

# Project Title: UrbanGarden

Mohib Abbas Sayed – 2103158

Hamza Sayyed – 2103159

Om Shete – 2103163

# EXPERIMENT. NO: 05

**Aim:** Develop Activity and State Diagram for the project (Smart Draw, Lucid Chart).

## Theory:

**Activity diagram** is another important diagram in UML to describe the dynamic aspects of the system.

Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system.

The control flow is drawn from one operation to another. This flow can be sequential, branched, or concurrent. Activity diagrams deal with all type of flow control by using different elements such as fork, join, etc

Activity is a particular operation of the system. Activity diagrams are not only used for visualizing the dynamic nature of a system, but they are also used to construct the executable system by using forward and reverse engineering techniques. The only missing thing in the activity diagram is the message part.

It does not show any message flow from one activity to another. Activity diagram is sometimes considered as the flowchart. Although the diagrams look like a flowchart, they are not. It shows different flows such as parallel, branched, concurrent, and single.

The purpose of an activity diagram can be described as –

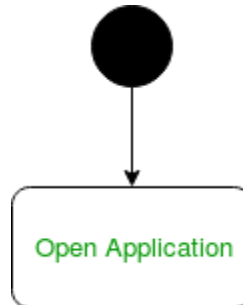
- Draw the activity flow of a system.
- Describe the sequence from one activity to another.
- Describe the parallel, branched and concurrent flow of the system.

## Activity Diagram Notations –

1. **Initial State** – The starting state before an activity takes place is depicted using the initial state.

● **Figure** – notation for initial state or start state A process can have only one initial state unless we are depicting nested activities. We use a

black filled circle to depict the initial state of a system. For objects, this is the state when they are instantiated. The Initial State from the UML Activity Diagram marks the entry point and the initial Activity State. For example – Here the initial state is the state of the system before the application is opened.

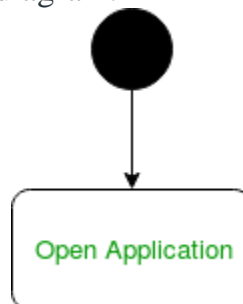


**Figure** – initial state symbol being used

2. **Action or Activity State** – An activity represents execution of an action on objects or by objects. We represent an activity using a rectangle with rounded corners. Basically any action or event that takes place is represented using an activity.



**Figure** – notation for an activity state For example – Consider the previous example of opening an application opening the application is an activity state in the activity diagram.

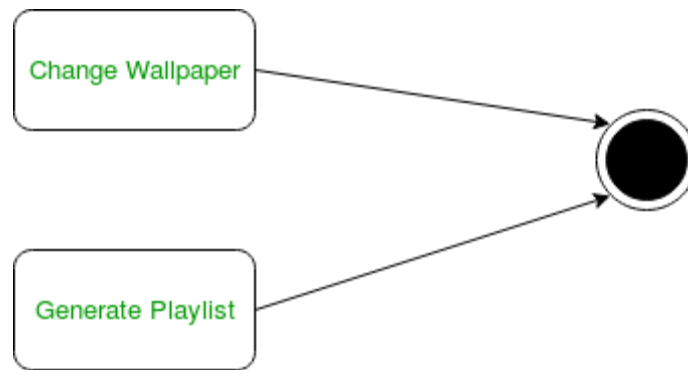


**Figure** – activity state symbol being used

3. **Action Flow or Control flows** – Action flows or Control flows are also referred to as paths and edges. They are used to show the transition from one activity state to another.



**Figure** – notation for control Flow An activity state can have multiple incoming and outgoing action flows. We use a line with an arrow head to depict a Control Flow. If there is a constraint to be adhered to while making the transition it is mentioned on the arrow. Consider the example – Here both the states transit into one final state using action flow symbols i.e. arrows.

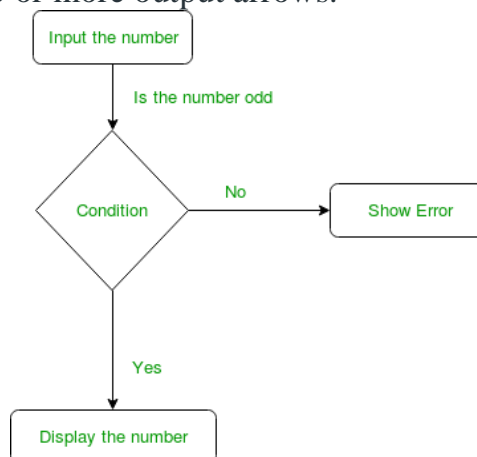


**Figure** – using action flows for transitions

4. **Decision node and Branching** – When we need to make a decision before deciding the flow of control, we use the decision node.

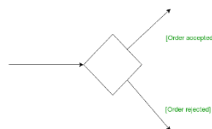


**Figure** – notation for decision node The outgoing arrows from the decision node can be labelled with conditions or guard expressions. It always includes two or more output arrows.



**Figure** – an activity diagram using decision node

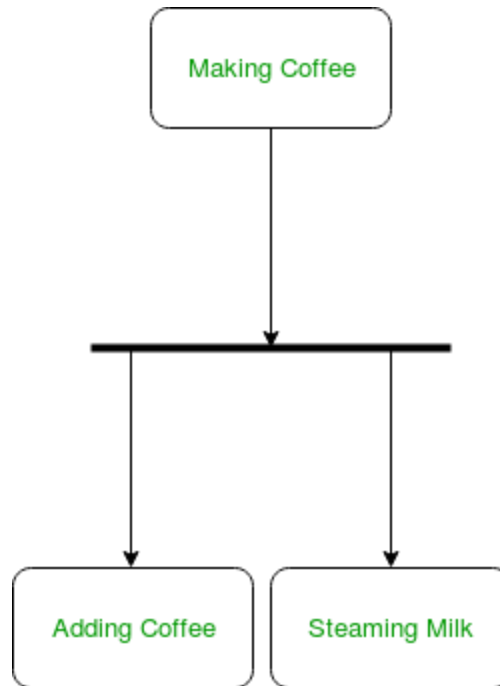
5. **Guards** – A Guard refers to a statement written next to a decision node on an arrow sometimes within square brackets.



**Figure** – guards being used next to a decision node The statement must be true for the control to shift along a particular direction. Guards help us know the constraints and conditions which determine the flow of a process.

6. **Fork** – Fork nodes are used to support concurrent activities.

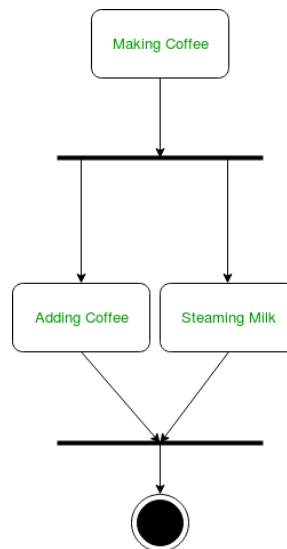
**Figure – fork notation** When we use a fork node when both the activities get executed concurrently i.e. no decision is made before splitting the activity into two parts. Both parts need to be executed in case of a fork statement. We use a rounded solid rectangular bar to represent a Fork notation with incoming arrow from the parent activity state and outgoing arrows towards the newly created activities. For example: In the example below, the activity of making coffee can be split into two concurrent activities and hence we use the fork notation.



**Figure – a diagram using fork**

7. **Join** – Join nodes are used to support concurrent activities converging into one. For join notations we have two or more incoming edges and one outgoing edge.

**Figure – join notation for example** – When both activities i.e. steaming the milk and adding coffee get completed, we converge them into one final activity.

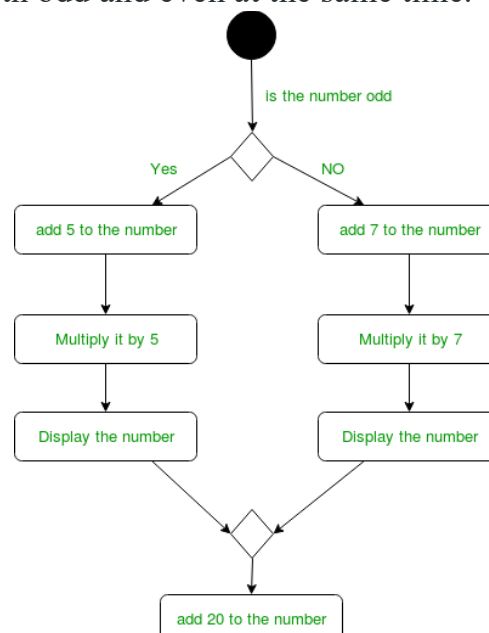


**Figure** – a diagram using join notation

8. **Merge or Merge Event** – Scenarios arise when activities which are not being executed concurrently have to be merged. We use the merge notation for such scenarios. We can merge two or more activities into one if the control proceeds onto the next activity irrespective of the path chosen.



**Figure** – merge notation For example – In the diagram below: we can't have both sides executing concurrently, but they finally merge into one. A number can't be both odd and even at the same time.



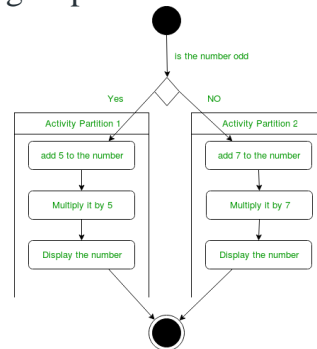
**Figure** – an activity diagram using merge notation

9. **Swimlanes** – We use swimlanes for grouping related activities in one column. Swimlanes group related activities into one column or one row. Swimlanes can be vertical and horizontal. Swimlanes are used to add


modularity to the activity diagram. It is not mandatory to use swimlanes. They usually give more clarity to the activity diagram. It's similar to creating a function in a program. It's not mandatory to do so, but, it is a recommended practice.

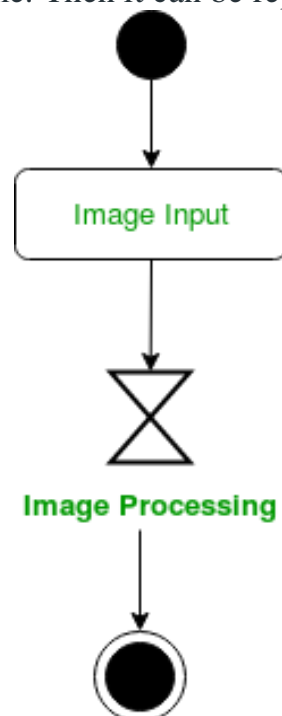


**Figure** – swimlanes notation We use a rectangular column to represent a swimlane as shown in the figure above. For example – Here different set of activities are executed based on if the number is odd or even. These activities are grouped into a swimlane.




**Figure** – an activity diagram making use of swimlanes

10. **Time Event** –  **Figure** – time event notation We can have a scenario where an event takes some time to complete. We use an hourglass to represent a time event. For example – Let us assume that the processing of an image takes a lot of time. Then it can be represented as shown below.



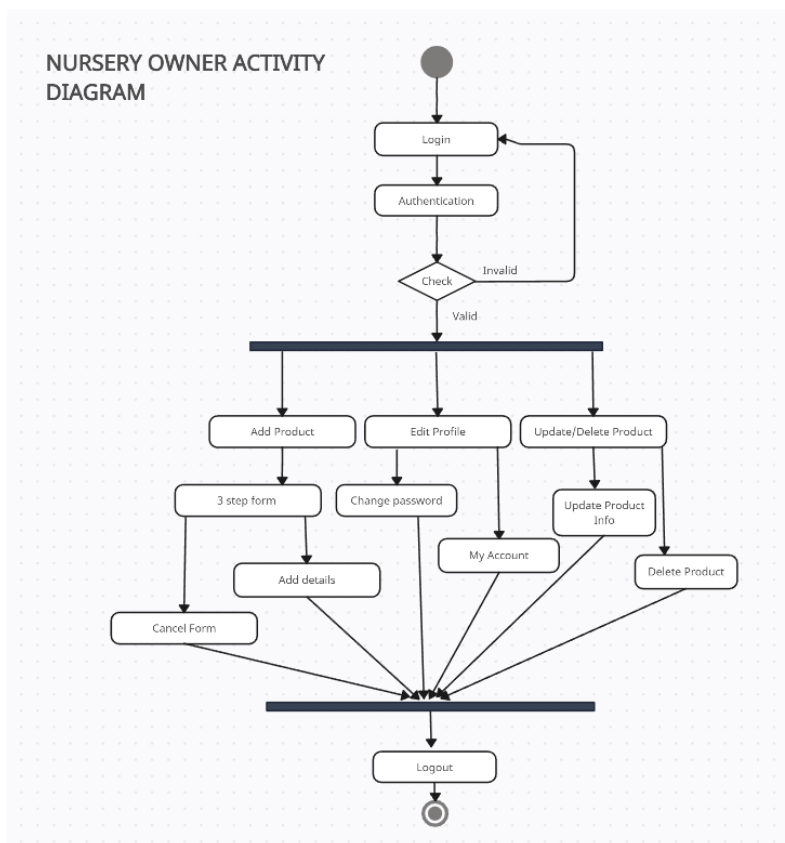
**Figure** – an activity diagram using time event

**11. Final State or End State** – The state which the system reaches when a particular process or activity ends is known as a Final State or End State. We use a filled circle within a circle notation to represent the final state in a state machine diagram. A system or a process can have multiple final

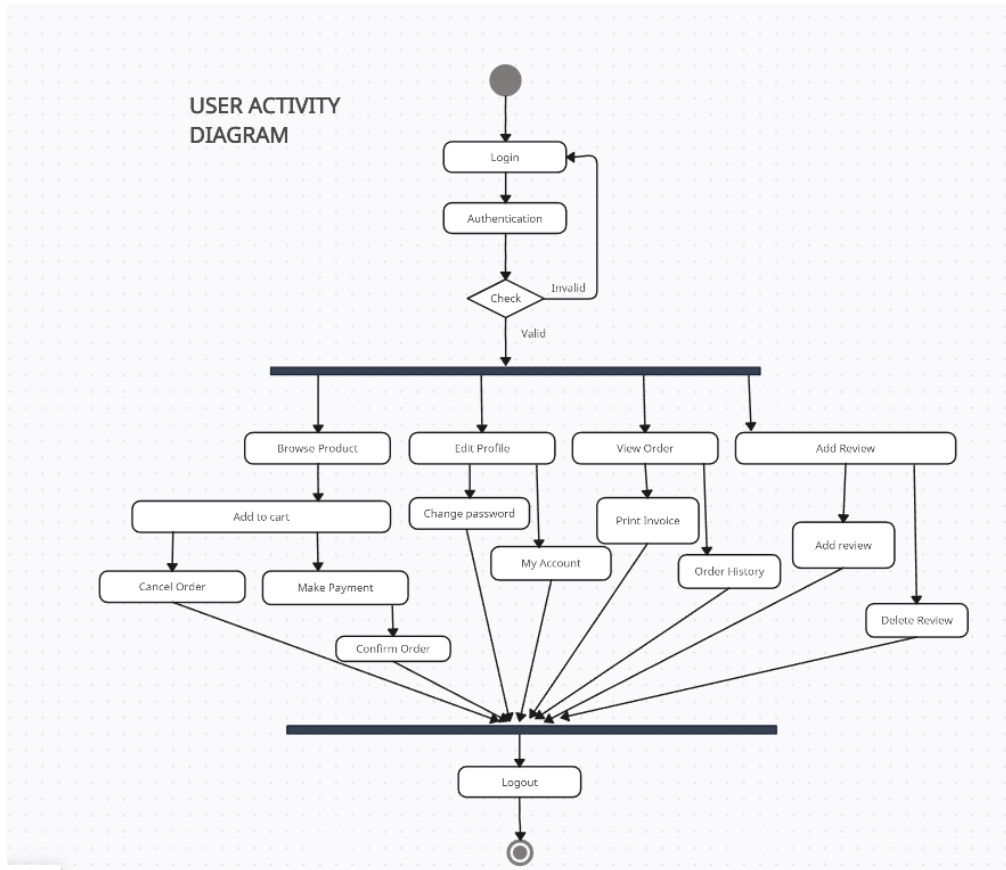
states.  **Figure** – notation for final state

### Uses of an Activity Diagram –

- Dynamic modelling of the system or a process.
- Illustrate the various steps involved in a UML use case.
- Model software elements like methods, operations and functions.
- We can use Activity diagrams to depict concurrent activities easily.
- Show the constraints, conditions and logic behind algorithms.







## State Diagram:

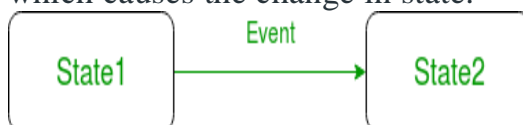
### Basic components of a statechart diagram –

1. **Initial state** – We use a black filled circle represent the initial state of a System or a class.



**Figure** – initial state notation

2. **Transition** – We use a solid arrow to represent the transition or change of control from one state to another. The arrow is labelled with the event which causes the change in state.



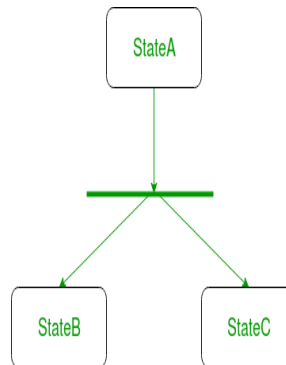
**Figure** – transition

3. **State** – We use a rounded rectangle to represent a state. A state represents the conditions or circumstances of an object of a class at an instant of time.



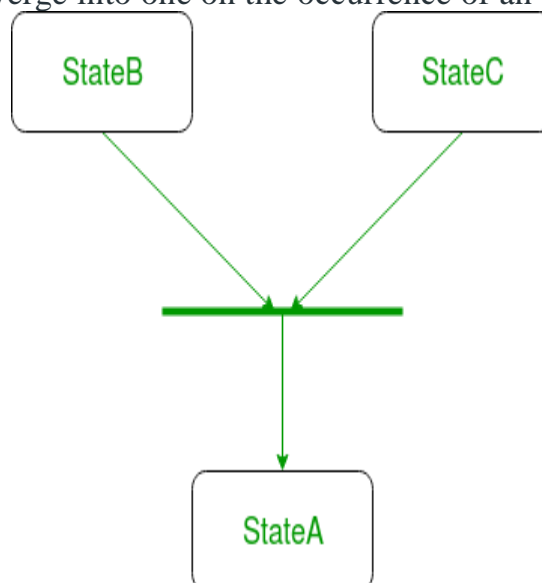
**Figure** – state notation

4. **Fork** – We use a rounded solid rectangular bar to represent a Fork notation with incoming arrow from the parent state and outgoing arrows towards the newly created states. We use the fork notation to represent a state splitting into two or more concurrent states.



**Figure** – a diagram using the fork notation

5. **Join** – We use a rounded solid rectangular bar to represent a Join notation with incoming arrows from the joining states and outgoing arrow towards the common goal state. We use the join notation when two or more states concurrently converge into one on the occurrence of an event or events.

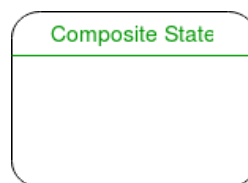


**Figure** – a diagram using join notation

6. **Self transition** – We use a solid arrow pointing back to the state itself to represent a self transition. There might be scenarios when the state of the object does not change upon the occurrence of an event. We use self transitions to represent such cases.

**Figure** – self transition notation

7. **Composite state** – We use a rounded rectangle to represent a composite state also. We represent a state with internal activities using a composite state.

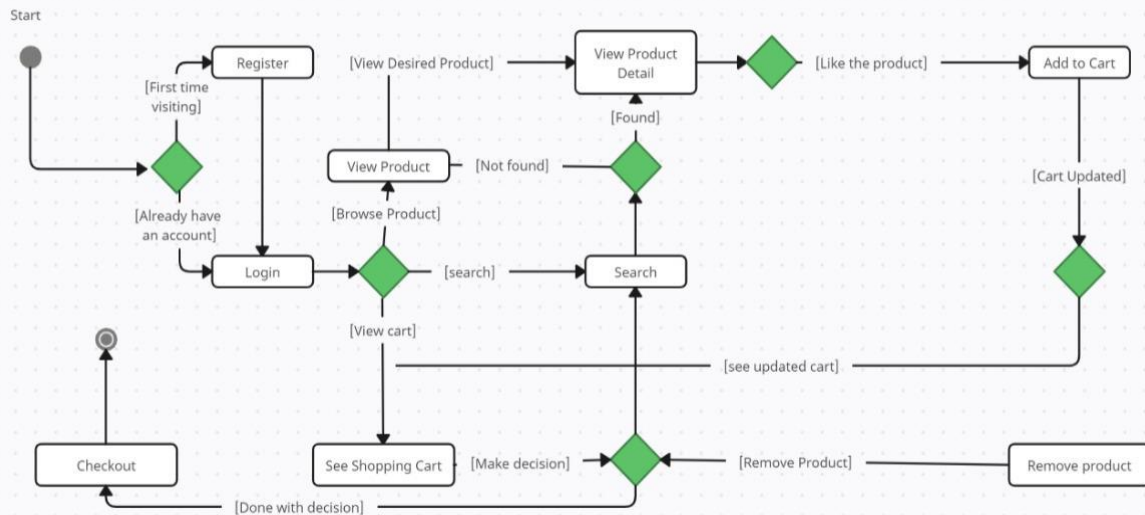
**Figure** – a state with internal activities

8. **Final state** – We use a filled circle within a circle notation to represent the final state in a state machine diagram.

**Figure** – final state notation

#### Uses of statechart diagram –

- We use it to state the events responsible for change in state (we do not show what processes cause those events).
- We use it to model the dynamic behaviour of the system .
- To understand the reaction of objects/classes to internal or external stimuli.



**USE STATE DIAGRAM FOR NURSERY OWNER**

