Refresh:

Structure example.

Diagram

Description automatically generated

The Context can have a number of internal States, whenever the request() method is called on the Context, the message is delegated to the State to handle. The State interface defines a common interface for all concrete states, encapsulating all behaviour associated with a particular state. The ConcreteState implements it's own implementation for the request. When a Context changes state, what really happens is that we have a different ConcreteState associated with it.

This is all quite similar to the Strategy pattern, except the changes happen at runtime rather than the client deciding. State saves you from lots of conditional code in your Context: by changing the ConcreteState object used, you can change the behaviour of the context.

You should use the State pattern when the behaviour of an object should be influenced by it's state, and when complex conditions tie object behaviour to it's state.

Exercise:

Use the state of an mp3 player to watch the state pattern in action. First, we set up a context for our mp3 player, that is the missing class at our StateExercise.zip

Check the complete solution at:

[Java User Input (Scanner class) (w3schools.com)](https://www.w3schools.com/java/java_user_input.asp)