Design of Object Behavior

The Questions for the Day

How do you determine the actions to executed by an object?
(The methods of the implementing class.)

How do you determine the valid conditions for executing an action?
(The preconditions for invoking a method.)

The Approach

Lexical Analysis + Scenario analysis

Status Attributes

State Machines

Design of Object Behavior

Specifications for a Cheap Microwave Oven

This microwave oven has a powertube, a light, a door, a timer, and just one button. If the door is closed and the oven is not cooking then pushing the button causes the light and the powertube to be turned on for a period of one minute. If the button is pushed while the oven is cooking then the cooking period is extended for one more minute. If the door is opened while the oven is cooking then the cooking period is suspended. If the door is closed while cooking is suspend then cooking is resumed with the duration being the time left on the timer when the door was opened plus one minute. If the button is pushed while the door is open any time remaining on the cooking period is terminated. If a cooking period terminates by the expiration of the timer then the oven becomes idle.

Design of Object Behavior

What are the objects?
- **microwave oven - tangible**
- **powertube - tangible**
- **light - tangible**
- **timer -tangible**
- **button - tangible**
- **door - tangible**

The powertube, the light, the timer, the button and the door are all components of the microwave oven.

The powertube and light objects have two events and two states, on and off and thus will need only on and off methods so we need not consider them any further.

The timer will be a system primitive so we need not consider it and further. We will use whatever behavior and interface the system timer provides.

The door and the button, however, generate <u>events</u> or actions which will be processed by the oven.

So only the oven has complex behavior and thus significant software to be implemented.

Design of Object Behavior

Attributes for Object MicrowaveOven

1. Components as Attribute.
Each of the components, powertube, light, beeper, timer, button and door will be attributes of the oven and instance variables of the resulting class.
2. Status Attribute - We need an attribute to record the state of the oven. Let us call this attribute "status" or "current state." In fact all objects with more than one or two actions will need a status variable.

Attributes for Other Objects:

1. Status - on or off
2. Unique ID

Timer

We need to know only the interface of the system supplied timer object.

Design of Object Behavior

Event/action - An event is a result of some action defined at the object level which results in a interaction among objects or within an object. An event always causes some other action to occur.

1. **Pushing the button creates an event.**
2. **Opening the door creates an event.**
3. **Closing the door creates an event.**
4. **The timer expiring creates an event.**

Every externally generated event that is directed to an object requires a public method in the implementing class to execute the action required by the event. We know that we will have at least 4 public methods in the class MicrowaveOven.

There is a Java class Event which has highly similar semantics to the definition for an event used here. It is a part of the java.awt package and is used for buttons, etc. in GUIs.

Design of Object Behavior

Example:

Pushing the button (when the door is closed and the oven is idle) causes:

a. The light to be turned on.
b. The powertube to be turned on
c. The timer to be set to expire in one minute.

This means that we need a method in class MicrowaveOven to execute this action.

Pushing the button when the oven is cooking causes:

a. One minute to be added to the duration before the timer expires.

Therefore we need a method to execute this action.

The action to be taken on generation of an event depends on the state of the object which is the target of the event. Which is why we need a status variable for each object with multiple possible states. We must select an appropriate action on the basis of the state of the object.

What should happen when the button is pushed when the door is open?

Design of Object Behavior

To realize the functionality of an object it is often necessary that events be generated in a specific order. This implies that the object will pass through a specific sequence of states while serving its purpose.

Opening the door and then pushing the button is not a useful sequence of events.

Furthermore some events should be ignored or taken as errors if they occur when an object is in some specified state.

A state machine is used to specify the valid sequence of states. A state machine for an object is simply a sequence of states with the transitions between states being invoked by occurrence of events targeting the object. An action is associated with each state.

Design of Object Behavior

Scenario Analysis for the Microwave Oven

Precondition: Oven idle with door closed.

Initiator	Action	Part.	Service	Prior State	New State
user	push button	oven	turn on light and powertube, set timer for one minute, change oven state	idle, door closed	first cooking period
user	push button	oven	add one minute to timer	first cooking period	cooking
timer	expires	oven	turn off oven	cooking	cooking completed
user	open oven door	oven	change oven state	cooking completed	idle. door open

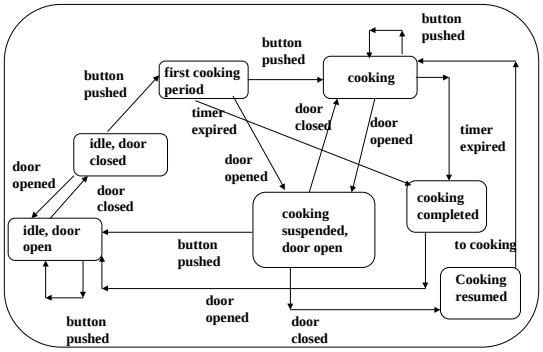
Design of Object Behavior

Scenario Analysis for Microwave Oven

Precondition: Oven idle with door closed.

Initiator	Action	Part.	Service	Prior State	NewState
user	push button	oven	turn on light and powertube, set timer for one minute, change oven state	idle, door closed	first cooking period
user	push button	oven	add one minute to timer	first cooking period	cooking
user	open door	oven	turn off light and powertube	cooking	idle, cooking suspended
user	close door	oven	turn on light, powertube	idle, cooking suspended	resume cooking
oven	return to cooking	oven	add minute to timer	resume cooking	cooking

Design of Object Behavior



Design of Object Behavior

Definition of Moore State Machines (State Model)

1. Let us associate an action with each state. The action associated with the "first cooking period" state is to turn on the powertube and the light and to set the timer for one minute.

2. Let us stipulate that execution of the action will be initiated when the state is entered.

3. The first thing the action should do is to determine whether or not it should execute its functionality at all (evaluate the precondition), should ignore the event that caused the transition or report an exception or error.

4. The second thing the action associated with a given state does (assuming the precondition is satisfied) is to change the object instance state to the new state.

5. The action is then executed and the state machine halts and waits for the next action. (The implementing method returns to the invocation site and the class instance is left in the new state.)

Design of Object Behavior

	idle door closed	idle door open	first cooking period	cooking	cooking suspended, door open	cooking completed	Resume cooking
button pushed	First cooking period	ignore	cooking	cooking	idle door open	ignore	can't happen
door opened	idle door open	can't happen	Cooking suspended, door open	Cooking suspended, door open	can't happen	idle door open	can't happen
door closed	can't happen	idle door closed	can't happen	can't happen	cooking	can't happen	can't happen
timer expires	can't happen	can't happen	idle door closed	idle door closed	can't happen	can't happen	can't happen

Design of Object Behavior

Why bother with all of this rigmarole?

1. It is wonderful documentation.
   Pictures are much better than words.

2. It leads to a uniform implementation structure.
   All classes will have the same structure and anyone can read them.
   Furthermore you will be able to read them yourselves.

3. It is the basis for an algorithm for designing classes which will always work.

4. It eliminates wasted thought.
   If you don't have to think about structure you can think about content.

5. It facilitates debugging.
   Preconditions are naturally defined.

6. The code will be robust to external errors.
   Errors are detected before they corrupt the program state.

Design of Object Behavior

How does this model of execution map to an implementation in Java?

1. Define a public method for each event. If the event has only one possible action then implement the action in this method.

2. If the action to be executed upon the occurrence of an event is state dependent then define a private method for each action and invoke the appropriate method for the current state of the object from the public method.

Design of Object Behavior

```
Public class MicrowaveOven
{       private Powertube powertube;
        private Light light;
        private Timer timer;
        private String currentState;
        public MicrowaveOven(String initialState)
        {       currentState = initialState;  }
/* The method signatures and implementations follow. */
}
```


Design of Object Behavior

```
public void buttonPushed() {
        if (currentState = "idle door closed")
        then {this.firstCookingPeriod();
        return;}
        if (currentState =" idle door open")
        then {return;}
        if (currentState= "firstCookingPeriod")
        then {this.cooking();
        return;}
        if (currentState = "cooking")
        then {this.cooking();
        return;}
        if (currentState = "idle cooking suspended")
        then {return;}
        /* Throw appropriate Exception. */
        return; }
```


Design of Object Behavior

```
public void doorOpened()
{       if (currentState = "idle door closed")
        then {this.idleDoorOpened();
        return;}
        if (currentState = "first cooking period")
        then {this.idleCookingSuspended();
        return; }
        if (currentState = "cooking")
        then {this.idleCookingSuspended();
        return;}
        if (currentState= "idle door open")
        then { /* Throw appropriate Exception.*/;
        return;}
}
```


Design of Object Behavior

```
public void doorClosed()
{       if (currentState = "idleDoorOpen")
        then {this. IdleDoorClosed();
        return;}
        if (currentState = "cookingSuspended")
        then {this.resumeCooking();
        return;}
        / Throw appropriate Exception.*/
        return;
}
```

Design of Object Behavior

```
Public void TimerExpires()
{       if (currentState = "first cooking period")
        then {this.cookingCompleted();
        return;}
        if (currentState = "cooking")
        then {this.cookingCompleted();
        return;}
}
}
```


Design of Object Behavior

Private Methods - one for each state of the state model.

```
private void idleDoorClosed()
{ /* Set state to "idle door closed." */ }
private void idleDoorOpen()
{/* Set state to "idle door open." */
 /* Clear Timer.*/ }
private void firstCookingPeriod()
{ /* Set state to "first cooking period." */
  /* Turn on powertube and light and set timer. */}
private void cooking()
{ /* Add one minute to timer. */ }
private void idleCookingSuspended()
{ /* Suspend powertube and light. }
private voidresumeCooking()
{ /* Turn on powertube and light." }
private void cookingCompleted()
{/* Turn off powertube and light. */
```