**Lab 3—Behavioural Design Pattern**

1. **Template Design Pattern**

* The design pattern is used where you have a few problems to solve, and these solutions have the same overall structure, but different details. For example, say you want to test similar devices. The structure of testing these devices may be same, for example, conducting similar tests.
* However, the way these tests are conducted, it can be different. So in this case, what you can do is to create a template class for testing each device.
* In this class, you create some methods to conduct different tests, these methods are the same for all devices. However, the implementation of carrying out these tests depends upon the actual device to test.

A picture containing text

Description automatically generated

* So that template design pattern asks to create an abstract class which works as a template for the overall operation. In this class, you create the methods for high level steps, these methods are also known as template methods.
* So here we have this method test and the method create report. These are our main high level steps known as template methods where you define the abstract methods that act as the helper methods to execute the template methods.

A picture containing diagram

Description automatically generated

==================================================================================

**devicetests.py**

If we want to conduct some test on different devices, we can create an abstract class for the device testing. So, our abstract class is DeviceTest and in this class, we have two main methods, the template methods. One is this test method, and another create reporting method. These methods will remain same in the subclasses.

Text

Description automatically generated

Besides that, we have certain abstract methods which are this electric test, mechanical test, video test and audio test. If you see in Fig 2, these all methods are abstract methods, they will be overwritten in the subclasses for each device. They will specify the steps to conduct these all-different tests.

Text

Description automatically generated

Fig 2

Similarly, we have one template method to create the report. The steps are defined for creating the report i.e., we need to gather the results, clean up the results, create the summary, and then we need to print the report. These steps are conducted, that will differ in the subclasses that will depend upon the device type. So that is why these methods are, again, abstract methods.

So, this class devicetest (in Fig 1) in an abstract class, and this represents the template for conducting the test in different devices. Once you have defined this template, then you can create another class for one particular device test.

==================================================================================

**device1test.py**

If we move to this particular file device1test.py , we have created one class for testing the device. We do not need to override the template methods. They will remain in the base class, only thing is we need to define the methods for conducting the test, the steps that is the helper methods or hooka methods.

So we are having one method for conducting the mechanical test in case of device1, we may be having two steps. For electrical test, we may be having three steps here for radio test. We have only one step for audio test. So similarly, you need to just define what all the concrete steps will be there for conducting these tests on device1.

Text

Description automatically generated

Fig 3

Similarly, when you are testing device2, you will create another class for say class device2test and in that class, the steps detail will be different. They will depend upon the device2 and on the same lines.

We can say that the steps for creating the report i.e., gathering the results, cleaning up the results and printing, they will depend upon the device types. So, they are all hooka methods, they are all helper methods, and they need to be defined in this particular class.

**Summary**

What we have done is we have created a one template class, which is an abstract class. In this abstract class, template methods are final i.e., to test the device and create the report. Only thing is the implementation detail of these steps are not given. They are defined in that child class that is for a particular device.

==================================================================================

main.py

Here you just need to create an instance of the child class that is a device1test, and then simply call the template method of that particular class. Test and create report.

Graphical user interface, text

Description automatically generated with medium confidence

Fig 4

=================================================================================

Exercise:

Do same practice considering we have more devices such as device2test? Explain and write completely new or integrate in existing code of device1.

Implement other behavioural design patterns in Python.