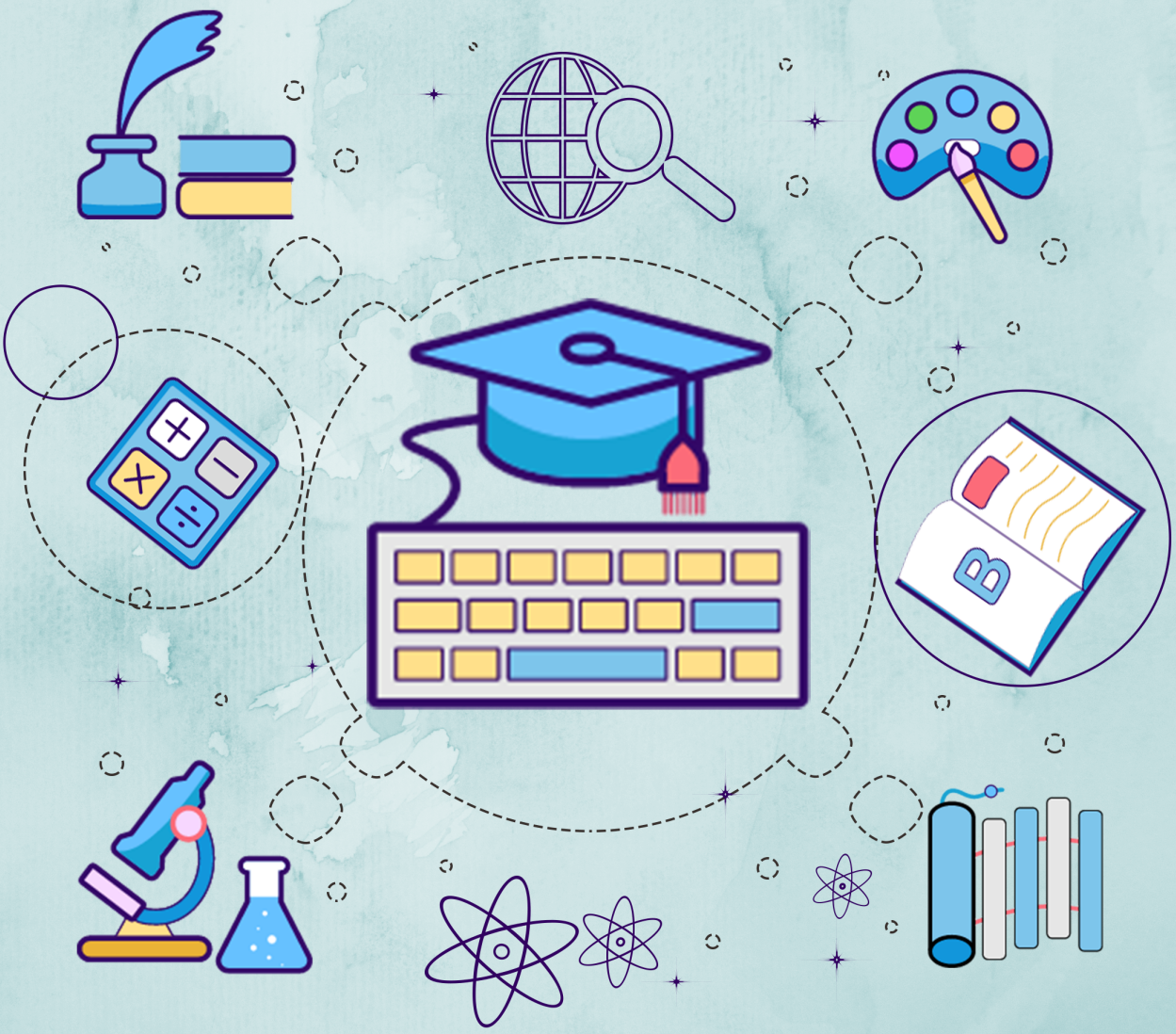
APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY

# KeralaNotes



SYLLABUS | STUDY MATERIALS | TEXTBOOK

PDF | SOLVED QUESTION PAPERS

www.keralanotes.com

KeralaNotes

# KTU STUDY MATERIALS

## OBJECT ORIENTED PROGRAMMING USING JAVA

## CST 205

# Module 1

Related Link :

- KTU S3 STUDY MATERIALS
- KTU S3 NOTES
- KTU S3 SYLLABUS
- KTU S3 TEXTBOOK PDF
- KTU S3 PREVIOUS YEAR SOLVED QUESTION PAPER

# MODULE 1- INTRODUCTION

## CHAPTER – 1
## Approaches to Software Design





# Software & Software Design

### ❖Software

- Software is a collection of instructions that enable the user to interact with a computer , its hardware or perform tasks
- Without software, most computers would be useless. For example, without your Internet browser software, you could not surf the Internet. Without an operating system, the browser could not run on your computer.

## There are two types of software

1. System Software
2. Application Software

- Examples of system software are Operating System, Compilers, Interpreter, Assemblers, etc.
- Examples of Application software are Railways Reservation Software, Microsoft Office Suite Software, Microsoft Word, Microsoft PowerPoint , etc.


System Software's


Application software's



## ❖Software Design

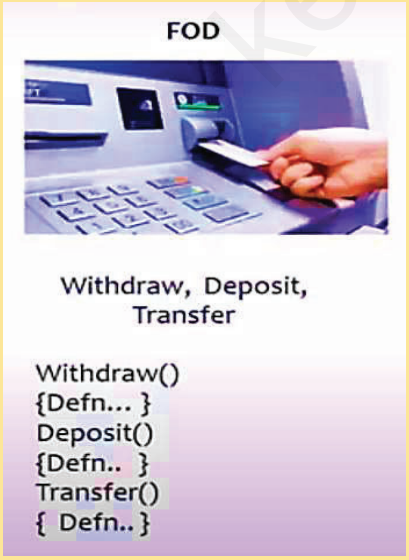
- Software design is a process to transform user requirements into some suitable form, which helps the programmer in software coding and implementation.
- The design process for software systems often has two levels. At the first level the focus is on deciding which modules are needed for the system on the basis of SRS (Software Requirement Specification) and how the modules should be interconnected.
- Software design is the first step in SDLC (Software Design Life Cycle)
- It tries to specify how to fulfill the requirements mentioned in SRS document.

# Functional Oriented Design (FOD)

- In function-oriented design, the system is comprised of many smaller sub-systems known as functions.
- These functions are capable of performing significant task in the system
- Function oriented design inherits some properties of structured design where divide and conquer methodology is used.
- This design mechanism divides the whole system into smaller functions



- These functional modules can share information among themselves by means of information passing and using information available globally.



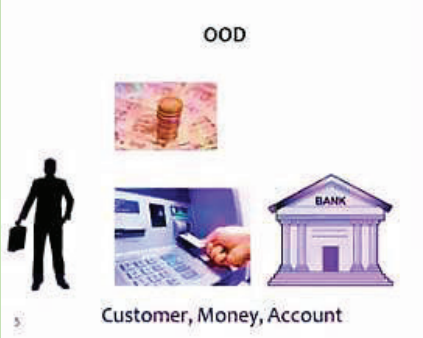

Eg: Banking process

Here withdraw, Deposit, Transfer are functions and that can be divided in to sub-functions again.

So, in FOD, the entire problem is divided in to number of functions and those functions are broken down in to smaller functions and these smaller functions are converted in to software modules.

# Object Oriented Design (OOD)

- OOD is based on Objects and interaction between the objects
- Interaction between objects is called message communication.
- It involves the designing of Objects, Classes and the relationship between the classes



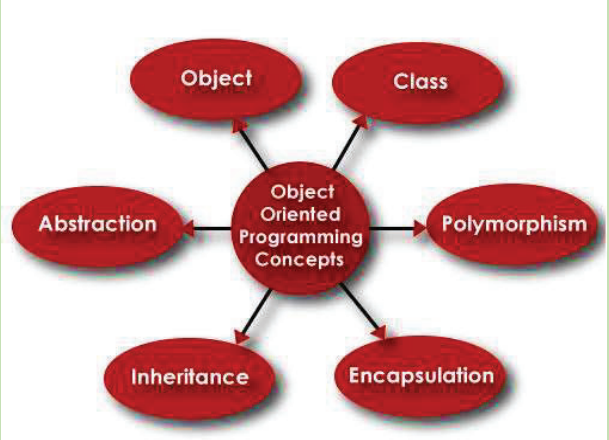

Consider the previous example of Banking process.

Here, customer, money and account are objects



- In OOD, implementation of a software based on the concepts of objects.
- This approach is very close to the real-world applications

## Basic Object Oriented concepts



## ❖OBJECT

- Objects are real-world entities that has their own properties and behavior.
- It has physical existence

  Eg: person, banks, company, customers etc

## ❖CLASS

- A class is a blueprint or prototype from which objects are created
- A class is a generalized description of an object.
- An object is an instance of a class



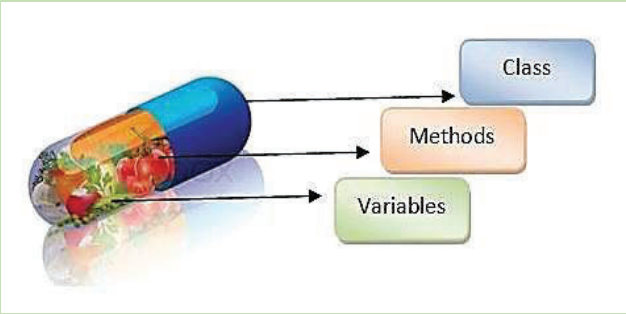


## ❖Relationship between Object & Class

- Let's take Human Being as a class. My name is John, and I am an instance/object of the class Human Being
- Object has a physical existence while a class is just a logical definition.

## ❖Encapsulation

- The wrapping up of data(variables) and function (methods) into a single unit (called class) is known as *encapsulation.*
- It is also called "**information hiding**"




## Key Points of Encapsulation

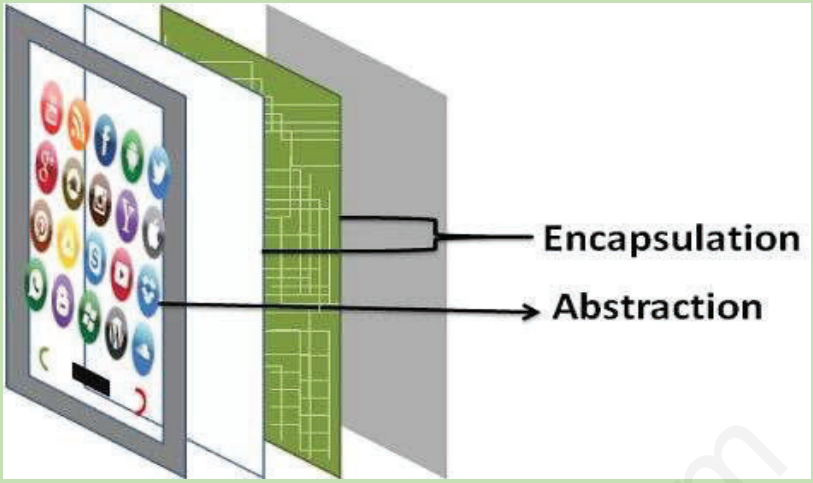
- Protection of data from accidental corruption
- Flexibility and extensibility of the code and reduction in complexity
- Encapsulation of a class can hide the internal details of how an object does something
- Encapsulation protects abstraction



## ❖ABSTRACTION

- Abstraction means displaying only essential information and hiding the details.
- Data abstraction refers to providing only essential information about the data to the outside world, hiding the background details or implementation.
- Consider a real-life example of a man driving a car. The man only knows that pressing the accelerators will increase the speed of the car or applying brakes will stop the car but he does not know about how on pressing accelerator the speed is actually increasing, he does not know about the inner mechanism of the car or the implementation of accelerator, brakes etc in the car. This is what abstraction is.

## Abstraction & Encapsulation






## ❖POLYMORPHISM

- The word polymorphism means having many forms
- In simple words, we can define polymorphism as the ability of a message to be displayed in more than one form.

Eg: A person at the same time can have different characteristic. Like a man at the same time is a father, a husband, an employee. So the same person posses different behavior in different situations. This is called polymorphism.

- An operation may exhibit different behaviors in different instances. The behavior depends upon the types of data used in the operation.


Fig: polymorphism



## ❖Inheritance

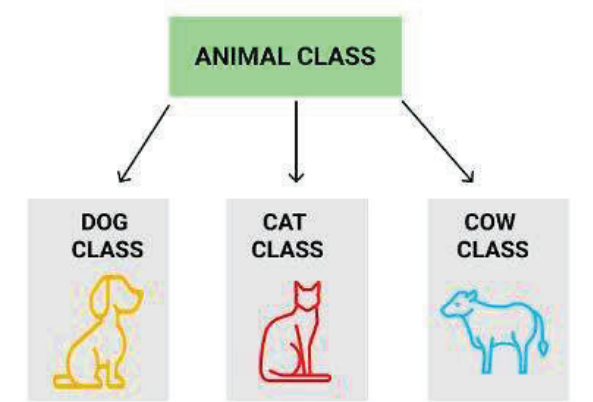
➢The capability of a class to derive properties and characteristics from another class is called Inheritance.

OR

*Inheritance* is the process by which objects of one class acquired the properties of objects of another classes

- **Sub Class** : The class that inherits properties from another class is called Sub class or Derived Class.
- **Super Class** : The class whose properties are inherited by sub class is called Base Class or Super class.

- **Reusability**: Inheritance supports the concept of "reusability", i.e. when we want to create a new class and there is already a class that includes some of the code that we want, we can derive our new class from the existing class. By doing this, we are reusing the fields and methods of the existing class.




**Eg**: Dog, Cat, Cow can be Derived Class of Animal Base Class.





# Unified Modeling Language (UML)

- UML (Unified Modeling Language) is a general-purpose, graphical modeling language in the field of Software Engineering
- UML is used to specify, visualize, construct, and document the artifacts (major elements) of the software system
- UML is a visual language for developing software blue prints (designs). A blue print or design represents the model.
- For example, while constructing buildings, a designer or architect develops the building blueprints. Similarly, we can also develop blue prints for a software system.

- UML is the most commonly and frequently used language for building software system blueprints
- UML is **not a programming language**, it is rather a visual language.

**The UML has the following features:**

- It is a generalized modeling language.
- It is distinct from other programming languages like C++, Python, etc.
- It is interrelated to object-oriented analysis and design.
- It is used to visualize the workflow of the system.
- It is a pictorial language, used to generate powerful modeling artifacts

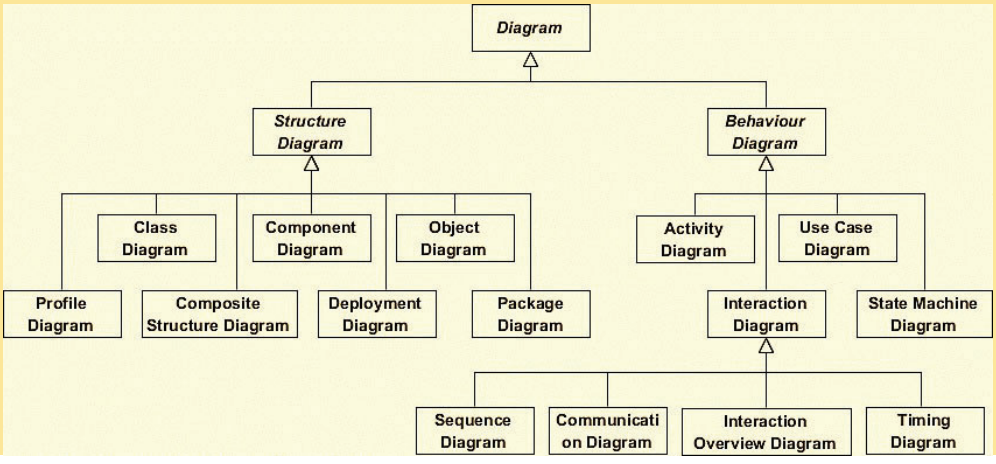


- UML is linked with object oriented design and analysis

**Diagrams in UML can be broadly classified as:**

**Structure Diagrams :** Capture static aspects or structure of a system

**Behavior Diagrams:** Capture dynamic aspects or behavior of the system



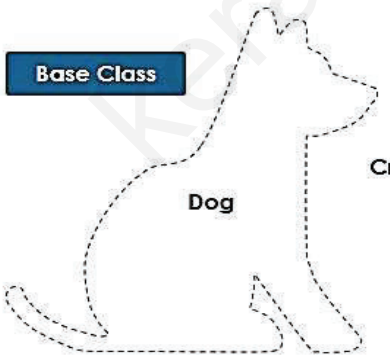
## ❖CLASS DIAGRAM

- The most widely use UML diagram is the class diagram. It is the building block of all object oriented software systems.
- Using class diagrams we can create the static structure of a system by showing system's classes, their methods and attributes.
- Class diagrams also help us identify relationship between different classes or objects.
- There are several software available which can be used online and offline to draw these diagrams Like Edraw max, lucid chart etc.

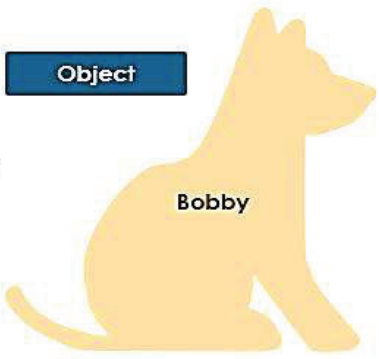




## Class & Object



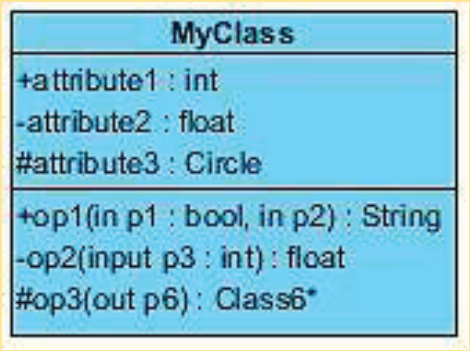

Create instance




| Properties | Methods |
|---|---|
| Color | Sit |
| Eye Color | Lay Down |
| Height | Shake |
| Length | Come |
| Weight | |

| Property Values | Methods |
|---|---|
| Color: Yellow | Sit |
| Eye Color: Brown | Lay Down |
| Height: 17 in | Shake |
| Length: 35 in | Come |
| Weight: 24 pounds | |

## Class Notation




A class notation consists of three parts:

➢**Class Name:**

- The name of the class appears in the first partition.

➢**Class Attributes:**

- Attributes are shown in the second partition.
- The attribute type is shown after the colon.
- Attributes map onto member variables (data members) in code.

➢**Class Operations (Methods):**

- Operations are shown in the third partition. They are services the class provides.
- The return type of a method is shown after the colon at the end of the method signature.
- The return type of method parameters are shown after the colon following the parameter name. Operations map onto class methods in code

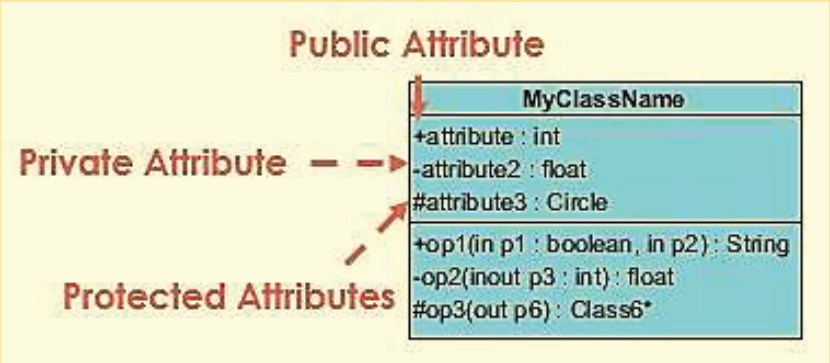


➢The +, - and # symbols before an attribute and operation name in a class denote the visibility of the attribute and operation.

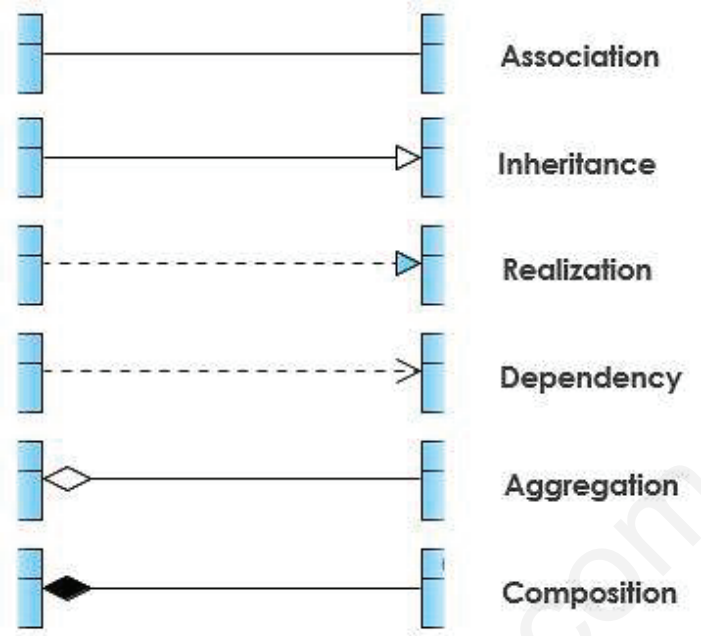
+ denotes public attributes or operations

- denotes private attributes or operations

# denotes protected attributes or operations



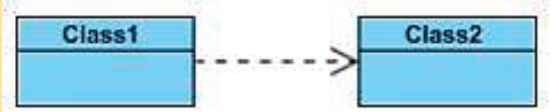
## Relationships between classes






### 1. Dependency

- A dependency means the relation between two or more classes in which a change in one may force changes in the other.
- Dependency indicates that one class depends on another.
- A dashed line with an open arrow




### 2. Inheritance (or Generalization)

- A generalization helps to connect a subclass to its superclass.
- A sub-class is inherited from its superclass.
- A solid line with a hollow arrowhead that point from the child to the parent class

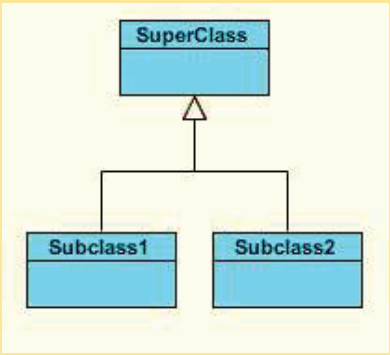



**Fig: Inheritance** (or Generalization)

## 3. Association

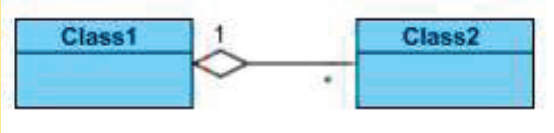
- This kind of relationship represents static relationships between classes A and B.
- There is an association between Class1 and Class2
- A solid line connecting two classes




Fig: Association

## 4. Aggregation

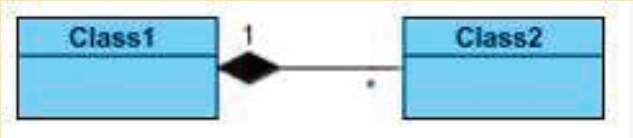
- A special type of association. It represents a "part of" relationship
- Class2 is part of Class1.




- Many instances (denoted by the *) of Class2 can be associated with Class1.
- A solid line with an unfilled diamond at the association end connected to the class of composite

## 5. Composition

- A special type of aggregation where parts are destroyed when the whole is destroyed.
- Objects of Class2 live and die with Class1.
- Class2 cannot stand by itself.
- A solid line with a filled diamond at the association connected to the class of composite

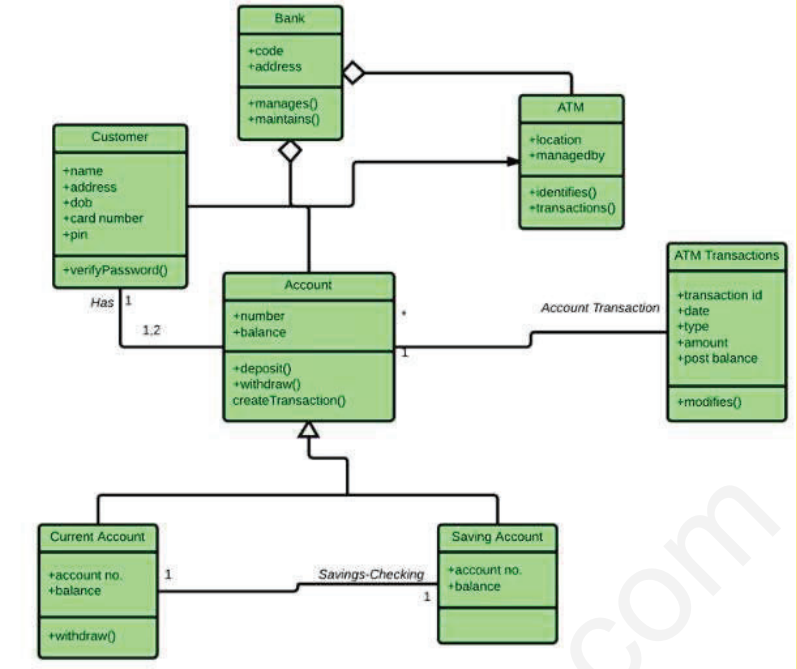





## Multiplicity

- It means, how many objects of each class take part in the relationships
- Exactly one - 1
- Zero or one - 0..1
- Many - 0..* or *
- One or more - 1..*
- Exact Number - e.g. 3..4 or 6
- Or a complex relationship - e.g. 0..1, 3..4, 6.* would mean any number of objects other than 2 or 5

## Eg: Class diagram for an ATM system






## ❖USE CASE MODEL / USE CASE DIAGRAM

- The purpose of a use case diagram in UML is to demonstrate the different ways that a user might interact with a system.
- It captures the dynamic behavior of a live system.
- a use case diagram can summarize the details of your system's users (also known as actors) and their interactions with the system.
- To build a use case diagram, we will use a set of specialized symbols and connectors
- A use case diagram doesn't go into a lot of detail, but it depicts a high-level overview of the relationship between use cases, actors, and systems.

➢A use-case model is a model of how different types of users interact with the system to solve a problem

## Use case diagram components

- **Actors:** The users that interact with a system. An actor can be a person, an organization, or an outside system that interacts with your application or system. They must be external objects that produce or consume data.
- **System:** A specific sequence of actions and interactions between actors and the system. A system may also be referred to as a scenario
- **Goals:** The end result of most use cases. A successful diagram should describe the activities and variants used to reach the goal.

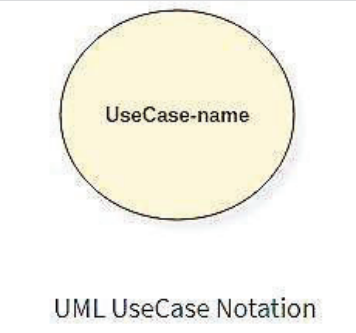




## Use case diagram symbols and notation

### 1. Use cases

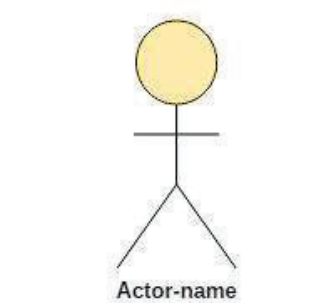
- Horizontally shaped ovals that represent the different uses that a user might have
- A use case represents a distinct functionality of a system, a component, a package, or a class




UML UseCase Notation

## 2. Actors

- Stick figures that represent the people actually employing the use cases.
- A user is the best example of an actor
- One actor can be associated with multiple use cases in the system




UML Actor Notation

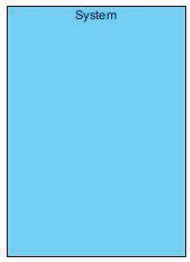


## 3. Associations

- A line between actors and use cases
- In complex diagrams, it is important to know which actors are associated with which use cases.

## 4. System boundary boxes

- A box that sets a system scope to use cases
- All use cases outside the box would be considered outside the scope of that system.



## 5. Packages

- A UML shape that allows you to put different elements into groups
- Just as with component diagrams, these groupings are represented as file folders.

**purposes of use case diagram**

✓Used to gather the requirements of a system.

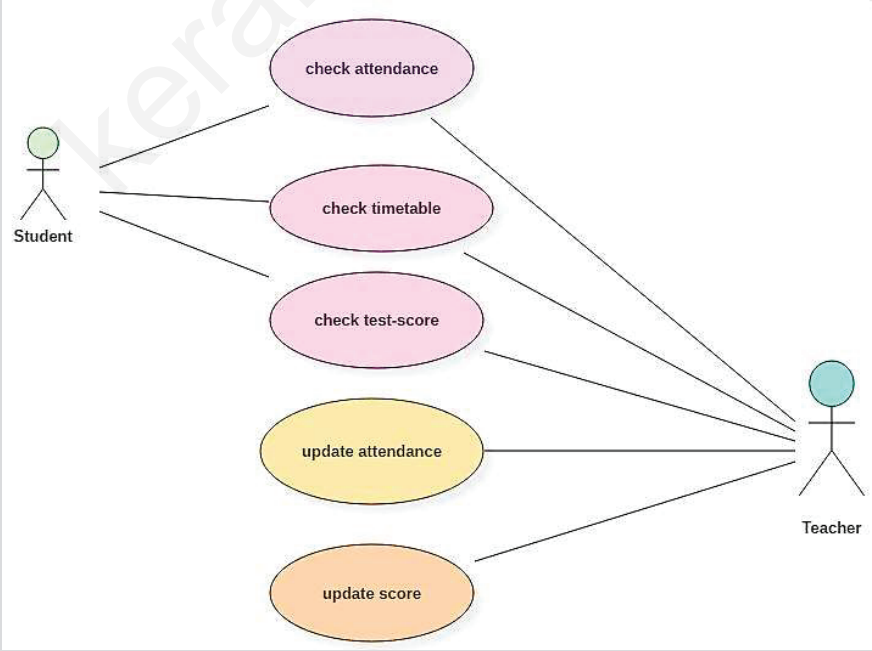
✓Used to get an outside view of a system.

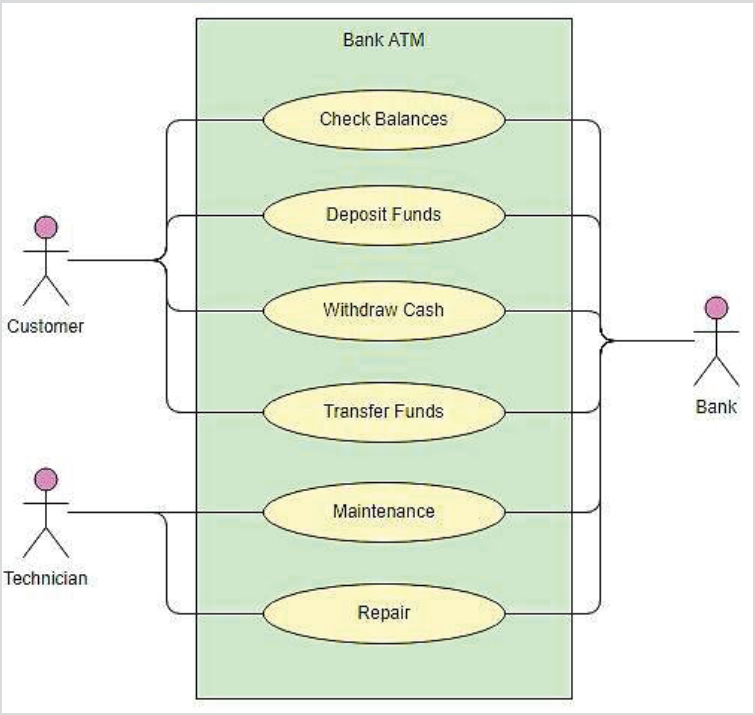
✓Identify the external and internal factors influencing the system.

✓Show the interaction among the requirements and actors



## Eg: Use case diagram of a student management system



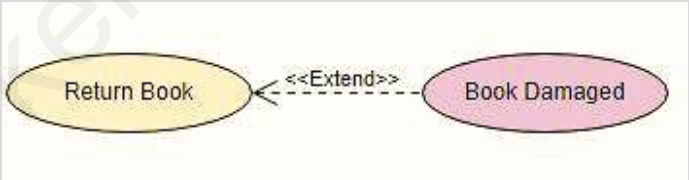



Eg: **ATM** use case diagram



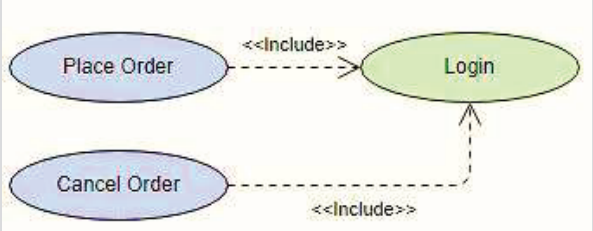
## <<extend>> Use Case

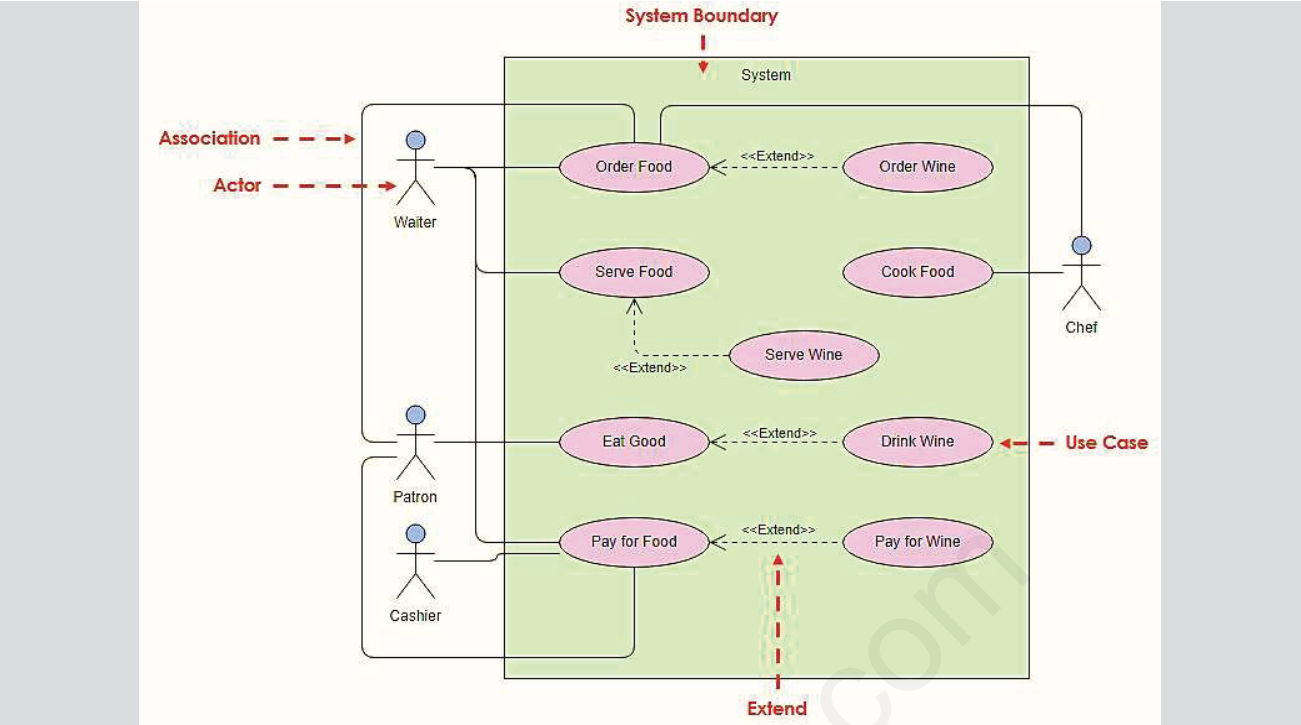
The <<extend>> use case inserting additional action sequences into the base use-case sequence.




## <<include>> Use Case

The time to use the <<include>> relationship is after you have completed the first cut description of all your main Use Cases.



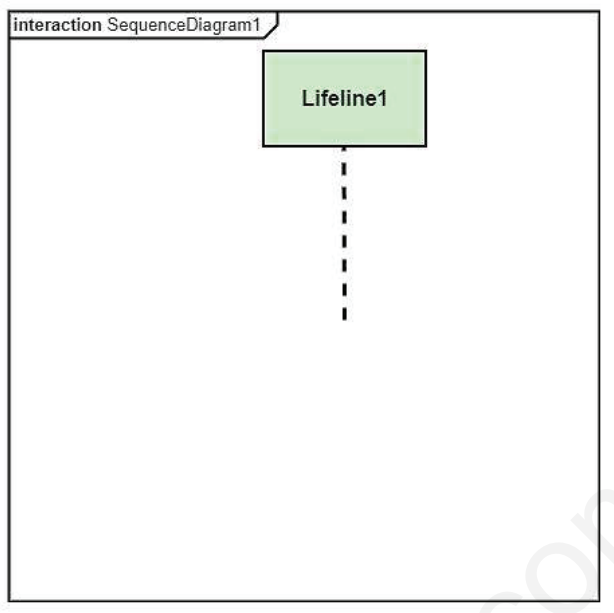





## ❖INTERACTION DIAGRAM

- **INTERACTION DIAGRAMS** are used in UML to establish communication between objects
- Interaction diagrams mostly focus on message passing and how these messages make up one functionality of a system
- The critical component in an interaction diagram is lifeline and messages.
- Interaction diagrams capture the dynamic behavior of any system
- The details of interaction can be shown using several notations such as sequence diagram, timing diagram, collaboration diagram.

## Notation of an Interaction Diagram






## Purpose of an Interaction Diagram

- To capture the dynamic behavior of a system.
- To describe the message flow in the system.
- To describe the structural organization of the objects.
- To describe the interaction among objects.
- Interaction diagram visualizes the communication and sequence of message passing in the system.
- Interaction diagram represents the ordered sequence of interactions within a system.
- Interaction diagrams can be used to explain the architecture of an object-oriented system.

## Different types of Interaction Diagrams

1. Sequence diagram

- Purpose - To visualize the sequence of a message flow in the system
- Shows the interaction between two lifelines

2. Collaboration diagram

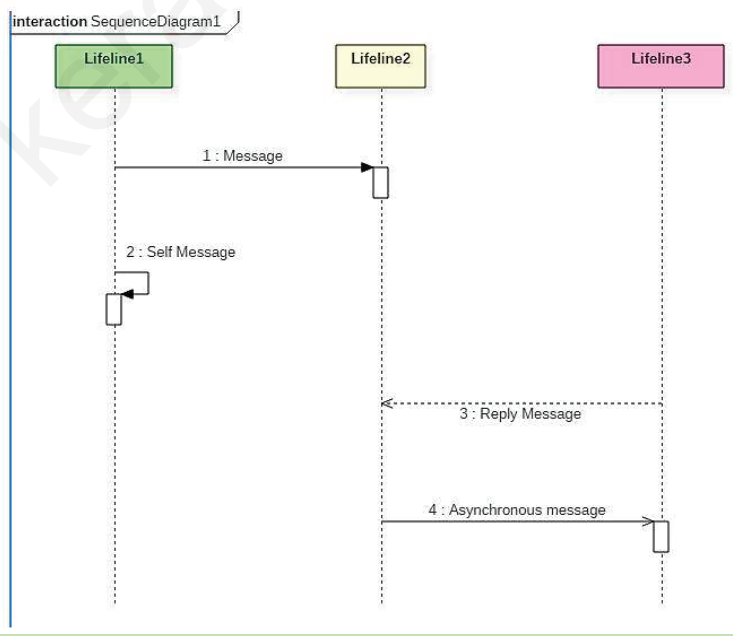
- Also called as a communication diagram
- Shows how various lifelines in the system connects.

**3.** Timing diagram

- Focus on the instance at which a message is sent from one object to another object.



## How to draw a Sequence Diagram



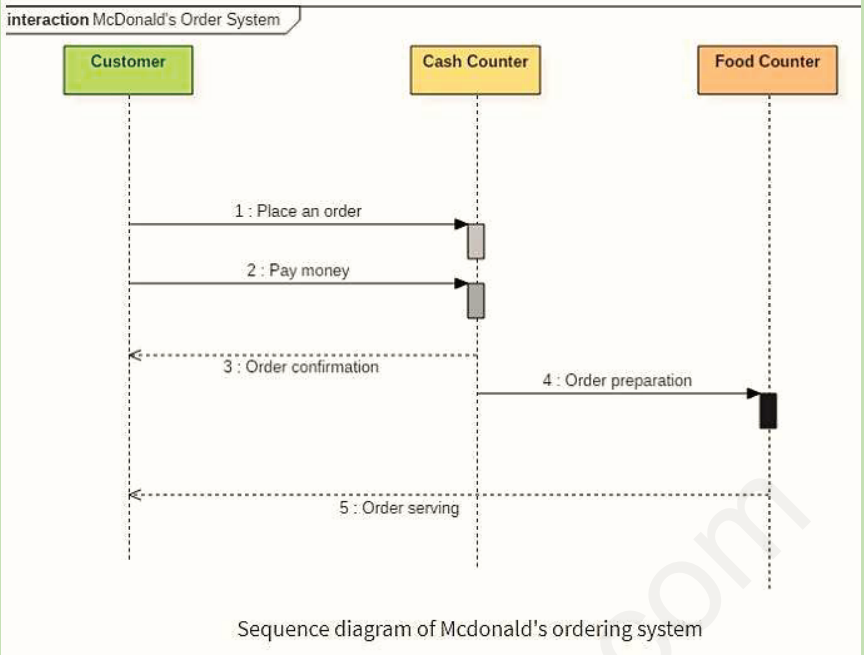
- In a sequence diagram, a lifeline is represented by a vertical bar.
- A lifeline represents an individual participant in a sequence diagram
- A lifeline will usually have a rectangle containing its object name
- A message flow between two or more objects is represented using a vertical dotted line which extends across the bottom of the page.
- In a sequence diagram, different types of messages and operators are used
- In a sequence diagram, iteration and branching are also used.



## Messages used

| Message Name | Meaning |
|---|---|
| Synchronous message | The sender of a message keeps waiting for the receiver to return control from the message execution. |
| Asynchronous message | The sender does not wait for a return from the receiver; instead, it continues the execution of a next message. |
| Return message | The receiver of an earlier message returns the focus of control to the sender. |
| Object creation | The sender creates an instance of a classifier. |
| Object destruction | The sender destroys the created instance. |
| Found message | The sender of the message is outside the scope of interaction. |
| Lost message | The message never reaches the destination, and it is lost in the interaction. |

## Sequence diagram example




Sequence diagram of Mcdonald's ordering system



## Benefits of a Sequence Diagram

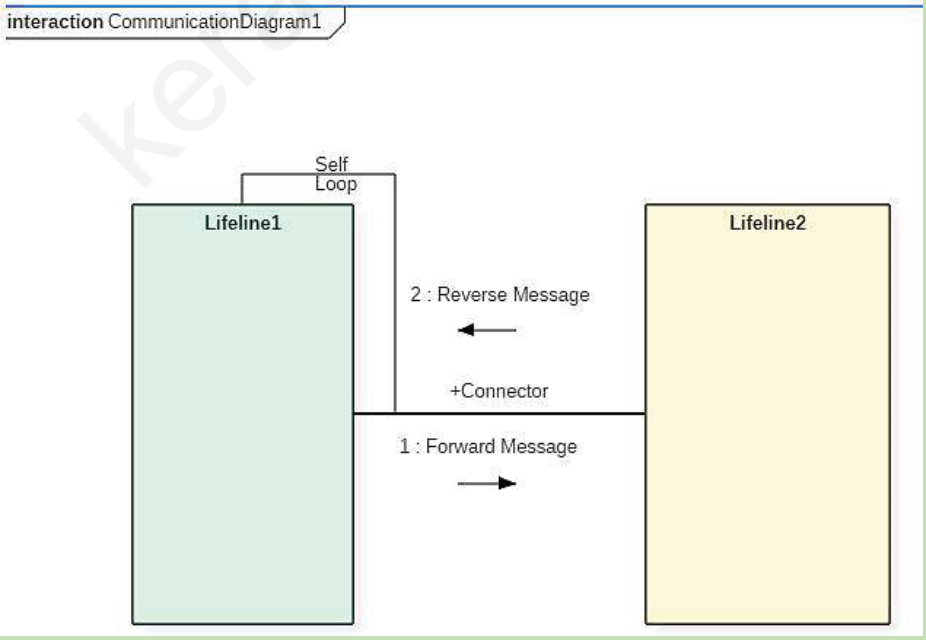
- Sequence diagrams are used to explore any real application or a system.
- Sequence diagrams are used to represent message flow from one object to another object.
- Sequence diagrams are easier to maintain.
- Sequence diagrams are easier to generate.
- Sequence diagrams can be easily updated according to the changes within a system.
- Sequence diagram allows reverse as well as forward engineering.

## Drawbacks of a sequence diagram

- Sequence diagrams can become complex when too many lifelines are involved in the system.
- If the order of message sequence is changed, then incorrect results are produced.
- Each sequence needs to be represented using different message notation, which can be a little complex.
- The type of message decides the type of sequence inside the diagram



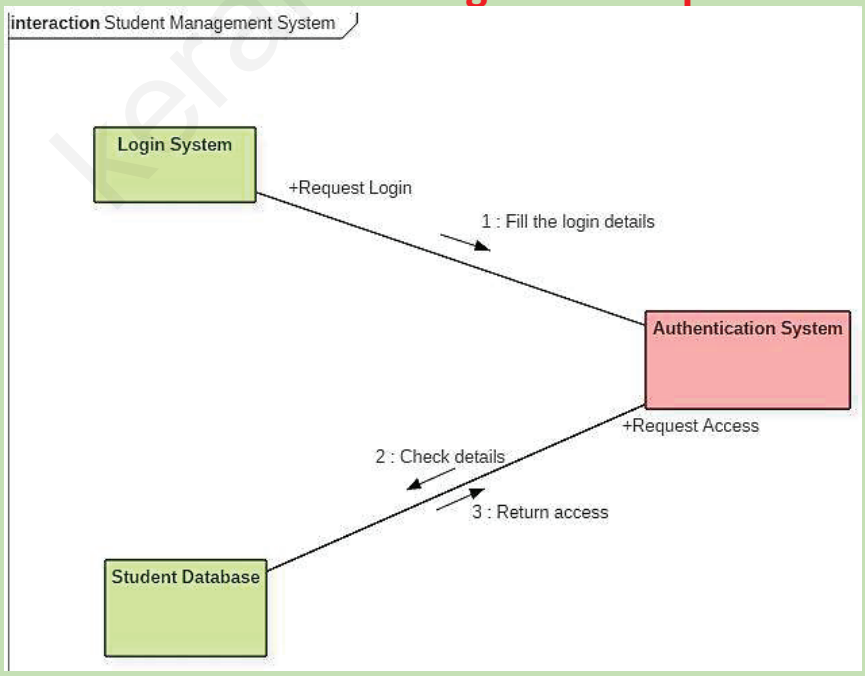
## How to draw a Collaboration /Communication Diagram



- As per Object-Oriented Programming (OOPs), an object entity has various attributes associated with it.
- Usually, there are multiple objects present inside an object-oriented system where each object can be associated with any other object inside the system
- Collaboration Diagrams are used to explore the architecture of objects inside the system.
- The message flow between the objects can be represented using a collaboration diagram.



## Collaboration Diagram Example



➢The above collaboration diagram represents a student information management system. The flow of communication in the above diagram is given by,

- A student requests a login through the login system.
- An authentication mechanism of software checks the request.
- If a student entry exists in the database, then the access is allowed; otherwise, an error is returned.





## Benefits of Collaboration Diagram

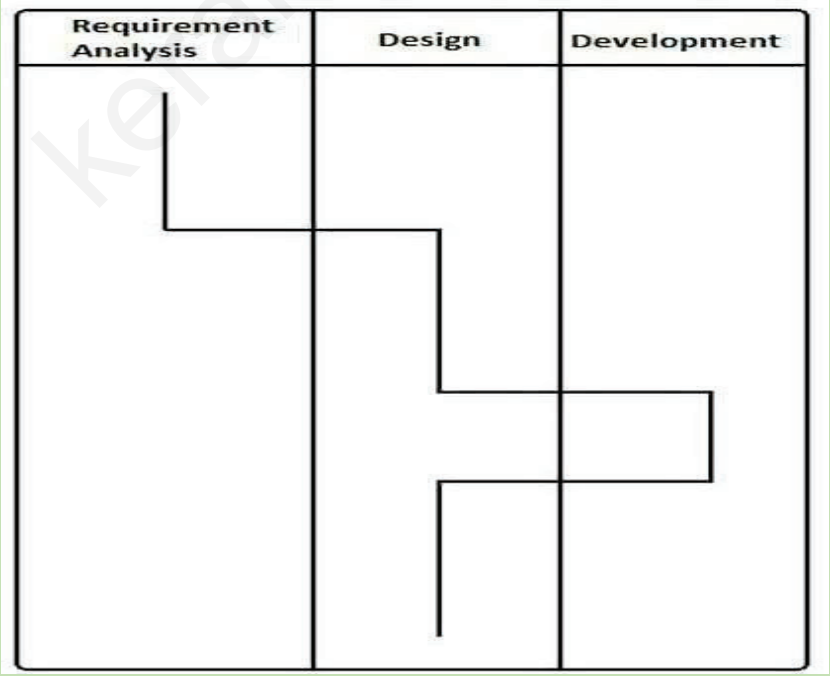
- It is also called as a communication diagram.
- It emphasizes the structural aspects of an interaction diagram - how lifeline connects.
- Its syntax is similar to that of sequence diagram except that lifeline don't have tails.
- Messages passed over sequencing is indicated by numbering each message hierarchically.
- It allows you to focus on the elements rather than focusing on the message flow as described in the sequence diagram.
- Sequence diagrams can be easily converted into a collaboration diagram as collaboration diagrams are not very expressive.

## Drawbacks of a Collaboration Diagram

- Collaboration diagrams can become complex when too many objects are present within the system.
- It is hard to explore each object inside the system.
- Collaboration diagrams are time consuming.
- The object is destroyed after the termination of a program.
- The state of an object changes momentarily, which makes it difficult to keep track of every single change the occurs within an object of a system.



## How to draw a Timing Diagram



- In the above diagram, first, the software passes through the requirements phase then the design and later the development phase.
- The output of the previous phase at that given instance of time is given to the second phase as an input
- Thus, the timing diagram can be used to describe SDLC (Software Development Life Cycle) in UML.





## Benefits of a Timing Diagram

- Timing diagrams are used to represent the state of an object at a particular instance of time.
- Timing diagram allows reverse as well as forward engineering.
- Timing diagram can be used to keep track of every change inside the system.

## Drawbacks of a Timing Diagram

- Timing diagrams are difficult to understand.
- Timing diagrams are difficult to maintain.

## ❖ACTIVITY DIAGRAM

- 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 basic purpose of activity diagrams is to capture the dynamic behavior of the system
- It is also called object-oriented flowchart
- 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.



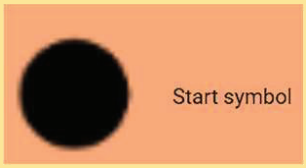


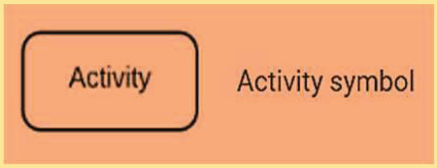
## Basic components of an activity diagram

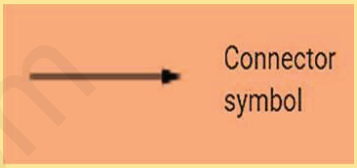
- **Action:** A step in the activity wherein the users or software perform a given task.
- **Decision node:** A conditional branch in the flow that is represented by a diamond. It includes a single input and two or more outputs.
- **Control flows:** Another name for the connectors that show the flow between steps in the diagram.
- **Start node:** Symbolizes the beginning of the activity. The start node is represented by a black circle.
- **End node:** Represents the final step in the activity. The end node is represented by an outlined black circle.

## Activity diagram symbols

- Start symbol - Represents the beginning of a process or workflow in an activity diagram.
- Activity symbol - Indicates the activities that make up a modeled process. These symbols, which include short descriptions within the shape, are the main building blocks of an activity diagram.
- Connector symbol - Shows the directional flow, or control flow, of the activity.

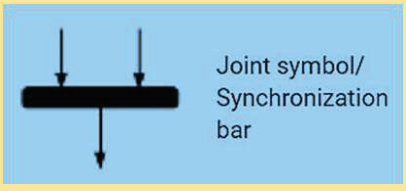


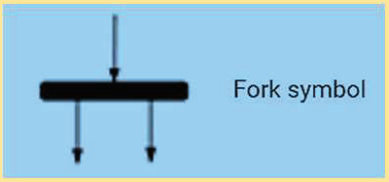



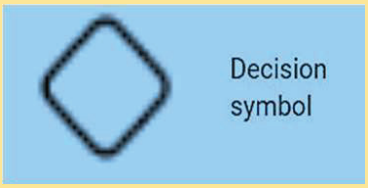





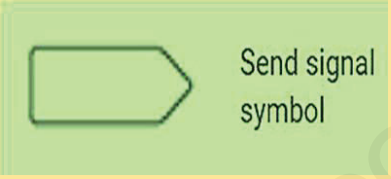

- Joint symbol / Synchronization bar - Combines two concurrent activities and re-introduces them to a flow where only one activity occurs at a time. Represented with a thick vertical or horizontal line.
- Fork symbol - Splits a single activity flow into two concurrent activities. Symbolized with multiple arrowed lines from a join.
- Decision symbol - Represents a decision and always has at least two paths branching out with condition text.



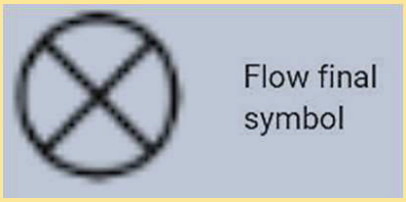
- Note symbol - Allows the diagram creators or collaborators to communicate additional messages that don't fit within the diagram itself. Leave notes for added clarity and specification.
- Send signal symbol - Indicates that a signal is being sent to a receiving activity
- Receive signal symbol - Demonstrates the acceptance of an event. After the event is received, the flow that comes from this action is completed.

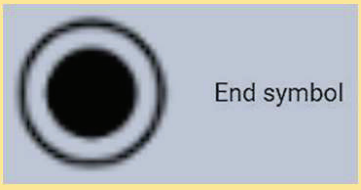


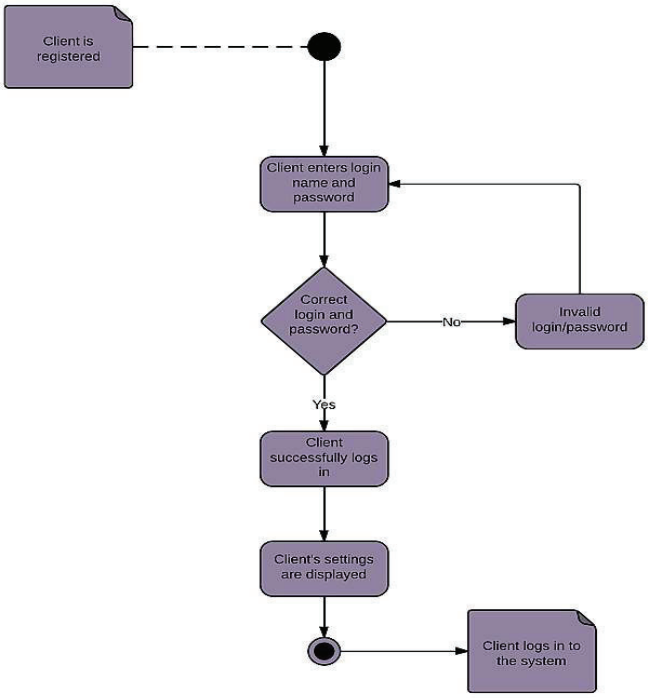


- Flow final symbol - Represents the end of a specific process flow. This symbol shouldn't represent the end of all flows in an activity. The flow final symbol should be placed at the end of a single activity flow.
- Condition text - Placed next to a decision marker to let you know under what condition an activity flow should split off in that direction
- End symbol - Marks the end state of an activity and represents the completion of all flows of a process.



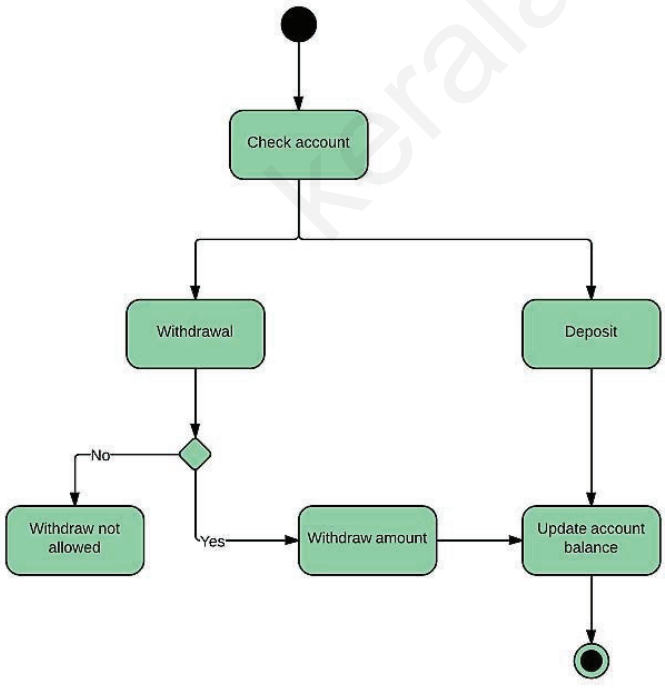







## Activity diagram - a login page






## Activity Diagram - Banking system.

## ❖STATE CHART DIAGRAM

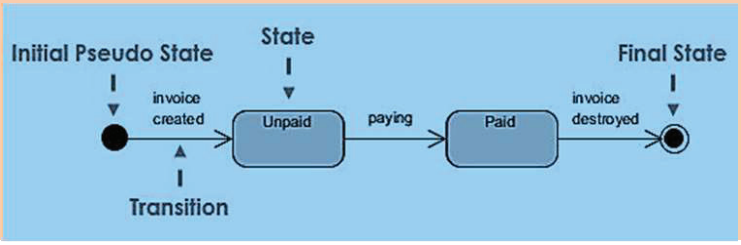
- State chart diagram is used to capture the dynamic aspect of a system
- An object goes through various states during its lifespan. The lifespan of an object remains until the program is terminated. The object goes from multiple states depending upon the event that occurs within the object.
- Each state represents some unique information about the object.
- State chart diagram visualizes the flow of execution from one state to another state of an object.
- It represents the state of an object from the creation of an object until the object is destroyed or terminated.



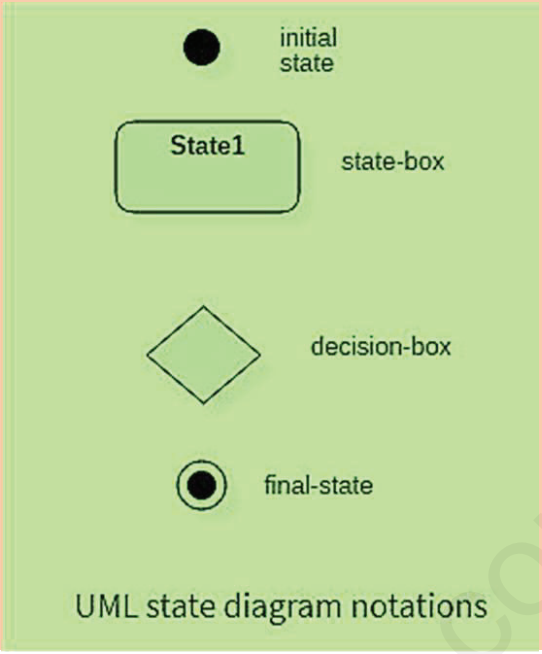


- The primary purpose of a state chart diagram is to model interactive systems and define each and every state of an object.
- State chart diagrams are also referred to as State machines and state diagrams.
- A state machine consists of states, linked by transitions. A state is a condition of an object in which it performs some activity or waits for an event

Simple State Machine Diagram Notation



## Notation and Symbol for State Machine / State Chart Diagram






- **Initial state** - The initial state symbol is used to indicate the beginning of a state machine diagram.
- **Final state** - This symbol is used to indicate the end of a state machine diagram.
- **Decision box** - It contains a condition. Depending upon the result of an evaluated guard condition, a new path is taken for program execution.
- **Transition** - A transition is a change in one state into another state which is occurred because of some event. A transition causes a change in the state of an object.

- **State box**

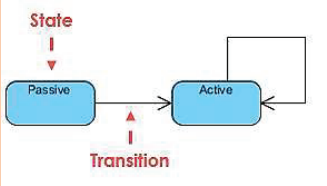
➢States represent situations during the life of an object.

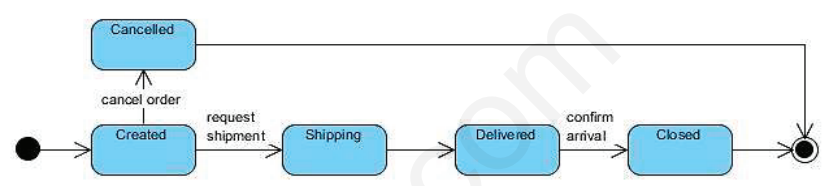
➢It is denoted using a rectangle with round corners.

➢The name of a state is written inside the rounded rectangle.

➢ A state can be either active or inactive.

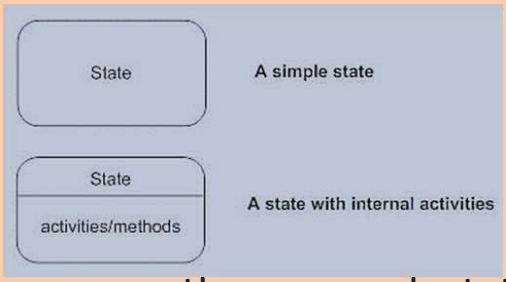
➢When a state is in the working mode, it is active, as soon as it stops executing and transits into another state, the previous state becomes inactive, and the current state becomes active.






## Types of State




➢Simple state

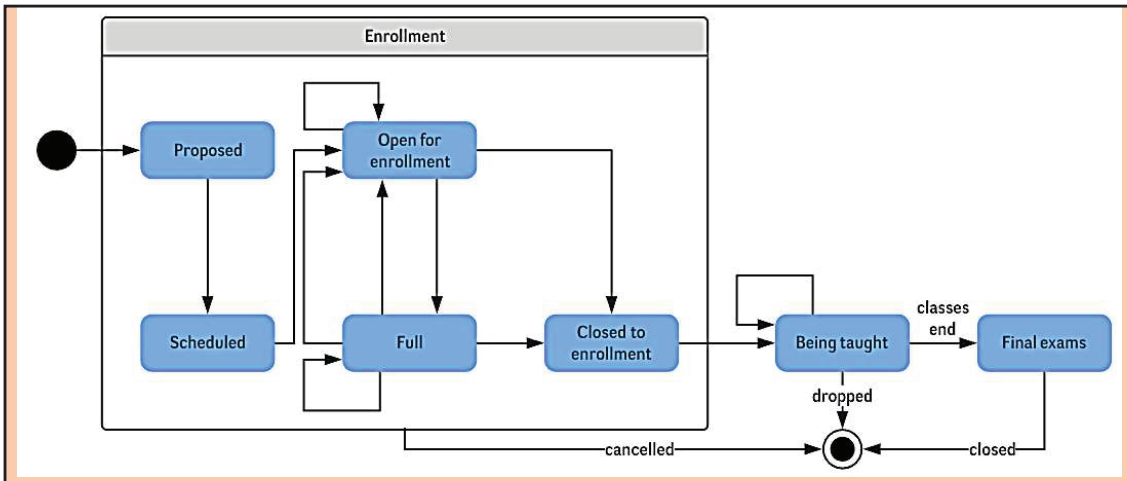
- They do not have any sub state.

➢Composite state

- These types of states can have one or more than one sub state.
- A composite state with two or more sub states is called an orthogonal state.

➢Submachine state

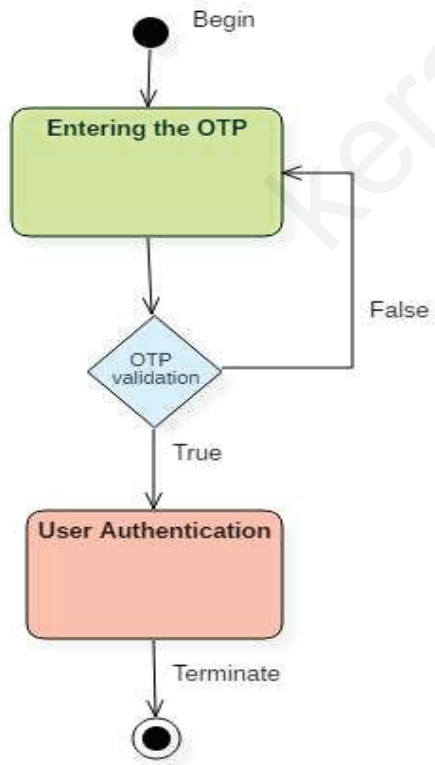
- These states are semantically equal to the composite states
- Unlike the composite state, we can reuse the submachine states.




University state Diagram

- The composite state “Enrollment” is made up of various sub states that will lead students through the enrollment process.
- Once the student has enrolled, they will proceed to “Being taught” and finally to “Final exams.”






Eg: state chart diagram
user authentication process.

## State machine vs. Flowchart

| Statemachine | FlowChart |
|---|---|
| It represents various states of a system. | The Flowchart illustrates the program execution flow. |
| The state machine has a WAIT concept, i.e., wait for an action or an event. | The Flowchart does not deal with waiting for a concept. |
| State machines are used for a live running system. | Flowchart visualizes branching sequences of a system. |
| The state machine is a modeling diagram. | A flowchart is a sequence flow or a DFD diagram. |
| The state machine can explore various states of a system. | Flowchart deal with paths and control flow. |

# MODULE 1

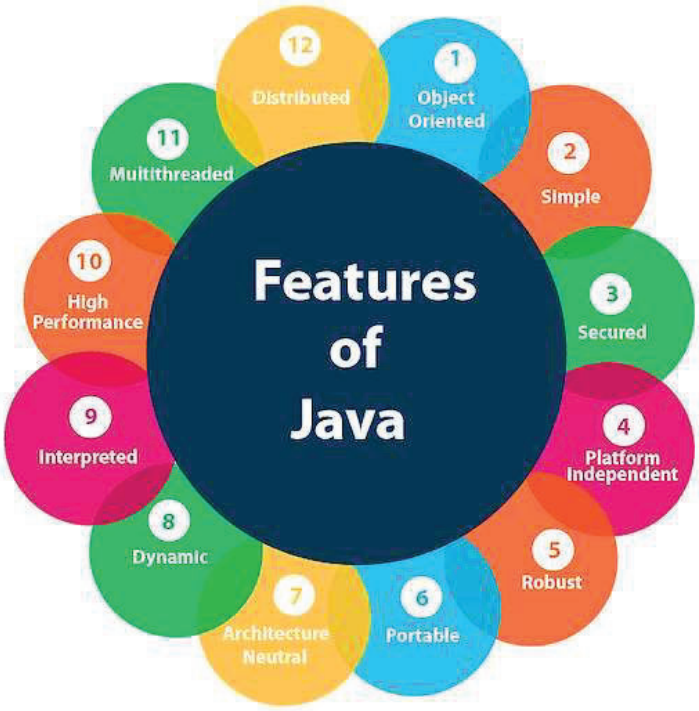
## CHAPTER 2
## INTRODUCTION TO JAVA





# JAVA

- Java is a powerful general-purpose , Object Oriented programming language developed by Sun Micro System of USA in 1991.
- Development team members are James Gosling, Patrick Naughton, Chris Warth, Ed Frank, and Mike Sheridan
- First name of Java is "Oak," but was renamed "Java" in 1995.
- Java derives much of its character from C and C++.
- Java Changed the Internet by simplifying web programming
- Java innovated a new type of networked program called the applet




## FEATURES OF JAVA
## ( Java Buzzwords)



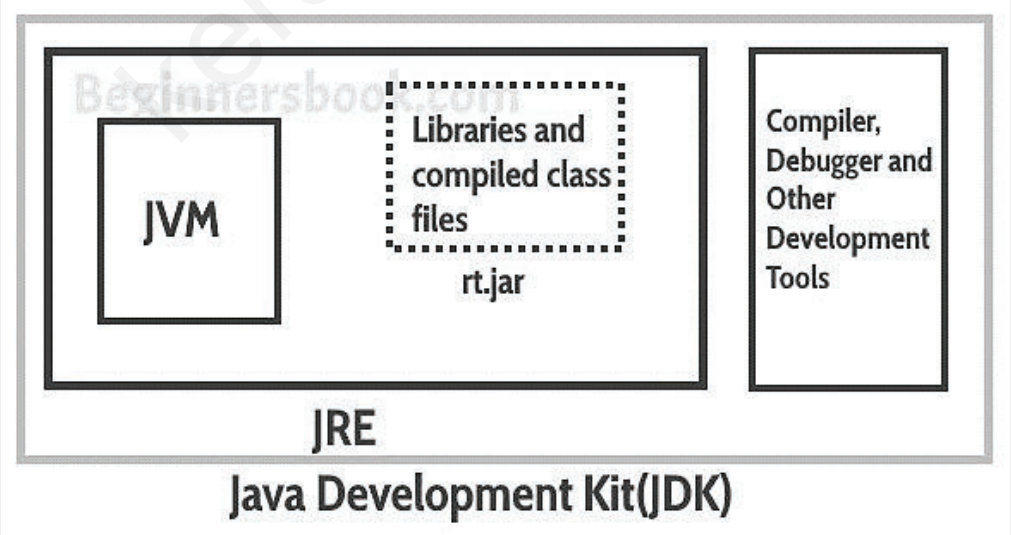
## ❖ JAVA RUNTIME ENVIRONMENT (JRE)

- A software program needs an environment to run .
- The runtime environment loads class files and ensures there is access to memory and other system resources to run them.
- Java Runtime Environment provides the minimum requirements for executing a Java application programs.
- JRE is an installation package which provides environment to only run(not develop) the java program(or application)onto your machine.
- JRE is only used by them who only wants to run the Java Programs i.e. end users of your system. JRE can be view as a subset of JDK.

## ❖ JAVA DEVELOPMENT KIT (JDK)

- The Java Development Kit (JDK) is a software development environment used for developing and executing Java applications and applets
- It includes the Java Runtime Environment (JRE), an interpreter/loader (Java), a compiler (javac), an archiver (jar), a documentation generator (Javadoc) and other tools needed in Java development.
- JDK is only used by Java Developers.





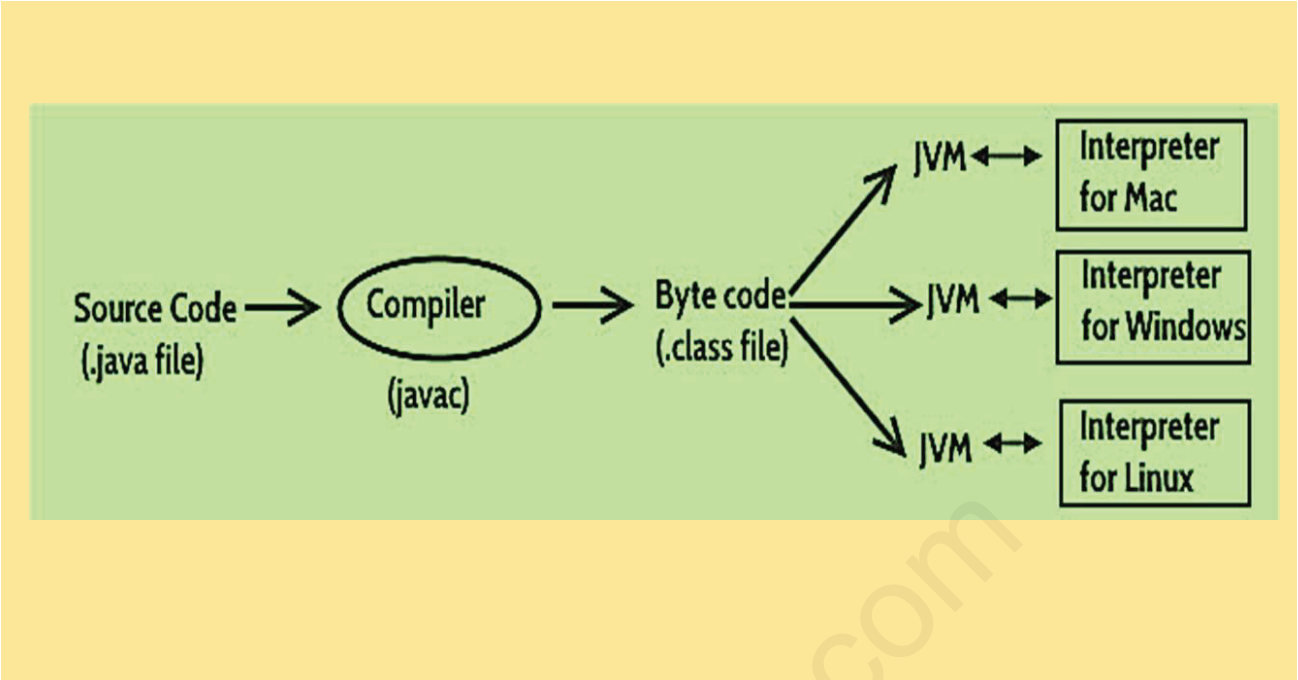
## ❖ JAVA VIRTUAL MACHINE (JVM)

- JVM is a program which provides the runtime environment to execute Java programs. Java programs cannot run if a supporting JVM is not available.
- JVM is a virtual machine that resides in the real machine (your computer) and the machine language for JVM is byte code.
- The Java compiler generate byte code for JVM rather than different machine code for each type of machine.
- JVM executes the byte code generated by compiler and produce output.
- JVM is the one that makes java platform independent.





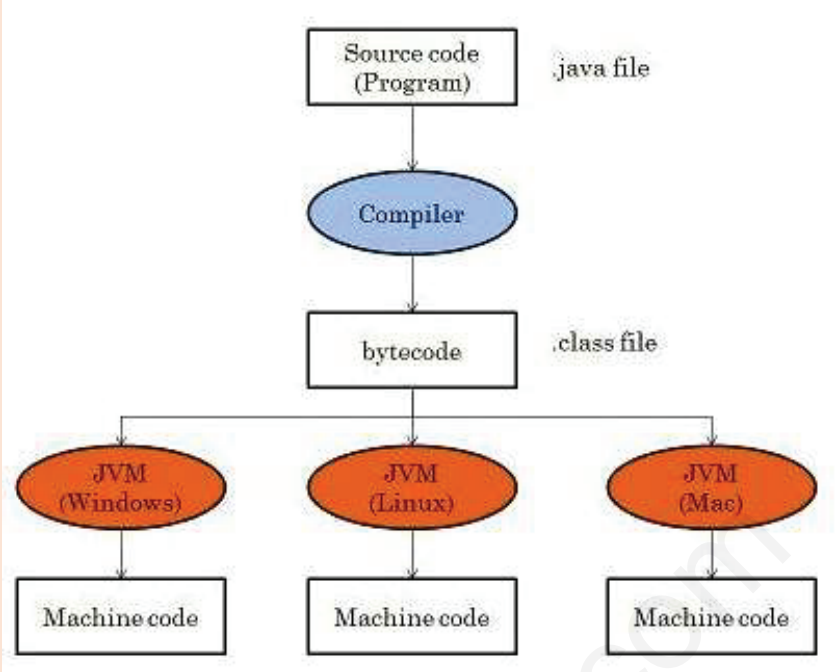
- The primary function of JVM is to execute the byte code produced by compiler
- The JVM doesn't understand Java source code, that's why we need to have **javac** compiler
- Java compiler (javac) compiles *.java files to obtain *.class files that contain the byte codes understood by the JVM.
- JVM makes java portable (write once, run anywhere).
- Each operating system has different JVM, however the output they produce after execution of byte code is same across all operating systems.






## ❖ BYTE CODE

- Java byte code is the instruction set for the Java Virtual Machine
- It is the machine code in the form of a .class file.
- Byte code is a machine independent code
- It is not completely a compiled code but it is an intermediate code somewhere in the middle which is later interpreted and executed by JVM.
- Byte code is a machine code for JVM.
- Byte code implementation makes Java a platform- Independent language.






## ❖JAVA COMPILER

- Java is compiled language. But it is very different from traditional compiling in the way that after compilation source code is converted to byte code.
- **Javac** is the most popular Java compiler
- Java has a virtual machine called JVM which then converts byte code to target code of machine on which it is run.
- JVM performs like an interpreter. It doesn't do it alone, though. It has its own compiler to convert the byte code to machine code. This compiler is called **Just In Time** or **JIT compiler.**

## ❖JAVA APPLET

- An *applet* is a special kind of Java program that is designed to be transmitted over the Internet and automatically executed by a Java-compatible web browser
- It runs inside the web browser and works at client side
- Applets are used to make the web site more dynamic and entertaining
- Applets are not stand-alone programs. Instead, they run within either a web browser or an applet viewer. JDK provides a standard applet viewer tool called applet viewer.
- In general, execution of an applet does not begin at main() method.




## Lifecycle of Java Applet

## Java Applet vs Java Application

| Java Application | Java Applet |
|---|---|
| Java Applications are the stand-alone programs which can be executed independently | Java Applets are small Java programs which are designed to exist within HTML web document |
| Java Applications must have main() method for them to execute | Java Applets do not need main() for execution |
| Java Applications just needs the JRE | Java Applets cannot run independently and require API's |
| Java Applications do not need to extend any class unless required | Java Applets must extend java.applet.Applet class |
| Java Applications can execute codes from the local system | Java Applets Applications cannot do so |
| Java Applications has access to all the resources available in your system | Java Applets has access only to the browser-specific services |





# JAVA BUZZWORDS

### ➢Simple

- It's simple and easy to learn if you already know the basic concepts of Object Oriented Programming.
- C++ programmer can move to JAVA with very little effort to learn.
- Java syntax is based on C++
- Java has removed many complicated and rarely-used features, for example, explicit pointers, operator overloading, etc.

## ➢Object oriented

- Java is true object oriented language. Everything in Java is an object.
- All program code and data reside within objects and classes.
- Java comes with an extensive set of classes, arranged in packages that can be used in our programs through inheritance.

## ➢Distributed

- Java is designed for distributed environment of the Internet. Its used for creating applications on networks
- Java enables multiple programmers at multiple remote locations to collaborate and work together on a single project.





## ➢Compiled and Interpreted

- Usually a computer language is either compiled or Interpreted. Java combines both this approach and makes it a two-stage system.
- Compiled : Java enables creation of a cross platform programs by compiling into an intermediate representation called Java Byte code.
- Interpreted : Byte code is then interpreted, which generates machine code that can be directly executed by the machine that provides a Java Virtual machine.

## ➢Robust

- It provides many features that make the program execute reliably in variety of environments.
- Java is a strictly typed language. It checks code both at compile time and runtime.
- Java takes care of all memory management problems with garbage-collection.
- Java, with the help of exception handling captures all types of serious errors and eliminates any risk of crashing the system.





## ➢Secure

- Java provides a "firewall" between a networked application and your computer.
- When a Java Compatible Web browser is used, downloading can be done safely without fear of viral infection or malicious intent.
- Java achieves this protection by confining a Java program to the java execution environment and not allowing it to access other parts of the computer.

## ➢Architecture Neutral

- Java language and Java Virtual Machine helped in achieving the goal of "write once; run anywhere, any time, forever."
- Changes and upgrades in operating systems, processors and system resources will not force any changes in Java Programs.

## ➢Portable

- Java is portable because it facilitates you to carry the Java byte code to any platform. It doesn't require any implementation.
- Java Provides a way to download programs dynamically to all the various types of platforms connected to the Internet.

## ➢High Performance

- Java performance is high because of the use of byte code.
- The byte code can be easily translated into native machine code.





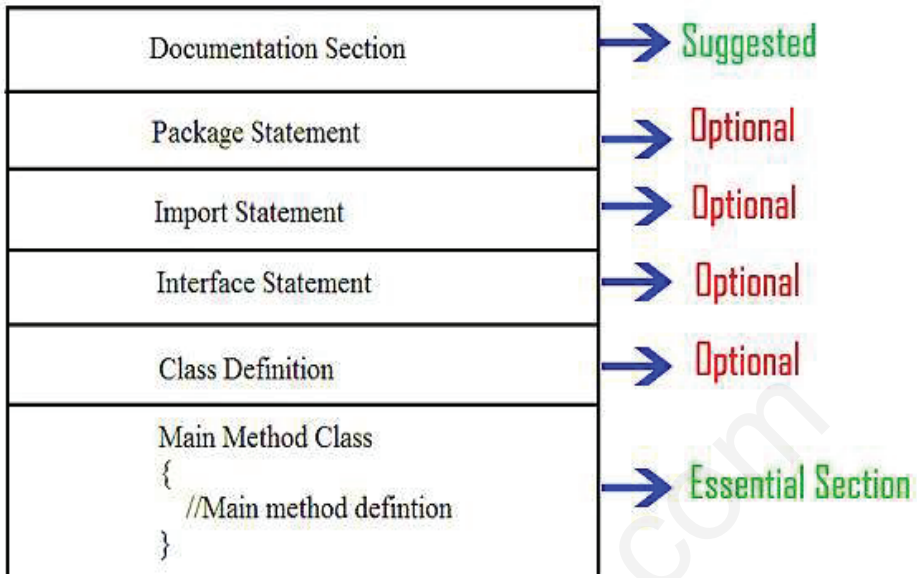
## ➢Multithreaded

- Multithreaded Programs handled multiple tasks simultaneously, which was helpful in creating interactive, networked programs.
- Java run-time system comes with tools that support multiprocess synchronization used to construct smoothly interactive systems

## ➢Dynamic

- Java is capable of linking in new class libraries, methods, and objects.
- It supports functions from native languages (the functions written in other languages such as C and C++).
- It supports dynamic loading of classes. It means classes are loaded on demand

# JAVA PROGRAM STRUCTURE






## ➢Documentation Section

- You can write a comment in this section. It helps to understand the code. These are optional
- It is used to improve the readability of the program.
- The compiler ignores these comments during the time of execution
- There are three types of comments that Java supports

➢Single line Comment //This is single line comment

➢Multi-line Comment /* this is multiline comment.
and support multiple lines*/

➢Documentation Comment /** this is documentation cmnt*/

## ➢Package Statement

- We can create a package with any name. A package is a group of classes that are defined by a name.
- That is, if you want to declare many classes within one element, then you can declare it within a package
- It is an optional part of the program, i.e., if you do not want to declare any package, then there will be no problem with it, and you will not get any errors.
- Package is declared as: package package_name;

Eg: package mypackage;





## ➢Import Statement

- If you want to use a class of another package, then you can do this by importing it directly into your program.
- Many predefined classes are stored in packages in Java
- We can import a specific class or classes in an import statement.

Examples:

```
import java.util.Date;     //imports the date class

import java.applet.*;    /*imports all the classes from the java
                           applet package*/
```

## ➢Interface Statement

- This section is used to specify an interface in Java
- Interfaces are like a class that includes a group of method declarations
- It's an optional section and can be used when programmers want to implement multiple inheritances within a program.

## ➢Class Definition

- A Java program may contain several class definitions.
- Classes are the main and essential elements of any Java program.
- A class is a collection of variables and methods





## ➢Main Method Class

- The main method is from where the execution actually starts and follows the order specified for the following statements
- Every Java stand-alone program requires the main method as the starting point of the program.
- This is an essential part of a Java program.
- There may be many classes in a Java program, and only one class defines the main method
- Methods contain data type declaration and executable statements.

## A simple java program to print hello world

```
public class Hello
 {
    //main method declaration
   public static void main(String[] args)
     {
        System.out.println("hello world");
      }
  }
```





- **public class Hello -** This creates a class called Hello. We should make sure that the class name starts with a capital letter, and the public word means it is accessible from any other classes.
- **Braces -** The curly brackets are used to group all the commands together
- **public static void main**

- When the main method is declared public, it means that it can be used outside of this class as well.
- The word static means that we want to access a method without making its objects
- The word void indicates that it does not return any value. The main is declared as void because it does not return any value.
- *main* is a method; this is a starting point of a Java program.

➢**String[] args**

It is an array where each element is a string, which is named as args. If you run the Java code through a console, you can pass the input parameter. The main() takes it as an input.

➢**System.out.println();**

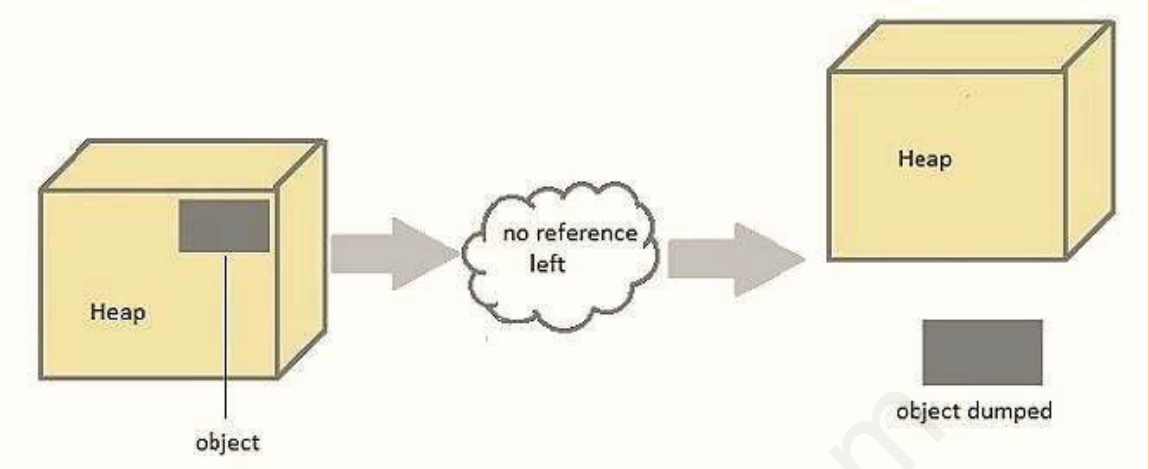
- This statement is used to print text on the screen as output
- system is a predefined class, and out is an object of the PrintWriter class defined in the system
- The method println prints the text on the screen with a new line.
- We can also use print() method instead of println() method. All Java statement ends with a semicolon.





## Garbage Collection in Java
(A process of releasing unused memory)

- When JVM starts up, it creates a heap area which is known as runtime data area. This is where all the objects (instances of class) are stored
- Since this area is limited, it is required to manage this area efficiently by removing the objects that are no longer in use.
- The process of removing unused objects from heap memory is known as Garbage collection and this is a part of memory management in Java.
- Languages like C/C++ don't support automatic garbage collection, however in java, the garbage collection is automatic.






- In java, garbage means unreferenced objects.
- Main objective of Garbage Collector is to free heap memory by destroying unreachable objects.
- Unreachable objects : An object is said to be unreachable iff it doesn't contain any reference to it.
- Eligibility for garbage collection : An object is said to be eligible for GC(garbage collection) iff it is unreachable.
- **finalize() method** – This method is invoked each time before the object is garbage collected and it perform cleanup processing.
- The Garbage collector of JVM collects only those objects that are created by new keyword. So if we have created any object without new, we can use finalize method to perform cleanup processing

## Request for Garbage Collection

- We can request to JVM for garbage collection however, it is upto the JVM when to start the garbage collector.
- Java **gc()** method is used to call garbage collector explicitly.
- However gc() method does not guarantee that JVM will perform the garbage collection.
- It only request the JVM for garbage collection. This method is present in **System and Runtime class**.

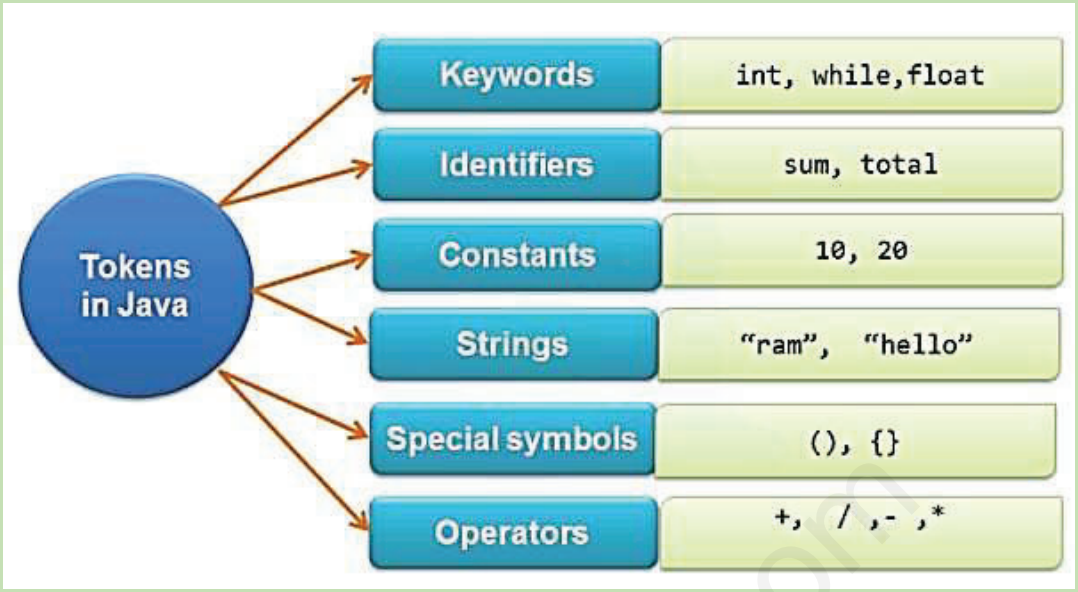




# Java Lexical Issues (Java Tokens)

## ❖TOKENS

- Java Tokens are the smallest individual building block or smallest unit of a Java program
- Java program is a collection of different types of tokens, comments, and white spaces.






## ➢Keywords

- A keyword is a reserved word. You cannot use it as a variable name, constant name etc.
- The meaning of the keywords has already been described to the java compiler. These meaning cannot be changed.
- Thus, the keywords cannot be used as variable names because that would try to change the existing meaning of the keyword, which is not allowed.
- Java language has reserved 50 words as keywords

| Keywords in Java | | | | |
|---|---|---|---|---|
| abstract | default | if | private | this |
| assert | do | implements | protected | throw |
| boolean | double | import | public | throws |
| break | else | instanceof | return | transient |
| byte | enum | int | short | try |
| case | extends | interface | static | void |
| catch | final | long | strictfp | volatile |
| char | finally | native | super | while |
| class | float | new | switch | |
| continue | for | package | synchronized | |



## ➢Identifiers

- Identifiers are the names of variables, methods, classes, packages and interfaces
- Identifier must follow some rules.

✓All identifiers must start with either a letter( a to z or A to Z ) or currency character($) or an underscore.

✓They must not begin with a digit

✓After the first character, an identifier can have any combination of characters.

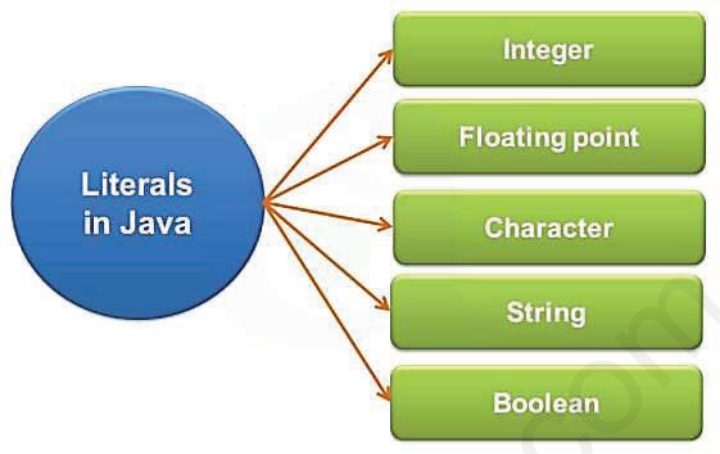
✓A Java keywords cannot be used as an identifier.

✓Identifiers in Java are case sensitive, foo and Foo are two different identifiers.

✓They can be any length Eg: int a; char name;

## ➢Constants or Literals

- Constants are fixed values of a particular type of data,which cannot be modified in a program.
- Java language specifies five major type of literals.






**Eg: Integer literal : 100**

**Floating-point literal : 98.6**

**Character literal : 's'**

**String literal : "sample"**

## ➢Comments

| Comment type | Meaning |
|---|---|
| // comment | Single-line comments |
| /* comment */ | Multi-line comments |
| /** documentation */ | Documentation comments |

## ➢ String

- In java, string is basically an object that represents sequence of char values.
- An array of characters works same as java string.

  Eg: char[] ch = {'a','t','n','y','l','a'};

  String s = "atnyla";

- Java String class provides a lot of methods to perform operations on string such as compare(), concat(), equals(), split(), length(), replace(), compareTo(), intern(), substring() etc.



## ➢ Special symbol

| | | | | |
|---|---|---|---|---|
| , | < | > | . | _ |
| ( | ) | ; | $ | : |
| % | [ | ] | # | ? |
| ' | & | { | } | " |
| ^ | ! | * | / | \| |
| - | \ | ~ | + | |

Brackets[] : Opening and closing brackets are used as array element reference. These indicate single and multidimensional subscripts.

Parentheses() : These special symbols are used to indicate function calls and function parameters.

Braces{} : These opening and ending curly braces mark the start and end of a block of code containing more than one executable statement.

semicolon ; : It is used to separate more than one statements like in for loop is separates initialization, condition, and increment.

comma , : It is an operator that essentially invokes something called an initialization list.

asterisk * : It is used for multiplication.

assignment operator = : It is used to assign values.

Period . : Used to separate package names from subpackages and classes



## ➢Operators

- An operator is a symbol that takes one or more arguments and operates on them to produce a result.
- Unary Operator
- Arithmetic Operator
- shift Operator
- Relational Operator
- Bitwise Operator
- Logical Operator
- Ternary Operator
- Assignment Operator

## ➢Whitespace

- Java is a free-form language. This means that you do not need to follow any special indentation rules
- White space in Java is used to separate tokens in the source file. It is also used to improve readability of the source code.

  Eg: int i = 0;

- White spaces are required in some places. For example between the int keyword and the variable name.
- In java whitespace is a space, tab, or newline