



State Machine

Designing components using a finite state machine model.



When State Matters

Some components behave differently depending on what *state* they are in.

Examples:

- ▯ Alarm Clock
- ▯ Calculator
- ▯ Stop Watch
- ▯ Point of Sale (POS) device
- ▯ most *parsers*



Really Simple Example

Stopwatch *behaves differently when it is* RUNNING or STOPPED.

What behavior depends on state?

Easy! just look for methods containing "if (running) ..."



Identify States

Stopwatch states: **RUNNING** and **STOPPED**

RUNNING

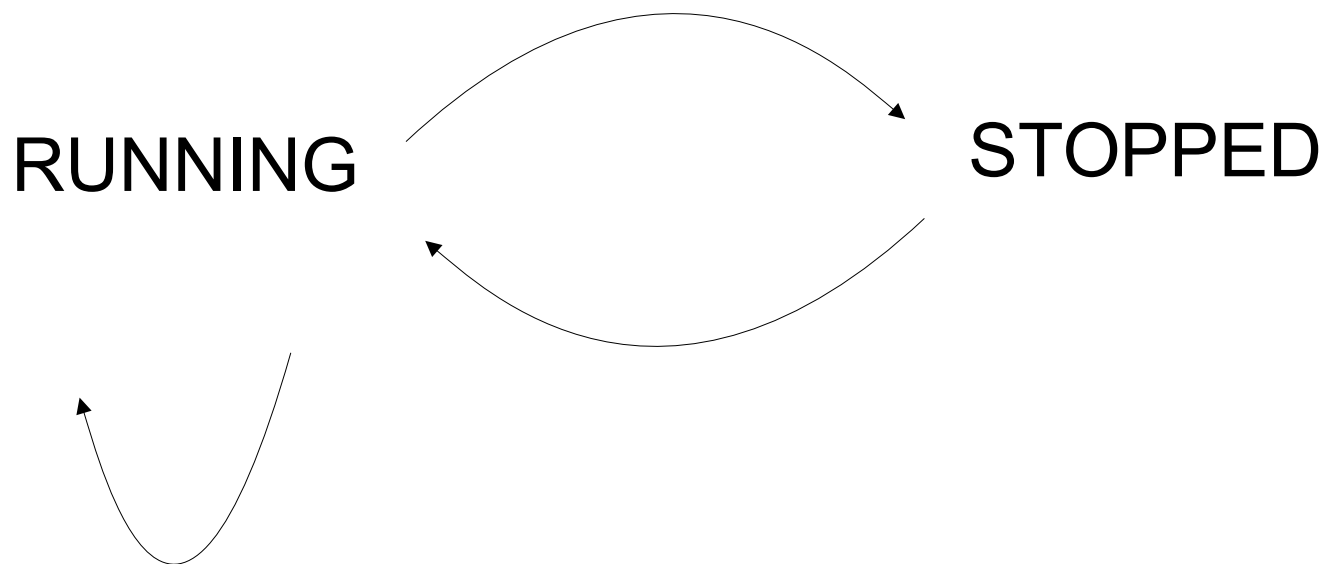
STOPPED



Events

What causes a stopwatch to **change state**?

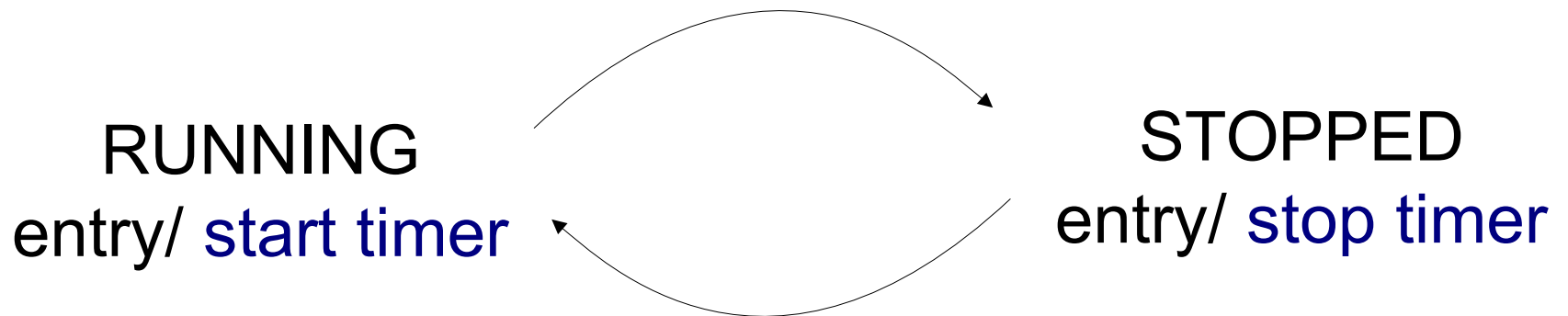
Why not stay in the same state forever?





Action

The stopwatch performs some *action* in response to an event.






Activities

An activity is something that lasts for some time.

An action is (nearly) instantaneous.

In the UI for Stopwatch, "*update display*" is an activity.



Programming a State Machine

Design the state machine first – step by step.

1. Identify the states
2. Identify *events*: external and internally generated
3. Identify actions or activities the state machine performs in response to events or change in state.
4. Draw a diagram.

Finally,

5. Code *state-dependent behavior* using state machine.



What behavior depends on state?

We use boolean `running` to keep track of state.

```
class Stopwatch {  
    private boolean running;  
  
    public void start( ) {  
        if (running) return;  
        startTime = System.nanoTime();  
        running = true; // change state  
    }  
}
```



What behavior depends on state?

```
public double getElapsed() {  
    if (running)  
        return (System.nanoTime()-startTime)  
            * NANOSECONDS;  
    else  
        return (stopTime-startTime)  
            * NANOSECONDS;  
}  
  
public void stop() {  
    if (! running) return;  
    stopTime = System.nanoTime();  
    running = false;  
}
```



The State Variable

We used a `boolean` (`running`) to record the state.
This only works when there are just 2 states.
For more states we need another type of state var.

Consider: a `StopWatch` with `Start`, `Stop`, and `Hold` buttons.

Each button is the source of *event*.

Now there are 3 states.



2 ways to represent state

// use "int" or "char"

```
class Stopwatch {  
    int state;  
    final int STOPPED = 0;  
    final int RUNNING = 1;  
    final int HOLDING = 2;  
    public void start( ) {  
        if (state == RUNNING)
```

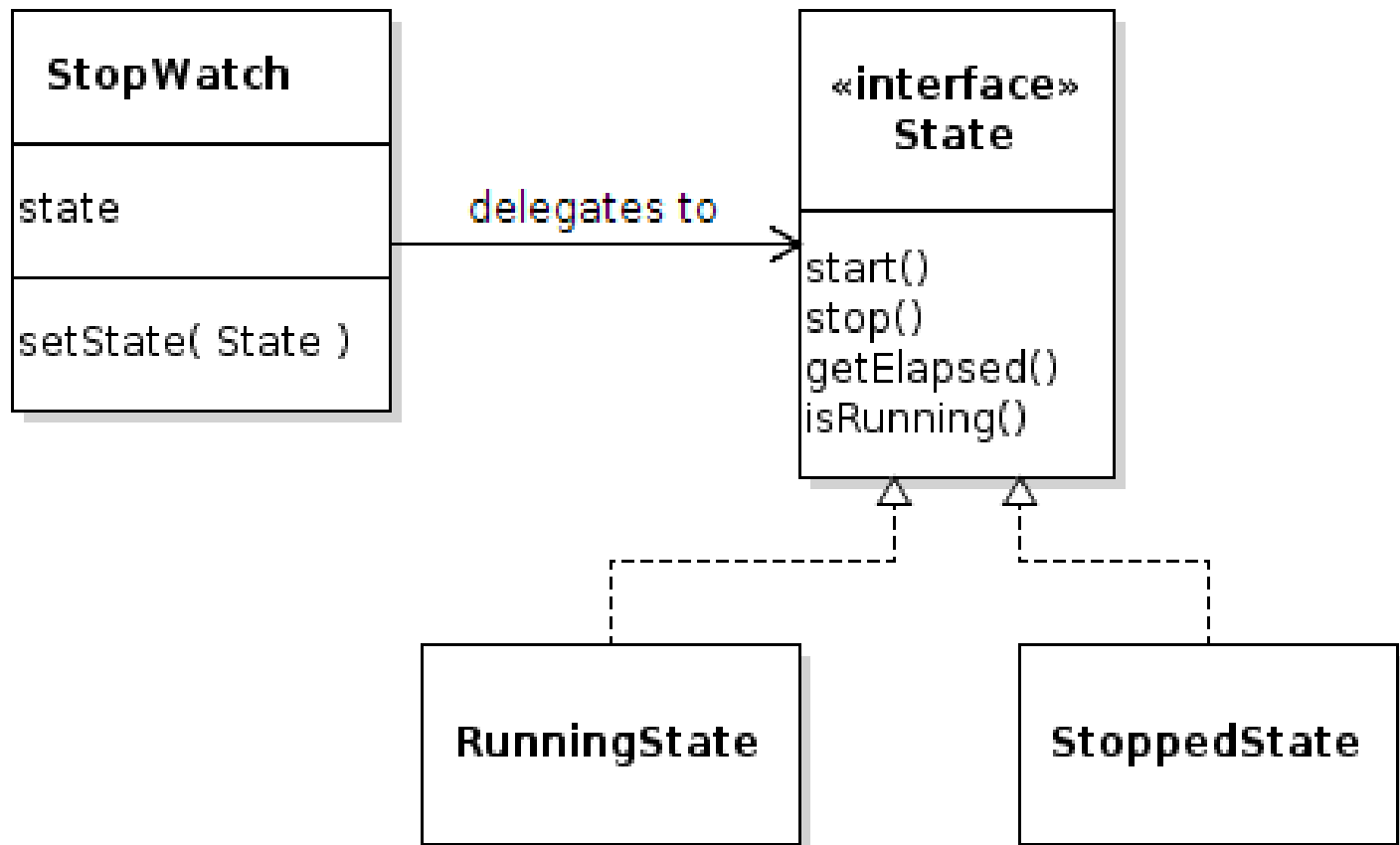
// use an enum

```
public enum State {  
    STOPPED,  
    RUNNING,  
    HOLDING;  
}  
class Stopwatch {  
    State state;  
    public void start() {  
        if (state ==  
RUNNING)
```

The O-O Approach

Use *Objects* to encapsulate state and the behavior that depends on state.

The *context* delegates behavior to state objects.





Delegating Behavior

Delegate means "let someone else do it".
Stopwatch delegates behavior to the state.

```
public class StopWatch {  
    private State state;  
  
    public void start( ) { state.start(); }  
    public void stop()   { state.stop(); }  
    public double getElapsed() {  
        return state.getElapsed();  
    }  
}
```

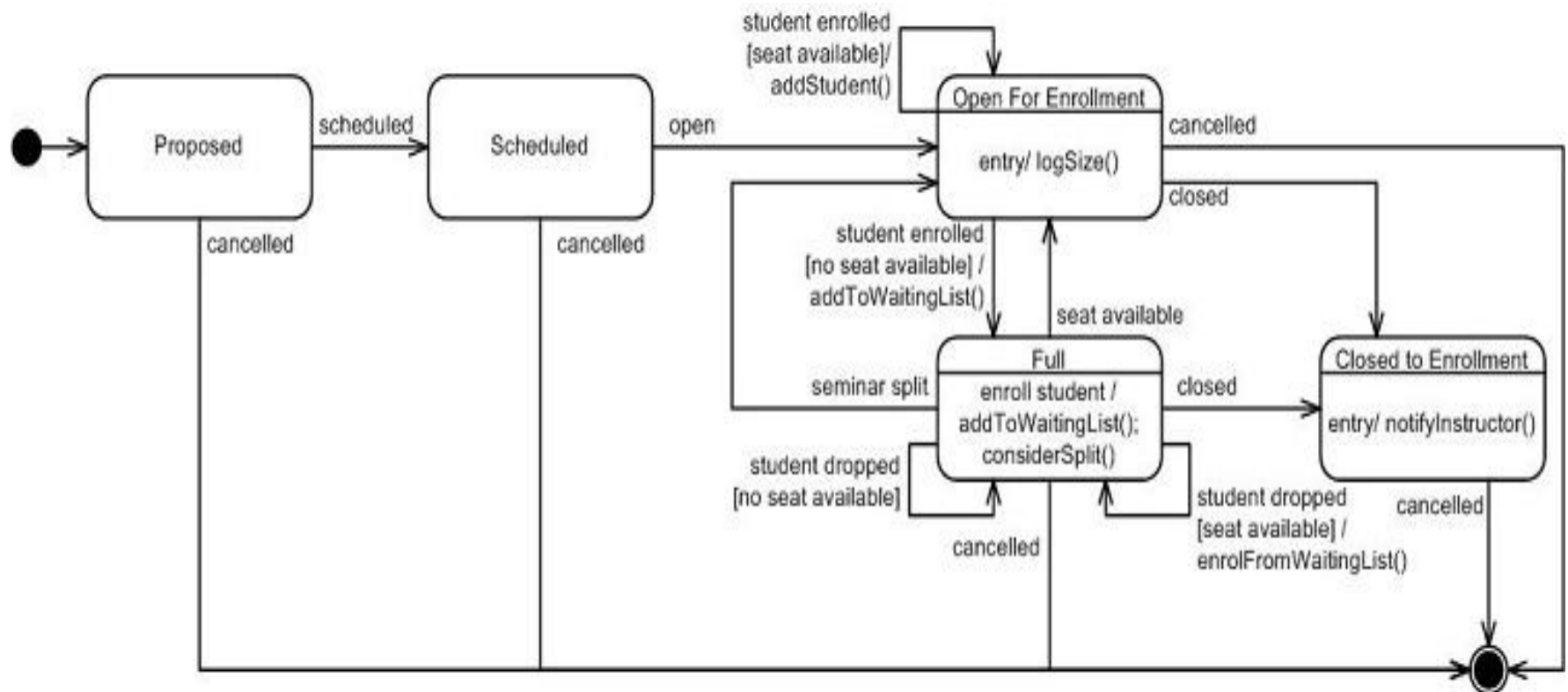


Glue

The *context* (StopWatch) needs a setState method.
The *states* need a *reference* to the *context*.

```
public void setState(State newstate) {  
    this.state = newstate;  
}  
  
// create the states with a reference to  
// the stopwatch  
final State running = new RunningState(this);  
final State stopped = new StoppedState(this);
```

UML State Machine Diagram





UML State Machine Diagram

Read *UML Distilled*, chapter 10.

Also good: *UML for Java Programmers*, chapter 10.



Exercise: Skytrain Ticket Machine

1. What are the states.
2. What are the events.
3. What actions/activities does ticket machine perform?
4. Draw a UML State Machine Diagram.

อัตราค่าโดยสาร Fare Information

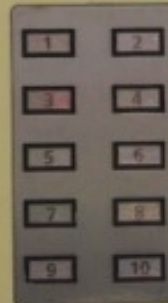
ไปสถานี Zone	1	2	3	4	5	6
ค่าโดยสาร (บาท) Fare (Baht)	15	20	25	30	35	40



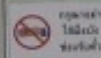
เครื่องจำหน่ายตั๋ว TICKETS

ตั๋วเดินทางเดียวใช้ได้ในวันที่ยี่สิบ
SINGLE JOURNEY TICKETS VALID
FOR DAY OF PURCHASE ONLY

กดปุ่มเลือกสถานี
SELECT ZONE

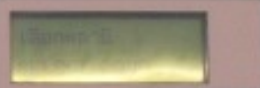


รับตั๋ว
TAKE TICKET



T16

รับเงินทอน
TAKE CHANGE



หยอดเหรียญ
INSERT COIN

วิธีซื้อตั๋ว
OPERATION STEPS

- กดปุ่มเลือกสถานี
SELECT ZONE
- หยอดเหรียญ
INSERT COIN
- รับตั๋ว
TAKE TICKET
- รับเงินทอน
TAKE CHANGE





Exercise: Syllable Counter

Count the syllables in a word.

As a heuristic, we will count *vowel sequences*.

Example:

object = (o)bj(e)ct = 2 vowel sequences

beauty = b(eau)t(y) = 2 vowel sequences

Special cases:

l(a)y(ou)t = *treat "y" as consonant after other vowel*

l(a)the = don't count final "e" if it a single vowel

th(e) = exception. count "e" if it is only vowel

anti-oxident = (a)nt(i)-(o)x(i)d(e)nt "-" is non-vowel



Example Words

How many vowel sequences in these words:

remarkable

selfie

county

coincidentally

she

mate

isn't



Exercise: Calculator

A calculator that behaves like Windows calc.

Use: <http://www.online-calculator.com>

1. What are the states.
2. What are the events.
3. What actions/activities does ticket machine perform?
4. Draw a UML State Machine Diagram.
(not so easy)



PA5: Cheap Digital Clock

cheap digital alarm clock.

Use states!