
Scheduled Task - Design Pattern

— By: Taylor D. Williams —

Intent



- What is a Scheduled Task Design Pattern?
 - A type of software design pattern that uses real-time systems
 - Not to be confused with the Scheduler pattern!
- A user can create a task scheduler reliant on the current time of their system

Overall, the pattern ensures that desired operations are performed at certain points in the future

Motivation

- Real time systems find it necessary to have tasks done at specific times, and implementing a system using a scheduled task pattern is one of the simplest approaches in doing so.
- A scheduler pattern would delay access to a resource until absolutely necessary whereas with a scheduled task pattern a task's execution is delayed until a determined time

Structure

Real time scheduled tasks mainly consists of three parts:

1. The task itself
 2. The jobs that schedule the task: when the task will run and for how long
 3. The job registry that executes the task
- Usually, systems that utilize the scheduled task pattern are implemented using cron on Linux servers

Structure cont.

This is the common configuration syntax regarding scheduling:

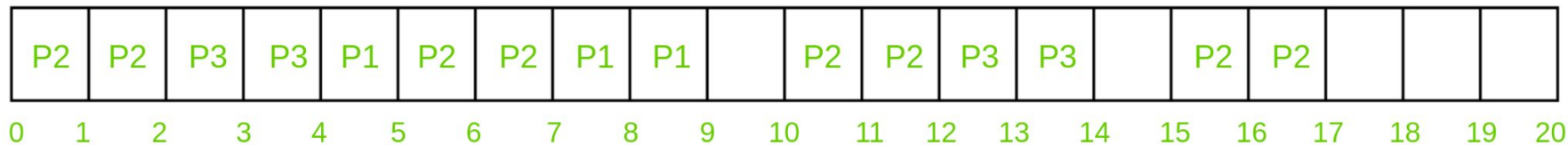
The diagram illustrates the structure of a cron job entry. It shows a sequence of five asterisks (*) followed by a vertical bar and the text <script/command_to_execute>. Five lines with arrows point from the labels to the corresponding fields in the sequence:

- minute (0 - 59)(*/n for every n-th minute)(only * means every
- hour (0 - 23)(*/n for every n-th hour)
- day of the month (1 - 31) (*/n for every n-th day)
- month (1 - 12)(*/n for every n-th month)
- day of the week (0 - 6) (Sunday to Saturday;
7 is also Sunday on some systems)

```
* * * * * <script/command_to_execute>
```

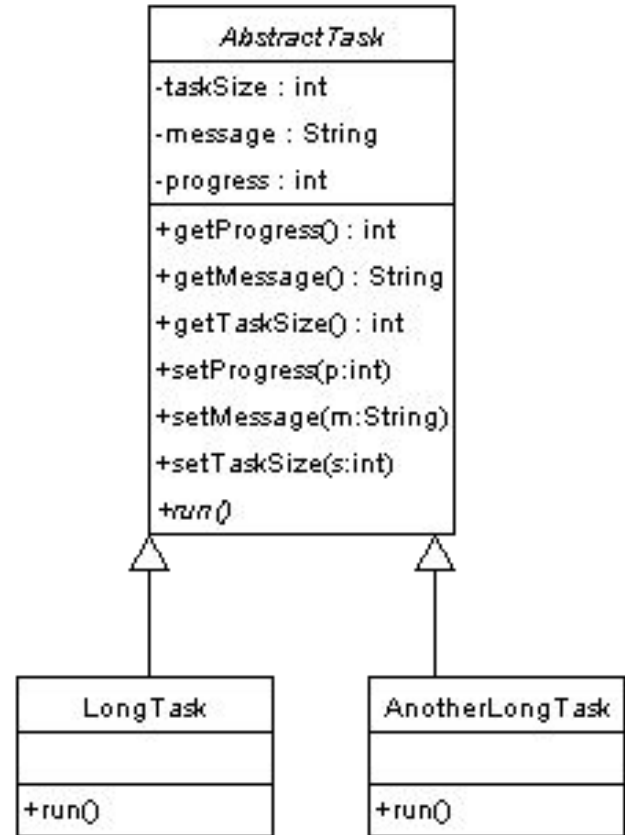
Structure cont.

- A real time scheduler is designed to provide real-time guarantees
 - So once a task is scheduled a certain amount of time, if a user asks to run one more task it will decide what action to take based on the load it is currently carrying
 - If it cannot guarantee that all task can continue running at their desired rates then it will refuse another task being added
- This mainly applies for when the user wants to run longer tasks, and is associated with the RMA (rate monotonic algorithm): a procedure for assigning fixed priorities to tasks
 - Assign the priority of each task according to its period, so that the shorter the period the higher the priority



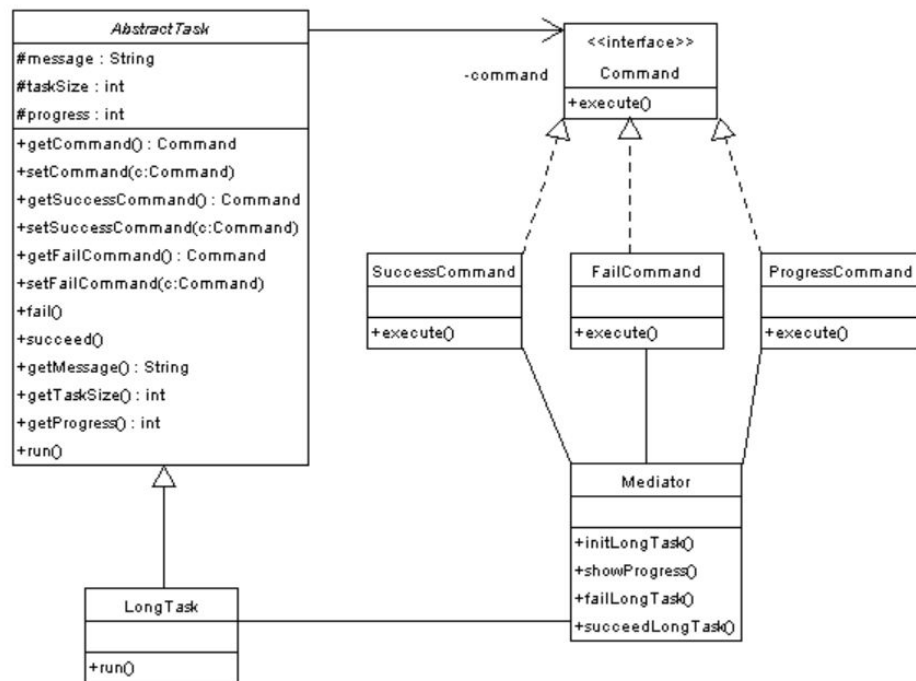
Basic UML Diagram for Task Pattern

- An overall class for the tasks, that contains the methods:
 - That schedule the tasks, set time in which they will run.
 - And the size of the tasks, how long the task will run
- And every task, that extends the main class and overrides the run method that other objects call to start the tasks



Advanced UML diagram for Task Pattern

- Includes an interface that displays commands to the user
- A mediator
 - that tracks the status of all the tasks, informing a user if the task is:
 - currently running,
 - was successful,
 - or failed



Consequences

There are some common issues that can happen when implementing scheduled task design patterns:

- When having tasks running concurrently, an OS can find its processes becoming slower and slower
- When implementing with cron, user's should not expose their cronjobs to the internet
- Not all tasks the user wants done in a given time frame can possibly be achieved depending on priority or if the scheduler implemented can guarantee the task's execution

Known Uses

Scheduled tasks are used in many applications today, some common examples of which include:

- Apps that set alarms or reminders
- The option to send an email at a specific time
- Setting a time for your computer or phone to update
- Restricting screen time on a device
- Using a calendar application to notify user when a date occurs

Demo

Code

The main class for the overall task - Alarm System

```
#class AlarmSystem
class AlarmSystem:

    #Display user's alarms to them
    def displayAlarm():
        #Welcome user message
        print("**** Welcome to Alarm System ****")

        #Stores the user's alarm names
        userAlarms = ["Work", "School", "Homework"]
        [print(i) for i in userAlarms]

    # returns all class tasks of the user's alarms, every task extends AlarmSystem and overrides the execute method
    def execute():
        return Work, School, Homework

    def getMessage():
        error_message = "Error has occurred when setting alarm, re-open Alarm System and try again!!"
        print(error_message)
```

Code

The classes that extend the Alarm System, the alarm tasks Work, School, and Homework

```
#class for task work, function to define execution time and when
class Work(AlarmSystem):
    def execute():
        print("Time to go to Work!")

    #alarm will go off every 20 seconds
    schedule.every(20).seconds.do(execute)

#class for task school, function to define execution time and when
class School(AlarmSystem):
    def execute():
        print("Time to go to School!")

    #alarm will go off every 15 seconds
    schedule.every(15).seconds.do(execute)

#class for task homework, function to define execution time and when
class Homework(AlarmSystem):
    def execute():
        print("Time to start homework!")

    #alarm will go off every 10 seconds
    schedule.every(10).seconds.do(execute)
```

Code

The main function to implement the scheduled task system

```
#Main program to implement
if __name__ == '__main__':
    AlarmSystem.displayAlarm()

    user_input = input("Do you wish to activate your alarms? Yes or No: ")

    if(user_input == "Yes" or "yes"):
        AlarmSystem.execute()

    elif(user_input == "No" or "no"):
        print("Exiting.....")
        exit()

    while True:
        schedule.run_pending()
        time.sleep(1)

    while False:
        AlarmSystem.getMessage()
```



Demo - Output

```
Command Prompt - python  X + v
Microsoft Windows [Version 10.0.22621.1265]
(c) Microsoft Corporation. All rights reserved.

C:\Users\denis> cd demo

C:\Users\denis\demo> python cis476demo.py
**** Welcome to Alarm System ****
Work
School
Homework
Do you wish to activate your alarms? Yes or No: yes
Time to start homework!
Time to go to School!
|
```

Demo - Output

```
Command Prompt - python  +   
Microsoft Windows [Version 10.0.22621.1265]  
(c) Microsoft Corporation. All rights reserved.  
  
C:\Users\denis> cd demo  
  
C:\Users\denis\demo> python cis476demo.py  
**** Welcome to Alarm System ****  
Work  
School  
Homework  
Do you wish to activate your alarms? Yes or No: yes  
Time to start homework!  
Time to go to School!  
Time to go to Work!  
Time to start homework!  
Time to go to School!  
Time to start homework!  
Time to go to Work!  
Time to start homework!  
Time to go to School!  
Time to start homework!
```


Resources

<http://wiki.c2.com/?ScheduledTask>

https://en.wikipedia.org/wiki/Scheduled-task_pattern

<https://subscription.packtpub.com/book/application-development/9781785887130/5/ch05lv1sec61/scheduled-task-pattern>

Recorded video link:

https://youtu.be/yvo2WJM_xk

https://www.developerdotstar.com/mag/articles/troche_taskpattern.html

<https://www.geeksforgeeks.org/how-to-schedule-python-scripts-as-cron-jobs-with-crontab/?ref=rp>

<https://www.geeksforgeeks.org/python-schedule-library/>