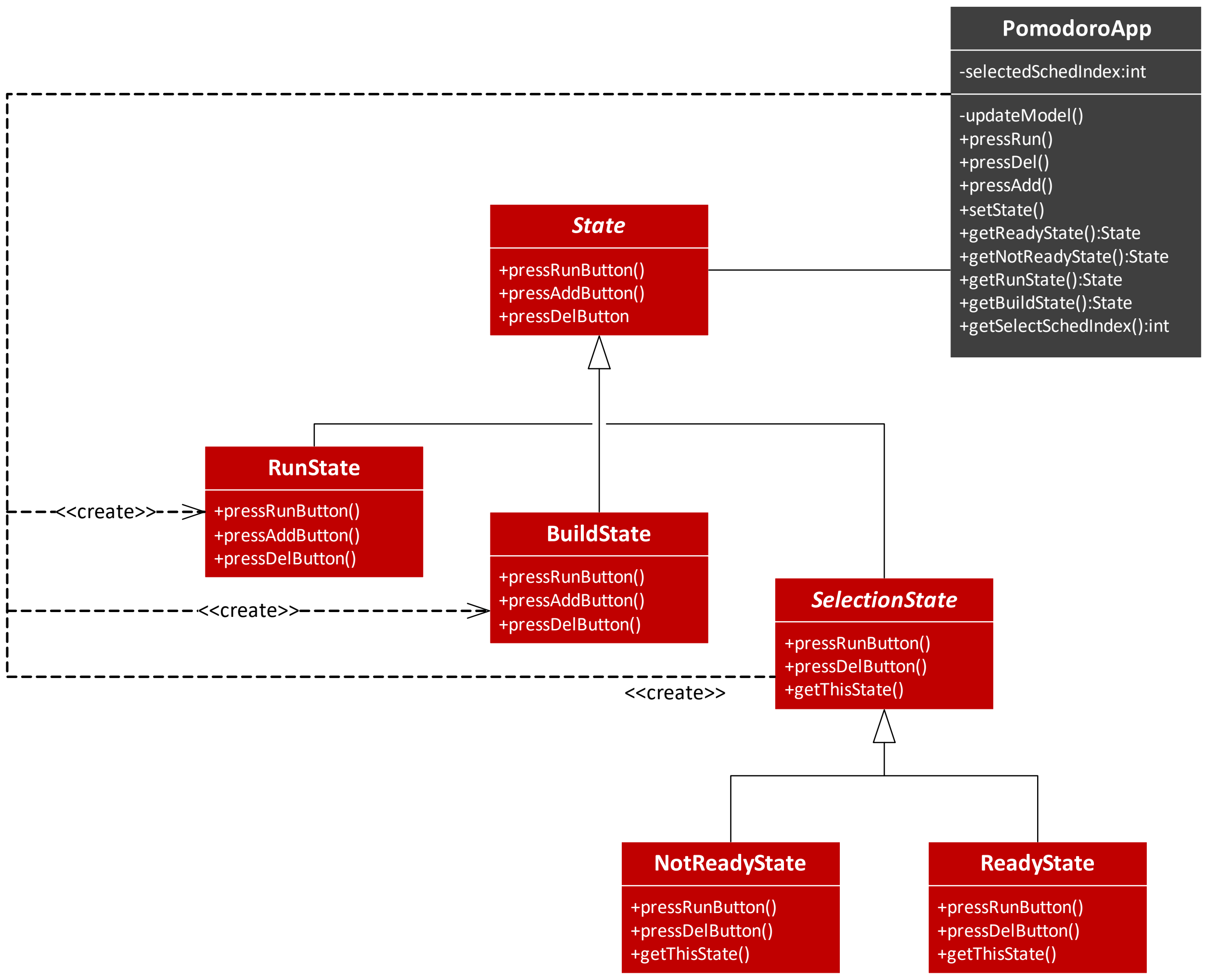
In our Pomodoro technique, we develop two schedule that allow a user to choose a default or custom schedule to use. In a custom schedule, the user does not have to input the study minutes or break minutes because they are provided in default. The study and break time are provided in rounds of five. In the custom schedule, the user is allowed to enter the amount of time they want for their study periods and break periods. They can also add many study and break periods according to what they desire.

In this system, four different design patterns are used. These include the State Design Pattern, Strategy Design Patterns, Builder Design Pattern and Composite Design Pattern.

SYSTEM DESIGN PATTERNS

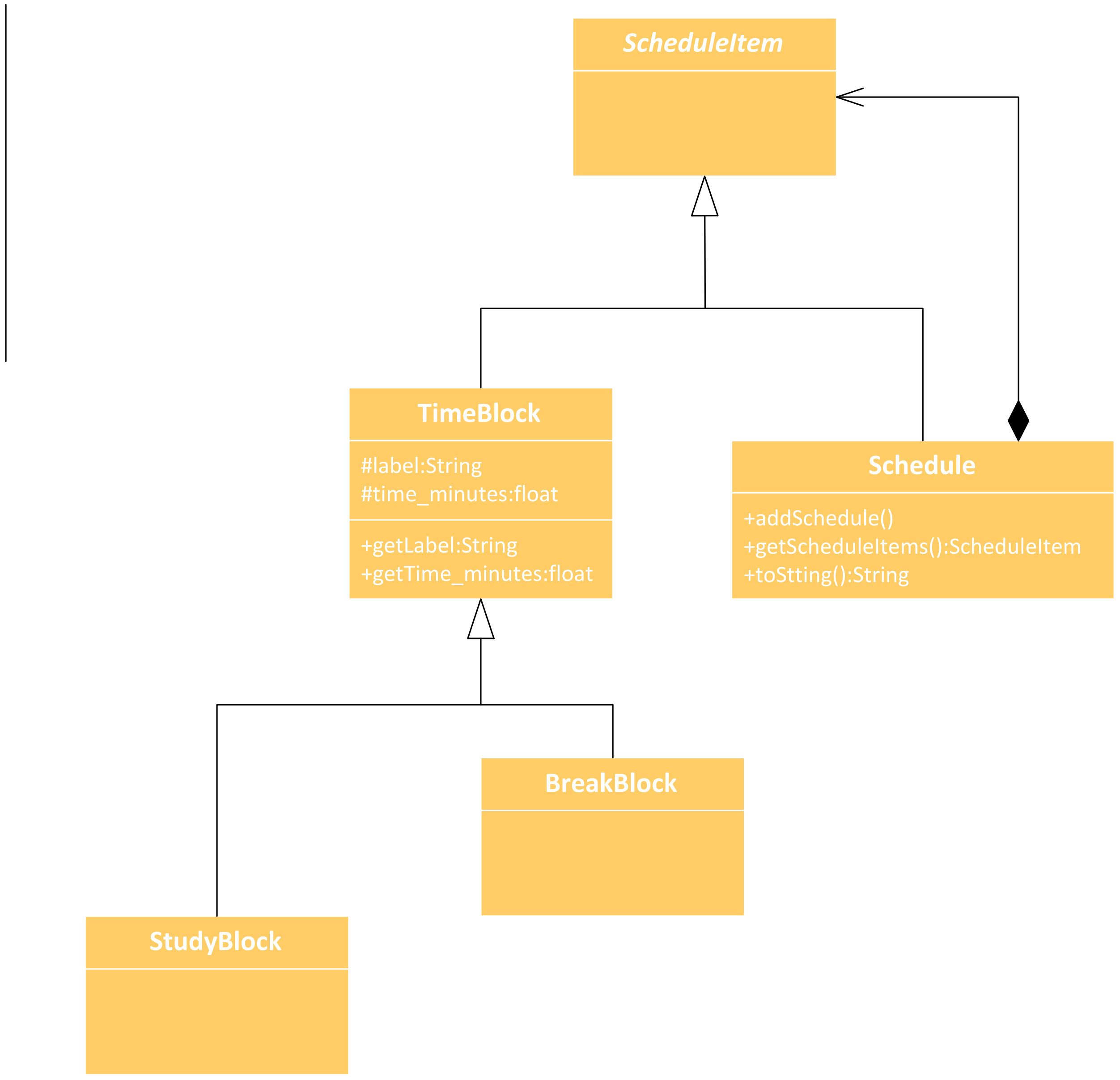
State Design Pattern

The State Pattern is used to direct the program flow through a set of states. The program will move through three states: The Run State, The Build State, and The Selection State. The Selection state is both Ready State and Note Ready State. The system changes state from building the schedules, selecting the one to run and running it when it is ready to run it.



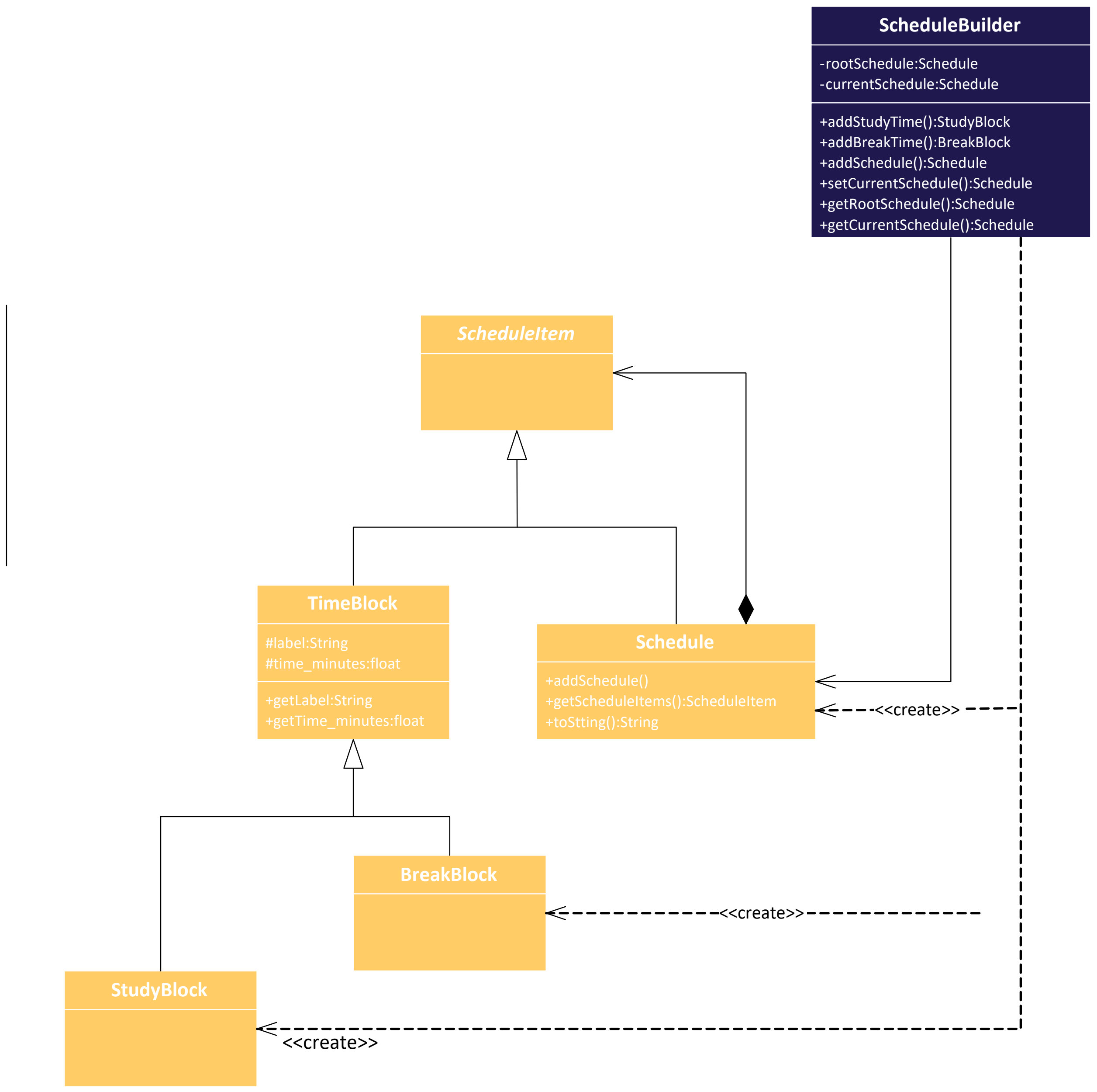
Composite:

The Composite Pattern is used to construct a Schedule and a Schedule System. A Schedule System will be composed of several Schedules and each schedule will be composed of a set of TimeBlock elements: StudyTimeBlock and BreakTimeBlock.



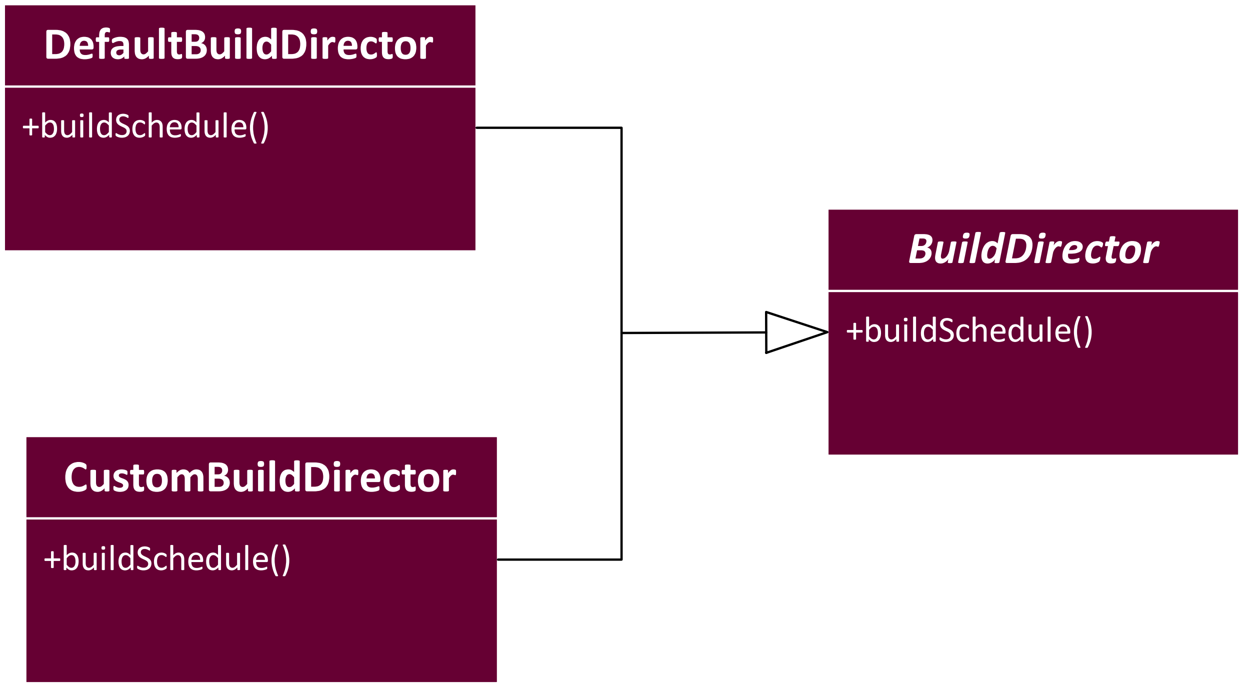
Builder:

The Builder pattern is implemented through the use of the ScheduleBuilder class which contains a series of methods that put together a composite Schedule. A BuildDirecter object will direct the ScheduleBuilder on how to put together a schedule.

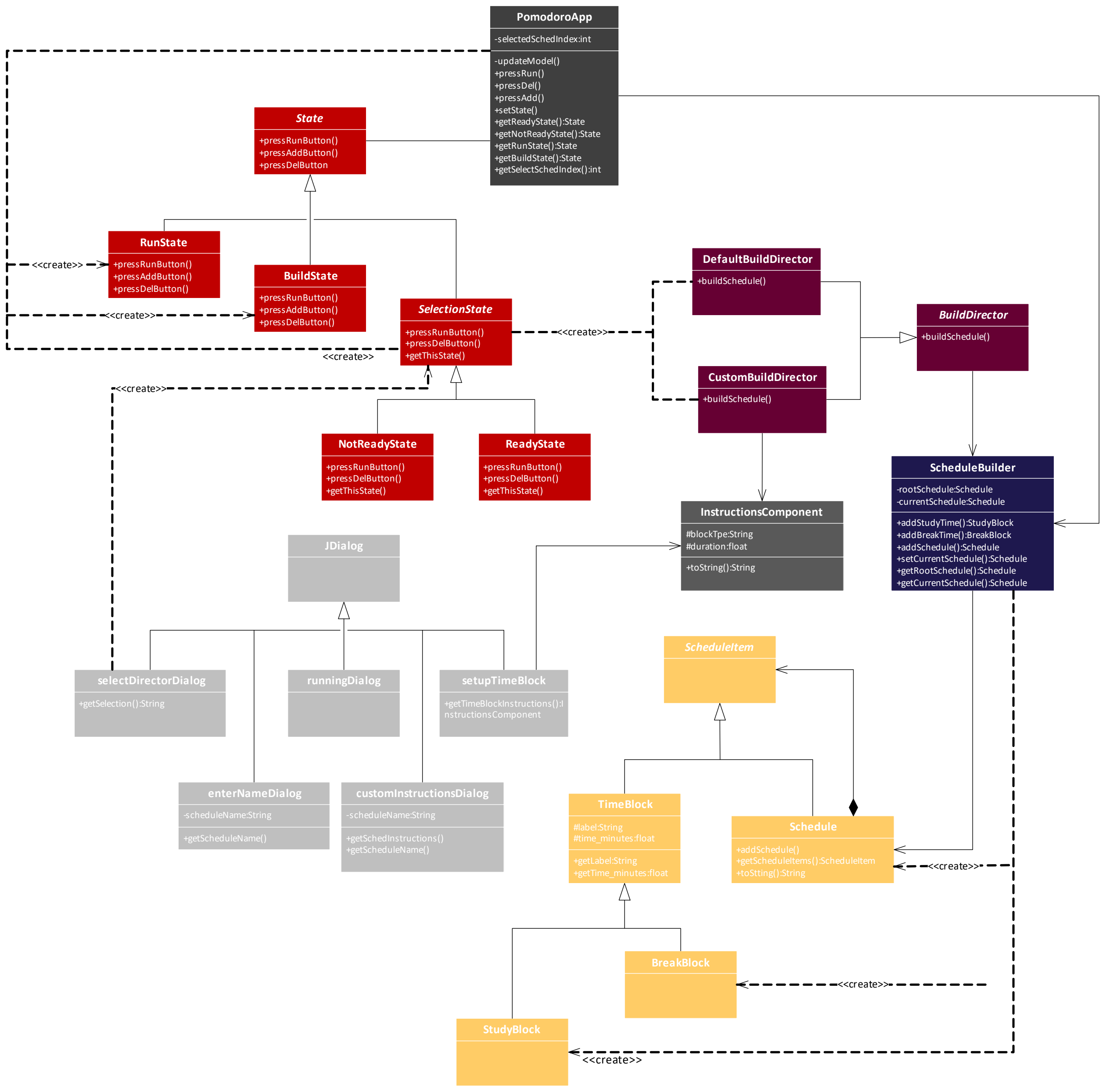


Strategy:

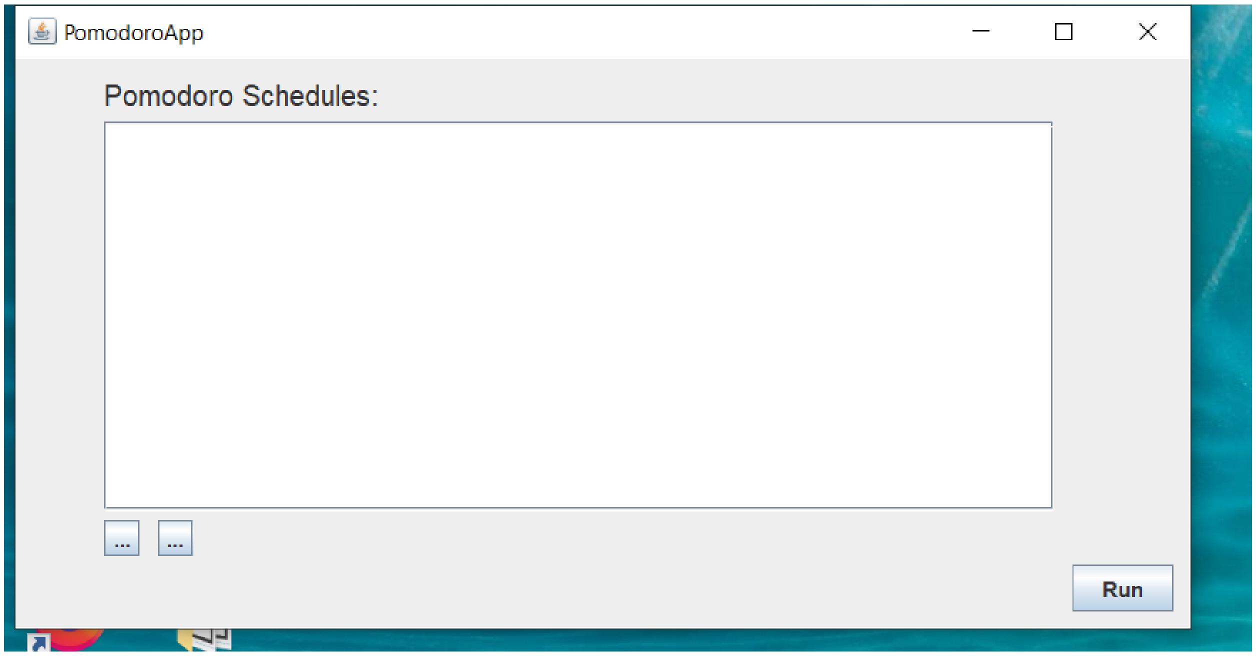
There are three two Director strategies which direct the ScheduleBuilder class in different ways. The first strategy, CustomBuildSchedule directs the building of a custom Pomodoro schedule with user defined study times and break times, and user defined number of study blocks. The second strategy, DefaultBuildSchedule, directs the building of a default Pomodoro schedule set to 25 minute study times and 5 minute break times.

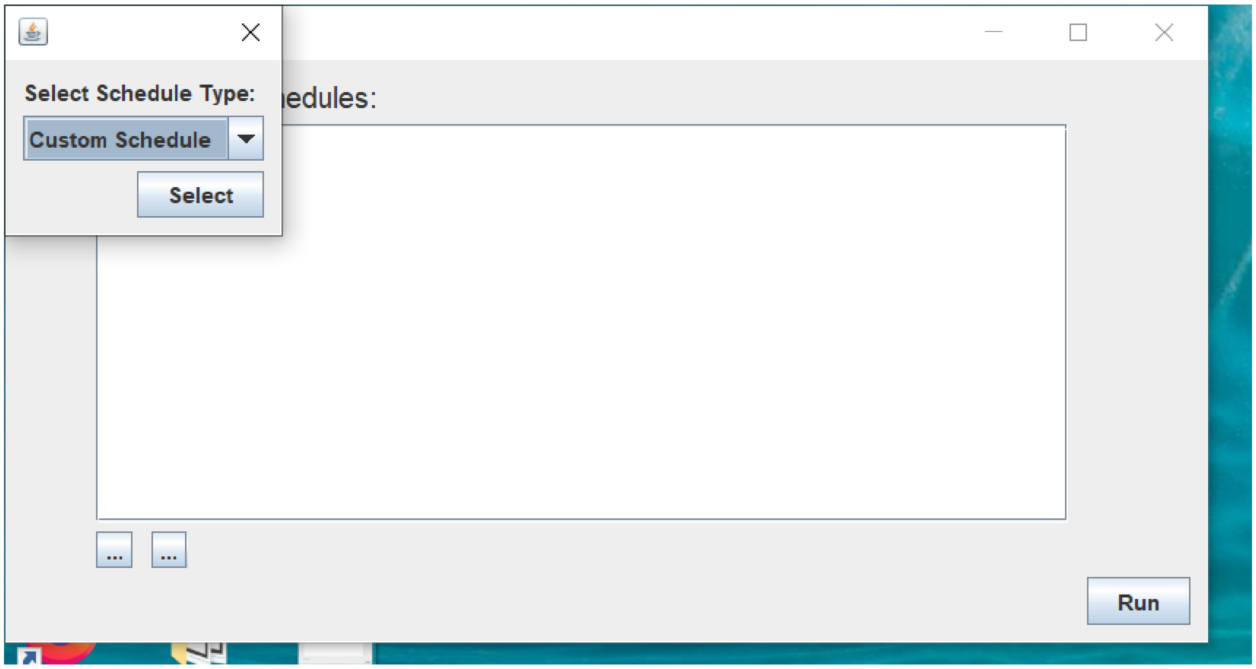


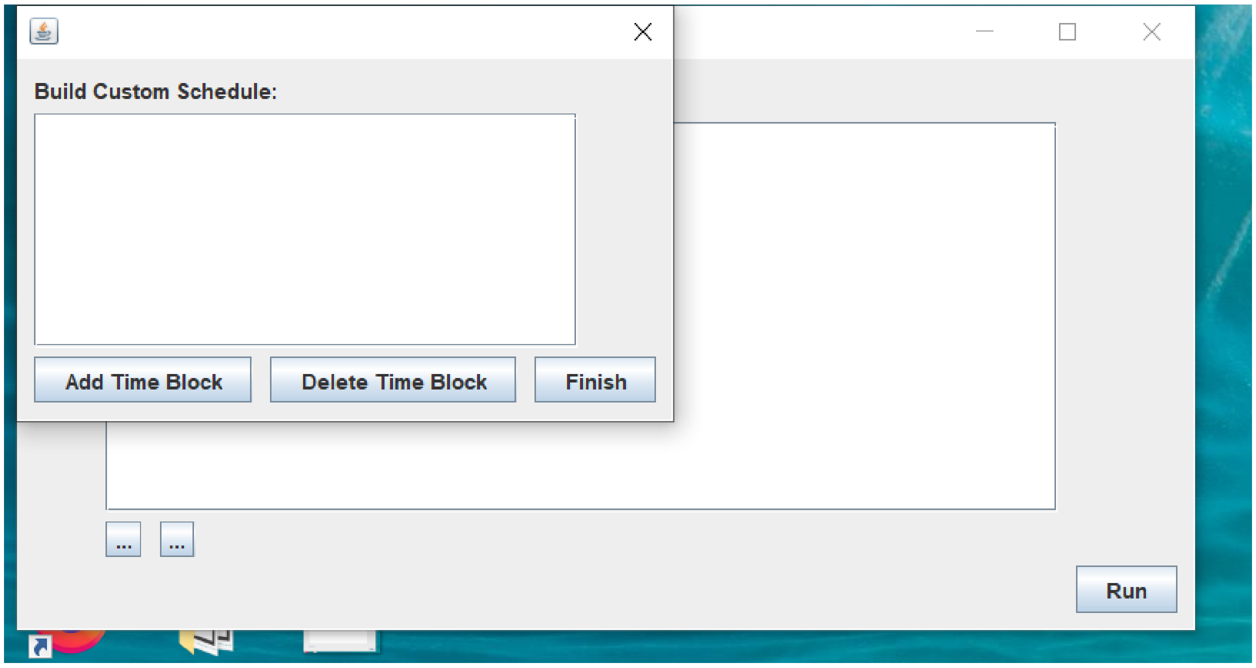
System Implementation

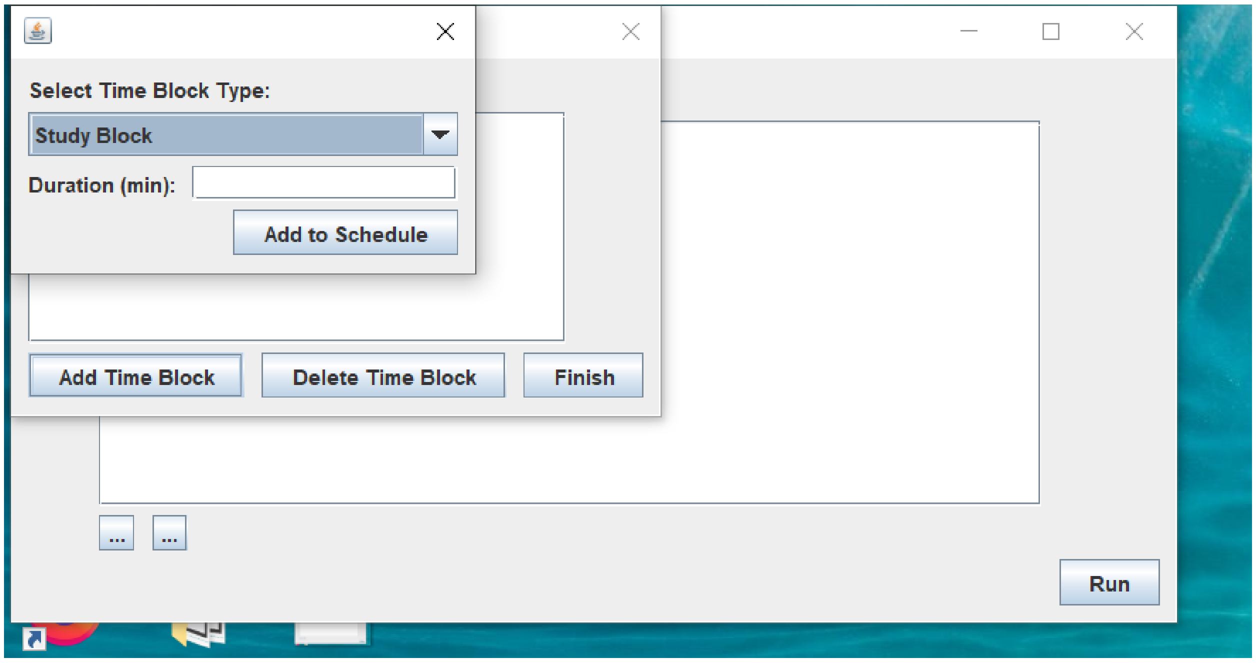


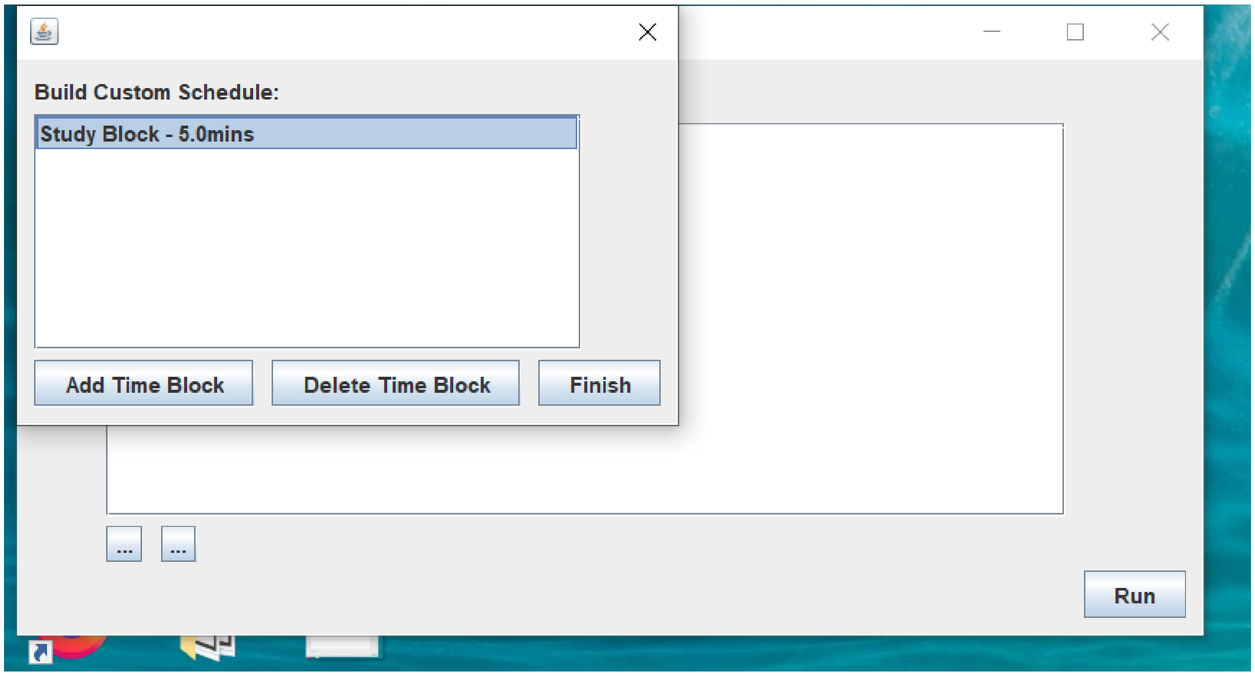
From the above diagram, we use some inbuilt Java classes like JDialog to build the GUI. Building and implementing the GUI also used some classes like JButtonand JtextField. Here is the screenshot of how the system runs from the beginning to a countdown before the first alarm:

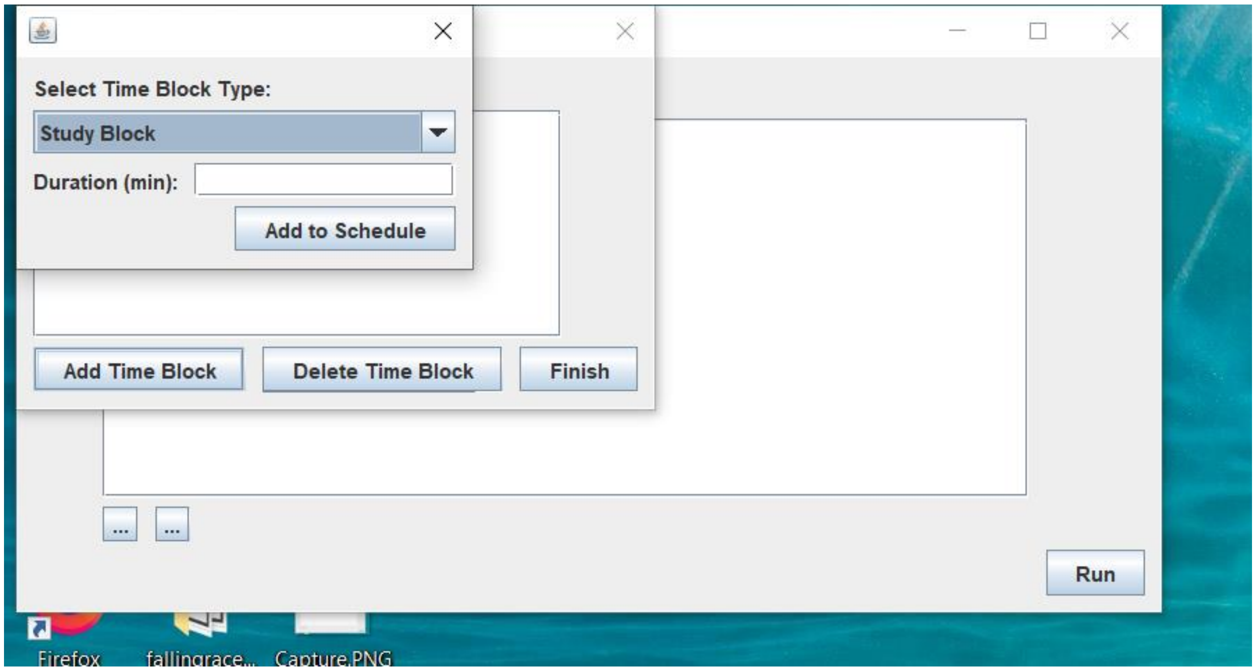

PomodoroApp
Pomodoro Schedules:
...
...
Run

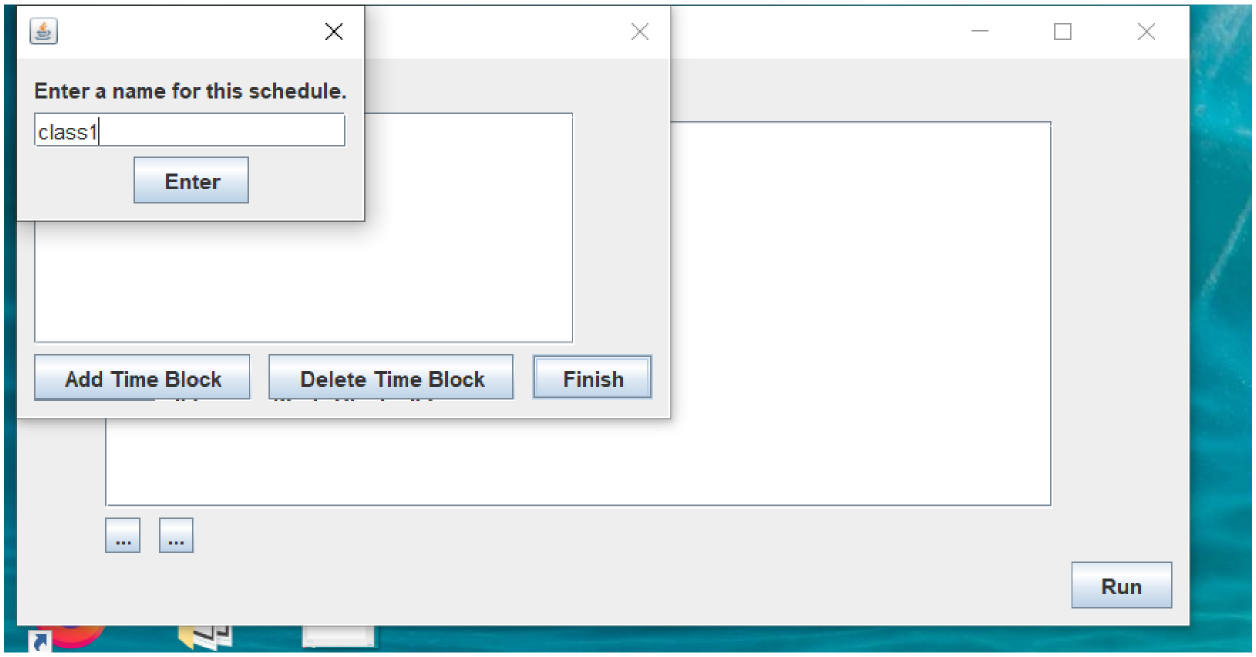

Select Schedule Type:
Custom Schedule
Select
edules:
...
...
Run

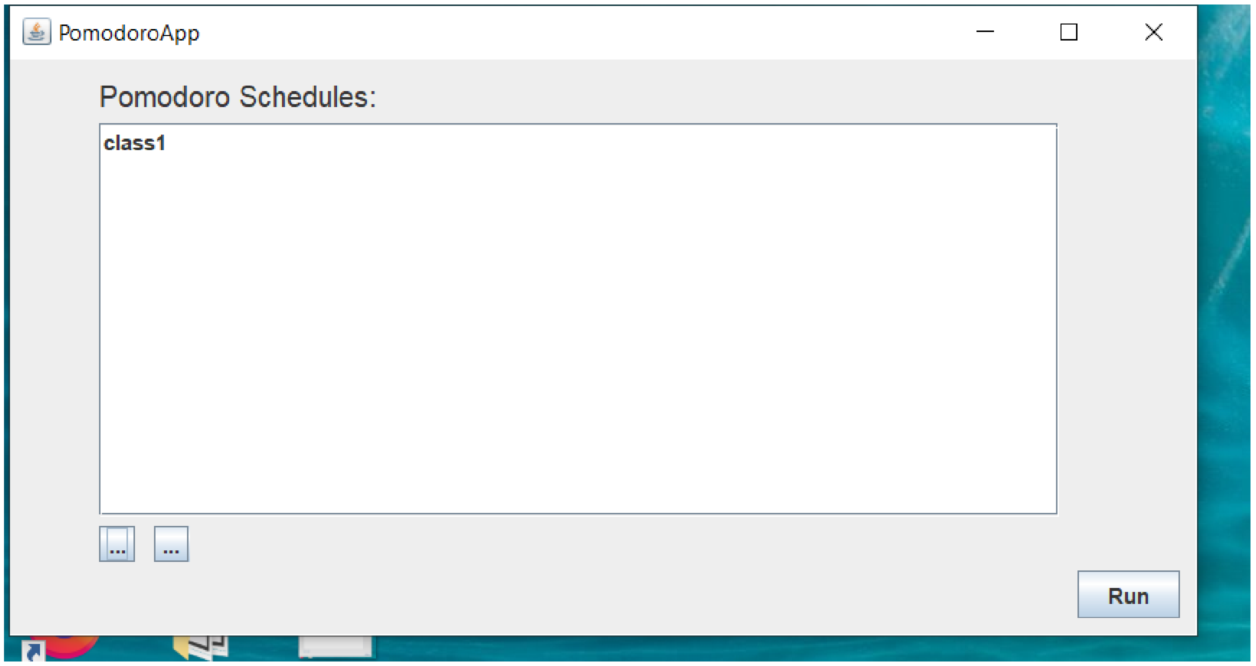

Build Custom Schedule:
Add Time Block
Delete Time Block
Finish
...
...
Run

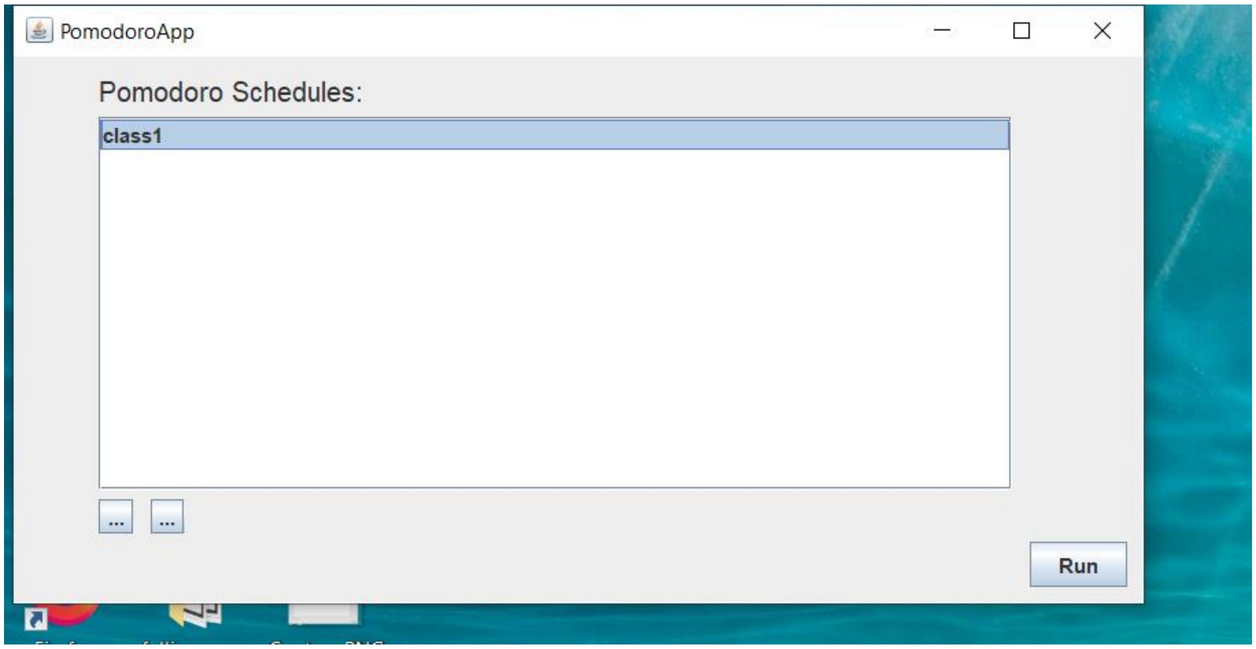

Select Time Block Type:
Study Block
Duration (min):
Add to Schedule
Add Time Block
Delete Time Block
Finish
...
...
Run


Build Custom Schedule:
Study Block - 5.0mins
Add Time Block
Delete Time Block
Finish
...
...
Run


Select Time Block Type:
Study Block
Duration (min):
Add to Schedule
Add Time Block
Delete Time Block
Finish
...
...
Run
Firefox
fallingrace...
Capture.PNG

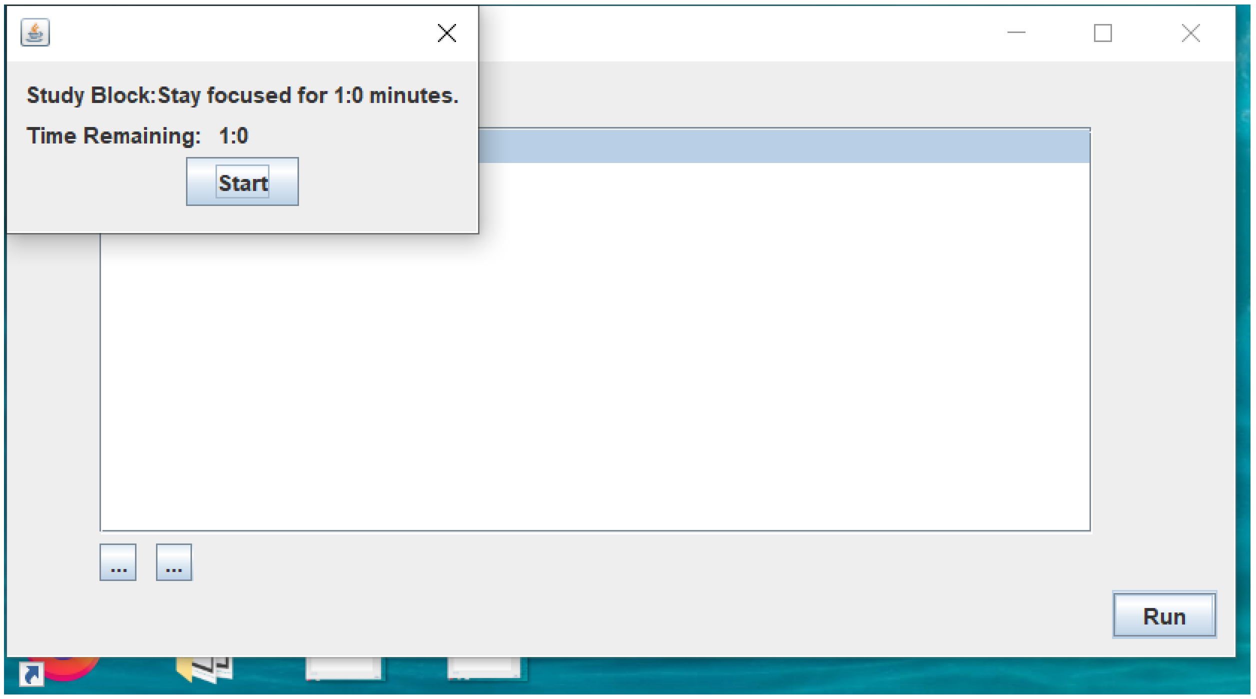

Enter a name for this schedule.
class1
Enter
Add Time Block
Delete Time Block
Finish
...
...
Run


PomodoroApp
Pomodoro Schedules:
class1
...
...
Run

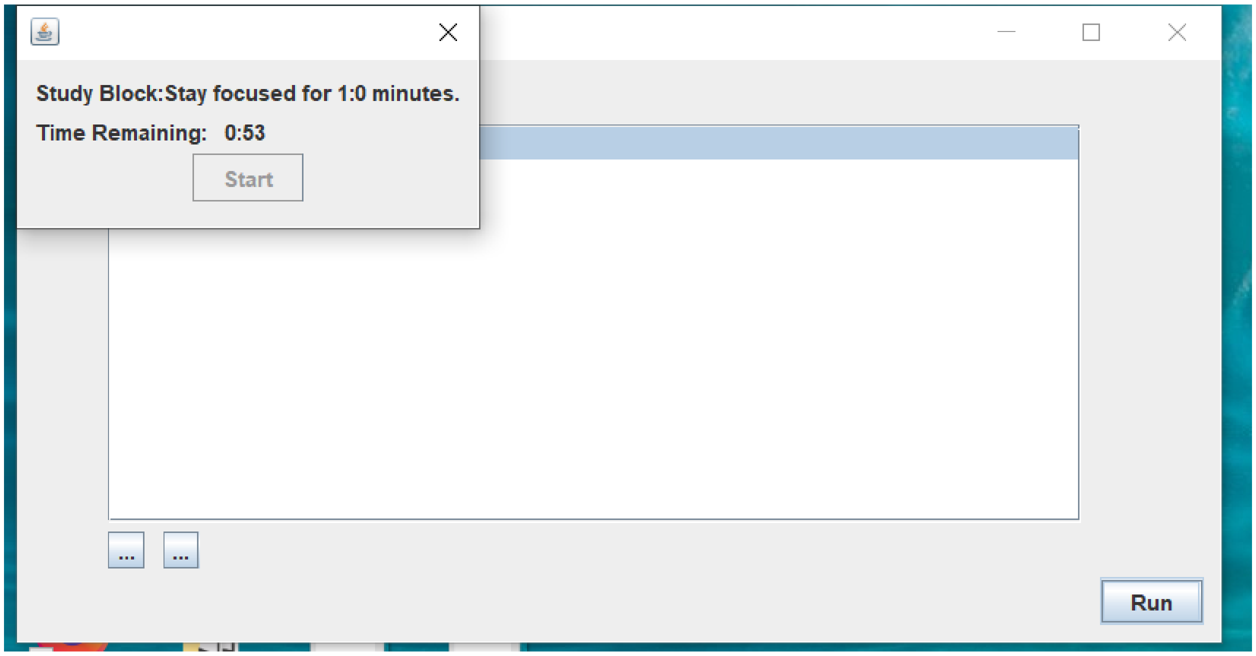

PomodoroApp

Pomodoro Schedules:

class1


Study Block:Stay focused for 1:0 minutes.

Time Remaining: 1:0


Study Block:Stay focused for 1:0 minutes.

Time Remaining: 0:53

Lessons Learned

Jorge Sanchez:

Coming up with a project idea is easy but implementing is a difficult task.

Working in a group is good because of divide and conquer.

Starting your own code is easier and better to get along with than continuing someone else's code because you first have to understand their code, try to see their perspective then catch up and continue the task.

Lilian Onyambu:

I learn the importance of teamwork in divide and conquer to finish tasks.

Learning from each other and knowing one’s strength areas and weaknesses helps to manage the work well.

Good communication is required for work to be efficient and continue as required.

Gregory Givens Jr

I learned more about designing my programs before coding them. It still is difficult for me but I learned more about planning out the program and see how it will function before coding it. I also learned how to express how the program functions using UML