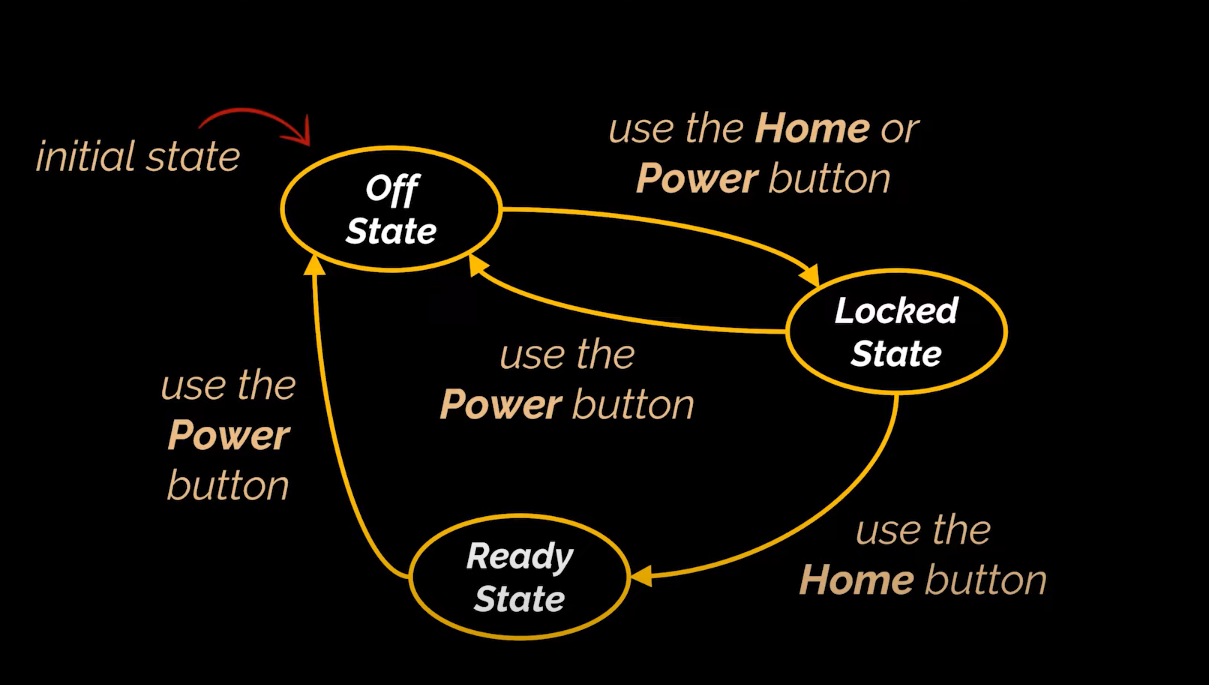
At any given moment there is a finite number of states in which the program can be,and within any unique state the program behaves differently.

A state can switch the state of our program to another one or not at all.

These switching rules are called transitions , are finite and predetermined.

For example a smart phone can be in the following states.



A State Pattern says that "the class behavior changes based on its state".

In State Pattern, we create objects which represent various states and a context object whose behavior varies as its state object changes.

The State Pattern is also known as Objects for States.

This type of design pattern comes under behavior pattern.

Usage:

When the behavior of object depends on its state and it must be able to change its behavior at runtime according to the new state.

It is used when the operations have large, multipart conditional (if-else or switch ) statements that depend on the state of an object.

This is very similar to Strategy design pattern .

State Pattern *Context* is the class that has a State reference to one of the concrete implementations of the *State*. Context forwards the request to the state object for processing.

Example :

public abstract class State {  
 protected Phone phone ;  
  
 public State(Phone phone){  
 this.phone = phone ;  
 }  
  
 public abstract void pressPowerButton() ;  
  
  
}

public class OffState extends State{  
  
 public OffState(Phone phone ){  
 super(phone);  
 }  
  
 @Override  
 public void pressPowerButton() {  
 phone.setState(new OnState(phone));  
 phone.turnOn();  
 }  
}

public class OnState extends State{  
  
 public OnState(Phone phone ){  
 super(phone);  
 }  
  
 @Override  
 public void pressPowerButton() {  
  
 phone.setState(new OffState(phone));  
 phone.turnOff();  
 }  
}

// Here Phone will act as the context

public class Phone {  
 private State state ;  
  
 public Phone(){  
 state = new OffState(this) ;  
 }  
  
 public State getState(){  
 return this.state ;  
 }  
  
 public void setState(State state ){  
 this.state = state ;  
 }  
  
 public void turnOn(){  
 System.*out*.println("Turning On");  
 }  
  
 public void turnOff(){  
 System.*out*.println("Turning Off");  
 }  
}

public class StateDesignPattern {  
  
 public static void main(String[] args) {  
  
 Phone phone = new Phone() ; *// By default it is in Off state* phone.getState().pressPowerButton(); *// It will go to On state* phone.getState().pressPowerButton(); *// // It will go to Off state again* }  
}

Output :

Turning On

Turning Off

Process finished with exit code 0