

Código Fuente

```
1 package States;
2 /**
3  * @author Victor Lavalle
4  */
5 public class Turnstile {
6
7     public void lock(){
8         System.out.println("Locked\n\n");
9     }
10
11     public void unlock(){
12         System.out.println("Unlocked\n\n");
13     }
14
15     public void thanks(){
16         System.out.println("Thanks!\n\n");
17     }
18
19     public void alarm(){
20         System.out.println("Sounding alarm...!\n\n");
21     }
22
23     public void resetAlarm(){
24         System.out.println("Alarm stopped\n\n");
25     }
26
27 }
```

```
1 package States;
2 /**
3  * @author Victor Lavalle
4  */
5 public class TurnstileFSM extends Turnstile {
6     private States state;
7
8     public TurnstileFSM(){
9         state = new Locked();
10    }
11
12    public void setState(States s){ state = s;}
13
14    public void coin(){ state.coin(this); }
15
16    public void pass(){ state.pass(this); }
17
18    public void reset(){ state.reset(this); }
19
20    public void ready(){ state.ready(this); }
21
22 }
```

```

1 package States;
2 /**
3  * @author Victor Lavalle
4  */
5 public interface States {
6     public void coin(TurnstileFSM f);
7
8     public void pass(TurnstileFSM f);
9
10    public void reset(TurnstileFSM f);
11
12    public void ready(TurnstileFSM f);
13 }

```

```

1 package States;
2 /**
3  * @author Victor Lavalle
4  */
5 public class Locked implements States{
6
7     @Override
8     public void coin(TurnstileFSM f) {
9         f.setState(new Unlocked());
10        f.unlock();    }
11
12    @Override
13    public void pass(TurnstileFSM f) {
14        f.alarm();
15        f.setState(new Violation());    }
16
17    @Override
18    public void reset(TurnstileFSM f) { }
19
20    @Override
21    public void ready(TurnstileFSM f) { }
22 }

```

```

1 package States;
2 /**
3  * @author Victor Lavalle
4  */
5 public class Unlocked implements States{
6     @Override
7     public void coin(TurnstileFSM f) {
8         f.thanks();    }
9
10    @Override
11    public void pass(TurnstileFSM f) {
12        System.out.println("Passing[...]");
13        f.setState(new Locked());
14        f.lock();    }
15
16    @Override
17    public void reset(TurnstileFSM f) { }
18
19    @Override
20    public void ready(TurnstileFSM f) { }
21 }

```

```

1 package States;
2 /**
3  * @author Victor Lavalle
4  */
5 public class Violation implements States {
6
7     @Override
8     public void coin(TurnstileFSM f) { }
9
10    @Override
11    public void pass(TurnstileFSM f) { }
12
13    @Override
14    public void reset(TurnstileFSM f){
15        f.resetAlarm();
16    }
17
18    @Override
19    public void ready(TurnstileFSM f){
20        f.resetAlarm();
21        f.setState(new Locked());
22        f.lock();
23    }
24
25 }

```

```

1 package Launcher;
2 /**
3  * @author Victor Lavalle
4  */
5 import States.*;
6 import java.util.Scanner;
7
8 public class Main {
9
10    public static void Menu(){
11        System.out.println("Select an Option:"
12            + "\n _____"
13            + "\n|1. Insert Coin   |"
14            + "\n|2. Pass           |"
15            + "\n|3. Reset          |"
16            + "\n|4. Ready          |"
17            + "\n|0. Exit           |"
18            + "\n -----"
19            + "\n>>");
20    }
21
22    public static void main(String[] args) {
23
24        Scanner scanAction = new Scanner(System.in);
25        TurnstileFSM turnstile = new TurnstileFSM();
26
27        turnstile.setState(new Locked()); //It starts being locked
28
29        System.out.println(" *Turnstile*");
30
31        while(true){
32            Menu();
33            String action= scanAction.nextLine();
34            switch(action){

```

```
35
36         case "0":
37             System.exit(0);
38         break;
39         case "1":
40             turnstile.coin();
41         break;
42
43         case "2":
44             turnstile.pass();
45         break;
46
47         case "3":
48             turnstile.reset();
49         break;
50
51         case "4":
52             turnstile.ready();
53         break;
54
55         default: System.out.println("Invalid Option!");
56     }
57 }
58 }
59 }
```

Diagrama de Clases

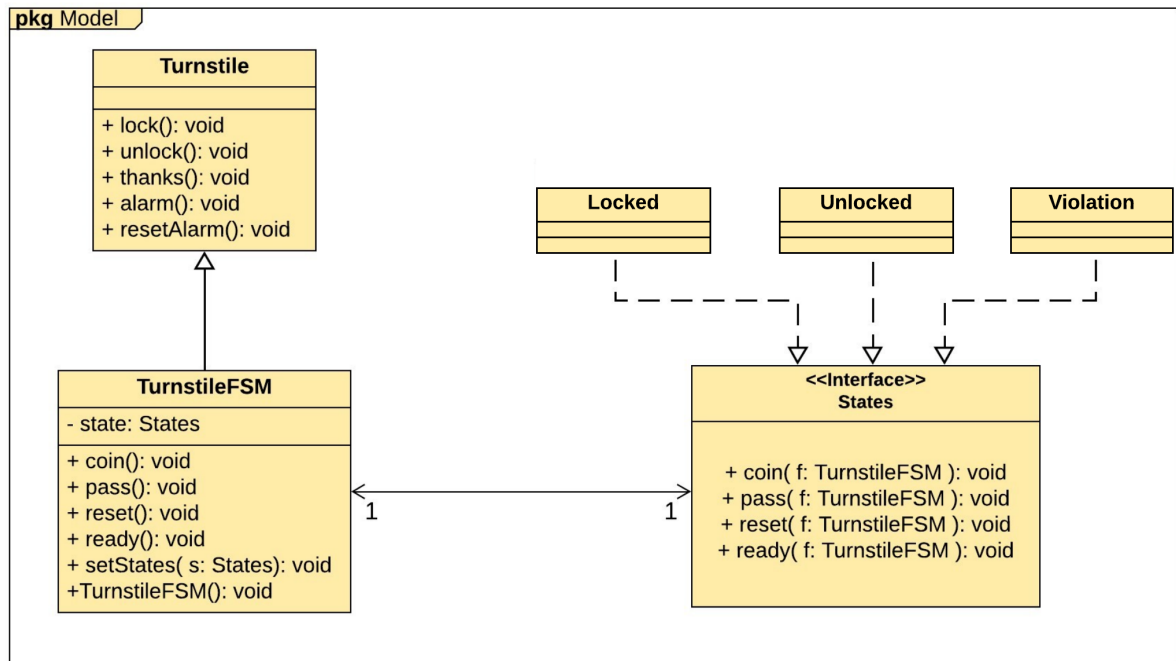
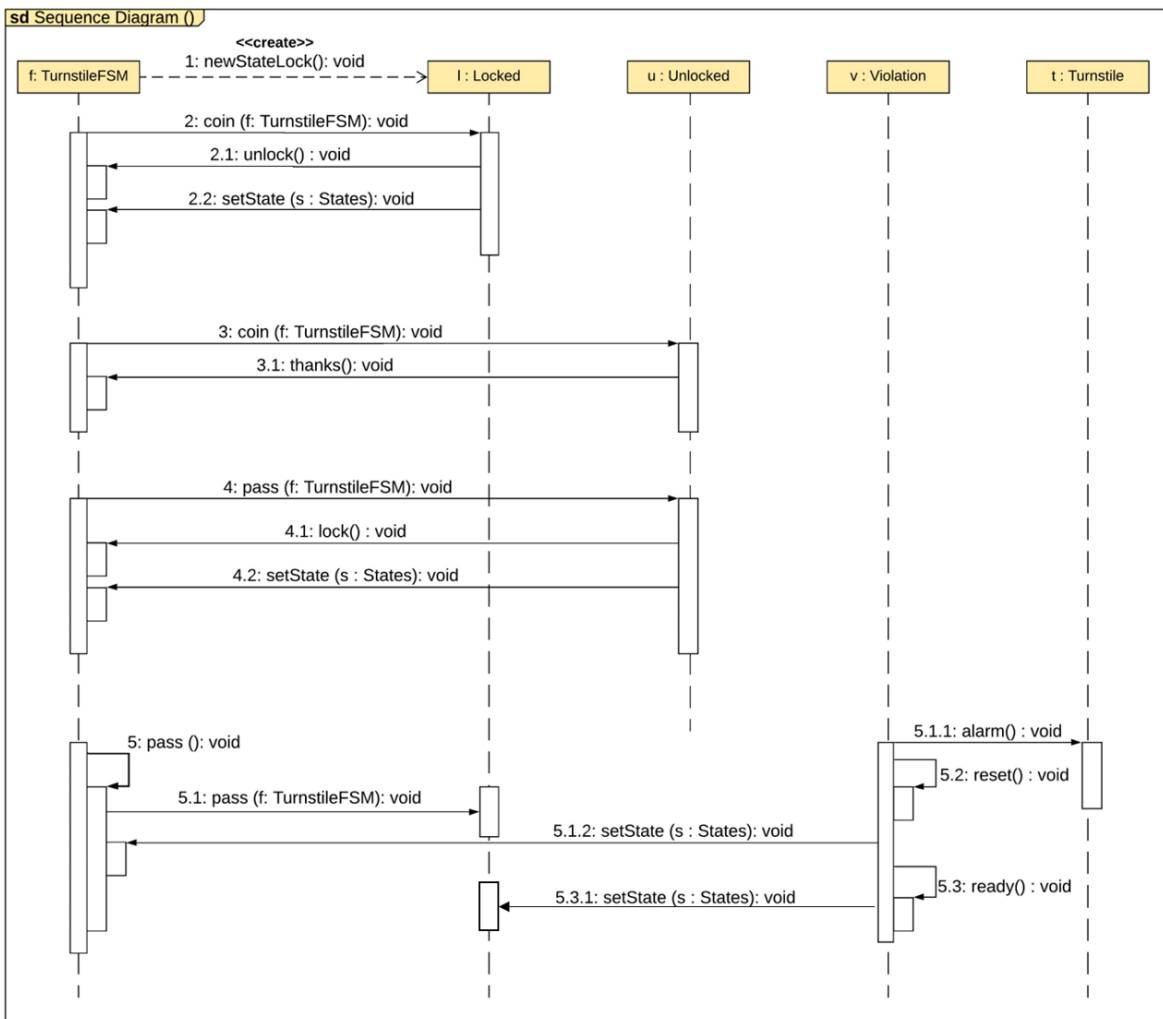


Diagrama de Secuencia



Ejecución del Programa

Ejecución Exitosa

```
run:
    *Turnstile*
Select an Option:

|1. Insert Coin |
|2. Pass       |
|3. Reset      |
|4. Ready      |
|0. Exit       |
-----

>>
1
Unlocked

Select an Option:

|1. Insert Coin |
|2. Pass       |
|3. Reset      |
|4. Ready      |
|0. Exit       |
-----

>>
1
Thanks!

Select an Option:

|1. Insert Coin |
|2. Pass       |
|3. Reset      |
|4. Ready      |
|0. Exit       |
-----

>>
2
Passing[...]
Locked
```

Ejecución cuando se quiere pasar sin haber ingresado una moneda y entra al estado de violación.

```
*Turnstile*
Select an Option:

|1. Insert Coin |
|2. Pass        |
|3. Reset       |
|4. Ready       |
|0. Exit        |
|-----|

>>
2
Sounding alarm...!

Select an Option:

|1. Insert Coin |
|2. Pass        |
|3. Reset       |
|4. Ready       |
|0. Exit        |
|-----|

>>
1

Select an Option:

|1. Insert Coin |
|2. Pass        |
|3. Reset       |
|4. Ready       |
|0. Exit        |
|-----|

>>
2

Select an Option:

|1. Insert Coin |
|2. Pass        |
|3. Reset       |
|4. Ready       |
|0. Exit        |
|-----|

>>
```

*Continuación estando en el estado de violación y como se sale de este.

```
3
Alarm stopped

Select an Option:

|1. Insert Coin |
|2. Pass       |
|3. Reset      |
|4. Ready      |
|0. Exit       |
|-----|

>>
4
Alarm stopped

Locked

Select an Option:

|1. Insert Coin |
|2. Pass       |
|3. Reset      |
|4. Ready      |
|0. Exit       |
|-----|

>>
1
Unlocked

Select an Option:

|1. Insert Coin |
|2. Pass       |
|3. Reset      |
|4. Ready      |
|0. Exit       |
|-----|

>>
2
Passing[...]
Locked
```