ESKİŞEHİR TEKNİK ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ

BIM492
Design Patterns

Project 3

1 - Statement of Work

I will create an example of “**State Pattern**”. My project is about ATM machines. ATM maschines works with states. First you need to insert your card then you need to insert pin. After that you can import or export money to machine.

I have a **ATMState**interface. ATMState have insertCard, ejectCard, insertPin and requestCash methods

I have 4 state. NoCash, NoCard, HasPin and HasCard.

That’s all!

2 - Design Pattern

So I will create a project works like this idea and I choose the **State Pattern**for this idea.

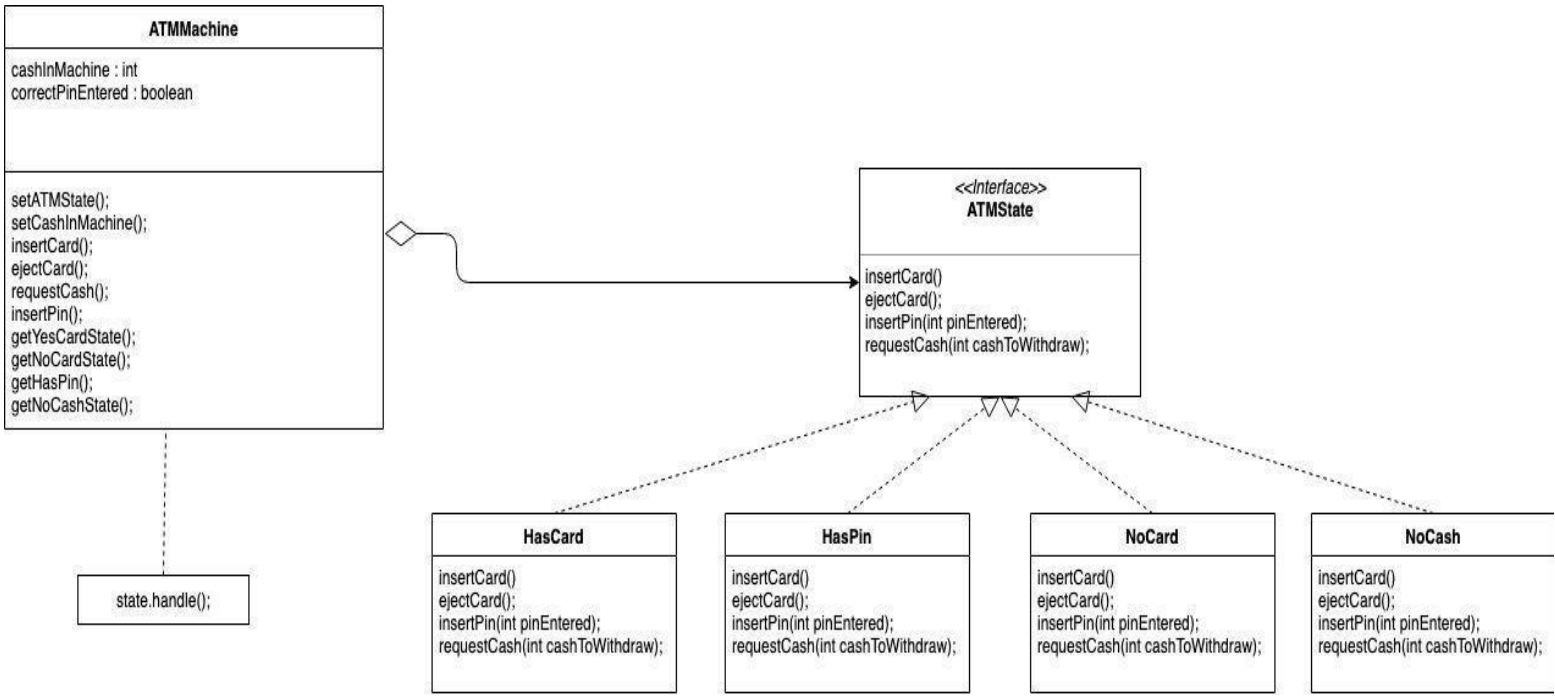
The State Pattern is a behavioral design pattern which allows an object to alter its behavior when its internal state changes. The object will appear to change its class.

In state pattern we encapsulate the state into separate classes and delegate the work to be done to the object representing the current state and behaviour changes along with the internal state which will be represented by the current state. Over time the current state changes across the set of state objects to reflect the internal state of the context, so the context behaviour changes over time as well. The client usually knows very little about the state objects.

This pattern is similar to the **strategy design pattern** but the intent of the two patterns are **different**. With **Strategy**, the client usually specifies the strategy object that the context is composed with. Now, while the pattern provides the flexibility to change the strategy object at runtime, often there is a strategy object that is most appropriate for a context object.

In general Strategy Pattern is a flexible alternative to subclassing. if you use inheritance to define the behaviour of a class, then you’re stuck with that behaviour even if you need to change it. With Strategy you can change the behaviour by composing with a different object. And State Pattern is an alternative to putting lots of conditionals in your context by encapsulating the behaviours within state objects, you can simply change the state object in context to change its behaviour.

3 - UML Diagram



4 - Research

State patternis one of the **behavioral**design pattern. State design pattern is used when an Object changes its behavior based on its internal state. If we have to change behavior of an object based on its state, we can have a state variable in the Object and use if-else condition block to perform different actions based on the state.

State pattern is used to provide a systematic and lose-coupled way to achieve this through Context and State implementations.

With State pattern, the benefits of implementing polymorphic behavior are evident, and it is also easier to add states to support additional behavior.

In the State design pattern, an object’s behavior is the result of the function of its state, and the behavior gets changed at runtime depending on the state. This removes the dependency on the if/else or switch/case conditional logic. For example, in the TV remote scenario, we could have also implemented the behavior by simply writing one class and method that will ask for a parameter and perform an action (switch the TV on/off) with an if/else block.

The State design pattern also improves Cohesion since state-specific behaviors are aggregated into the ConcreteState classes, which are placed in one location in the code.

In the other hand, The State design pattern can be used when we need to change state of object at runtime by inputting in it different subclasses of some State base class. This circumstance is advantage and disadvantage in the same time, because we have a clear separate State classes with some logic and from the other hand the number of classes grows up.

5 - Implementation

Codes in the Github.