# Object-Oriented Analysis and Design using JAVA

B.Tech (CSE/IT) 5th SEM
2020-2021

## Lecture-25 State Machine diagram

# Introduction

- The state machine view describes the dynamic behavior of objects over time by modeling the lifecycles of objects of each class.
- Each object is treated as an isolated entity that communicates with the rest of the world by detecting events and responding to them.
- Events represent the kinds of changes that an object can detect—the receipt of calls or explicit signals from one object to another, a change in certain values, or the passage of time.
- Anything that can affect an object can be characterized as an event. Real-world happenings are modeled as signals from the outside world to the system.

**Reference:** Unified Modeling Language (UML): Complete Guide & Examples -By James Rumbaugh, Ivar Jacobson, Grady Booch

# State and action

- A state is a set of object values for a given class that have the same qualitative response to events that occur.
- In other words, all objects with the same state react in the same general way to an event, so all objects in a given state execute the same action when they receive the same event.
- Objects in different states, however, may react differently to the same event, by performing different actions.
- For example, an automatic teller machine reacts to the cancel button one way when it is processing a transaction and another way when it is idle

**Reference:** Unified Modeling Language (UML): Complete Guide & Examples -By James Rumbaugh, Ivar Jacobson, Grady Booch

# State Machine

- A state machine is a graph of states and transitions. Usually a state machine is attached to a class and describes the response of an instance of the class to events that it receives.

- State machines may also be attached to operations, use cases, and collaborations to describe their execution.

- A state machine is a model of all possible life histories of an object of a class. The object is examined in isolation. Any external influence from the rest of the world is summarized as an event.

- When the object detects an event, it responds in a way that depends on its current state. The response may include the execution of an action and a change to a new state. State machines can be structured to inherit transitions, and they can model concurrency.

**Reference:** Unified Modeling Language (UML): Complete Guide & Examples -By James Rumbaugh, Ivar Jacobson, Grady Booch

# Event

- An event is a noteworthy occurrence that has a location in time and space. It occurs at a point in time; it does not have duration.

- Model something as an event if its occurrence has consequences. When we use the word event by itself, we usually mean an event descriptor—that is, a description of all the individual event occurrences that have the same general form, just as the word class means all the individual objects that have the same structure.

- A specific occurrence of an event is called an event instance. Events may have parameters that characterize each individual event instance, just as classes have attributes that characterize each object. As with classes, signals can be arranged in generalization hierarchies to share com-mon structure.

**Reference:** Unified Modeling Language (UML): Complete Guide & Examples -By James Rumbaugh, Ivar Jacobson, Grady Booch

# Types of Event

| *Event Type* | *Description* | *Syntax* |
|---|---|---|
| call event | Receipt of an explicit synchronous request among objects that waits for a response | op (a:T) |
| change event | A change in value of a Boolean expression | **when** (exp) |
| signal event | Receipt of an explicit, named, asynchronous communication among objects | sname (a:T) |
| time event | The arrival of an absolute time or the passage of a relative amount of time | **after** (time) |

**Reference:** Unified Modeling Language (UML): Complete Guide & Examples -By James Rumbaugh, Ivar Jacobson, Grady Booch

# Signal Event

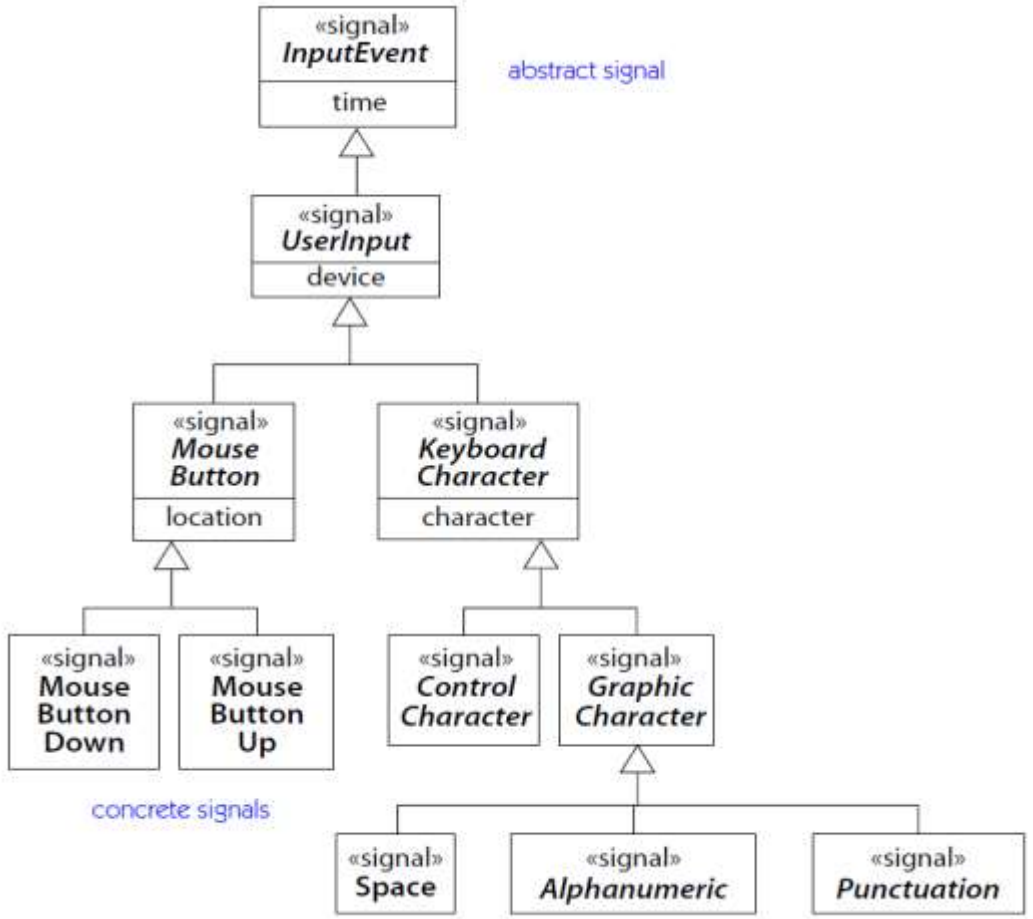
- Signal event. A signal is a named entity that is explicitly intended as a communication vehicle between two objects; the reception of a signal is an event for the receiving object.

- The sending object explicitly creates and initializes a signal instance and sends it to one or a set of explicit objects. Signals embody asynchronous one-way communication, the most fundamental kind.

- The sender does not wait for the receiver to deal with the signal but continues with its own work independently. To model two-way communication, multiple signals can be used, at least one in each direction. The sender and the receiver can be the same object.

**Reference:** Unified Modeling Language (UML): Complete Guide & Examples -By James Rumbaugh, Ivar Jacobson, Grady Booch

- Signals may be declared in class diagrams as classifiers, using the keyword «signal»; the parameters of the signal are declared as attributes. As classifiers, signals can have generalization relationships. Signals may be children of other signals; they inherit the parameters of their parents, and they trigger transitions that depend on the parent signal




**Reference:** Unified Modeling Language (UML): Complete Guide & Examples -By James Rumbaugh, Ivar Jacobson, Grady Booch

# Call Event

- Call event. A call event is the reception of a call by an object that chooses to implement an operation as a state machine transition rather than as a fixed procedure.

- To the caller, an ordinary call (implemented by a method) is indistinguishable from a call event. The receiver chooses whether an operation will be implemented as a method or a call event trigger in a state machine.

- The parameters of the operation are the parameters of the event. Once the receiving object processes the call event by taking a transition triggered by the event or failing to take any transition, control returns to the calling object. Unlike an ordinary call, however, the receiver of a call event may continue its own execution in parallel with the caller

**Reference:** Unified Modeling Language (UML): Complete Guide & Examples -By James Rumbaugh, Ivar Jacobson, Grady Booch

# Change Event

- Change event. A change event is the satisfaction of a Boolean expression that depends on certain attribute values.

- This is a declarative way to wait until a condition is satisfied, but it must be used with care, because it represents a continuous and potentially nonlocal computation (action at a distance, because the value or values tested may be distant).

- This is both good and bad. It is good because it focuses the model on the true dependency—an effect that occurs when a given condition is satisfied—rather than on the mechanics of testing the condition.

**Reference:** Unified Modeling Language (UML): Complete Guide & Examples -By James Rumbaugh, Ivar Jacobson, Grady Booch

# Time Event

- Time event. Time events represent the passage of time.
- A time event can be specified either in absolute mode (time of day) or relative mode (time elapsed since a given event).
- In a high-level model, time events can be thought of as events from the universe; in an implementation model, they are caused by signals from some specific object, either the operating system or an object in the application

**Reference:** Unified Modeling Language (UML): Complete Guide & Examples -By James Rumbaugh, Ivar Jacobson, Grady Booch

# State

- A state describes a period of time during the life of an object of a class.

- It can be characterized in three complementary ways: as a set of object values that are qualitatively similar in some respect; as a period of time during which an object waits for some event or events to occur; or as a period of time during which an object performs some ongoing activity.

- A state may have a name, although often it is anonymous and is described simply by its actions. In a state machine, a set of states is connected by transitions. Although transitions connect two states (or more, if there is a fork or join of control), transitions are processed by the state that they leave. When an object is in a state, it is sensitive to the trigger events on transitions leaving the state.

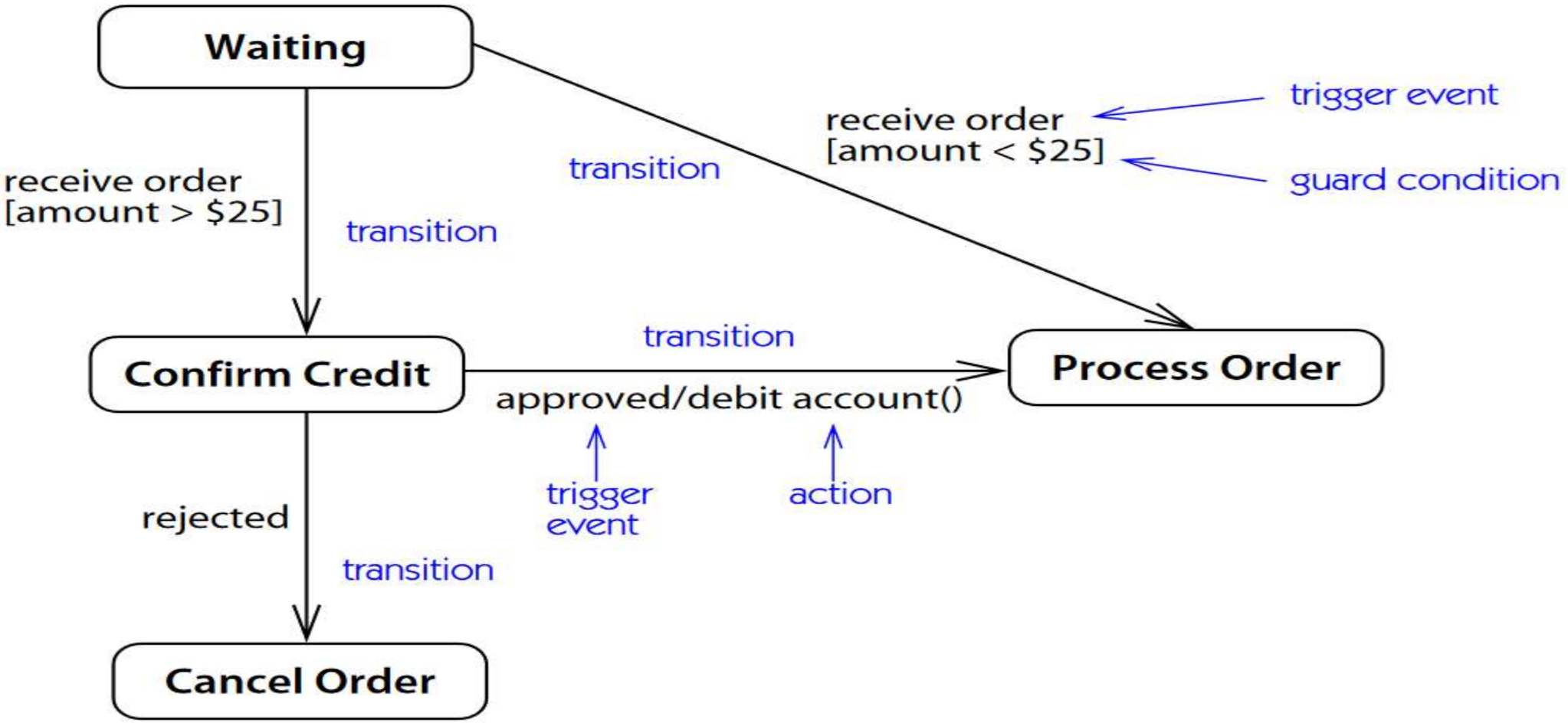
**Reference:** Unified Modeling Language (UML): Complete Guide & Examples -By James Rumbaugh, Ivar Jacobson, Grady Booch

# Transition

- A transition leaving a state defines the response of an object in the state to the occurrence of an event. In general, a transition has an event trigger, a guard condition, an action, and a target state

| *Transition Kind* | *Description* | *Syntax* |
|---|---|---|
| entry action | An action that is executed when a state is entered | entry/ action |
| exit action | An action that is executed when a state is exited | exit/ action |
| external transition | A response to an event that causes a change of state or a self-transition, together with a specified action. It may also cause the execution of exit and/or entry actions for states that are exited or entered. | e(a:T)[exp]/action |
| internal transition | A response to an event that causes the execution of an action but does not cause a change of state or execution of exit or entry actions | e(a:T)[exp]/action |

**Reference:** Unified Modeling Language (UML): Complete Guide & Examples -By James Rumbaugh, Ivar Jacobson, Grady Booch




**Reference:** Unified Modeling Language (UML): Complete Guide & Examples -By James Rumbaugh, Ivar Jacobson, Grady Booch

# Key references

Unified Modeling Language (UML): Complete Guide & Examples
-By James Rumbaugh, Ivar Jacobson, Grady Booch

Thank You