Noida Institute of Engineering and Technology, Greater Noida

OBJECT ORIENTED TECHNIQUES USING JAVA

Unit: 1

Introduction

Course Details
(B Tech 3rdSem /2nd Year)

Bhoomika Kaushik

CONTENTS

- Object Oriented Programming
- Introduction and Features
- Abstraction
- Encapsulation
- Polymorphism
- Inheritance.
- Modeling Concepts
- Class Diagram

CONTENTS

- Object Diagram.
- Control Statements
- Decision Making
- Looping
- Branching
- Argument Passing Mechanism
- Command Line Argument.

COURSE OBJECTIVES

- The objective of this course is to understand the object-oriented methodology and its techniques to design and develop conceptual models and demonstrate the standard concepts of object-oriented techniques modularity, I/O. and other standard language constructs.

- The basic objective of this course is to understand the fundamental concepts of object-oriented programming in Java language and also implement the Multithreading concepts, GUI based application and collection framework.

COURSE OUTCOME

After completion of this course students will be able to:

CO1	**Identify the concepts of object oriented programming and relationships among them needed in modeling.**
CO2	Demonstrate the Java programs using OOP principles and also implement the concepts of lambda expressions.
CO3	Implement packages with different protection level resolving namespace collision and evaluate the error handling concepts for uninterrupted execution of Java program.
CO4	Implement Concurrency control, I/O Streams and Annotations concepts by using Java program.
CO5	Design and develop the GUI based application, Generics and Collections in Java programming language to solve the real-world problem.

Prerequisite

- Student must know at least the basics of how to use a computer, and should be able to start a command line shell.

- Knowledge of basic programming concepts, as covered in ‘Programming Basic” course is necessary.

Object Oriented Programming

- The main aim of OOP is to bind together the data and the functions that operate on them so that no other part of the code can access this data except that function.
- In the object oriented paradigm, the list and the associated operations are treated as one entity known as an object.

The striking features of OOP include the following:

• The programs are data centered.

• Programs are divided in terms of objects and not procedures.

• Functions that operate on data are tied together with the data.

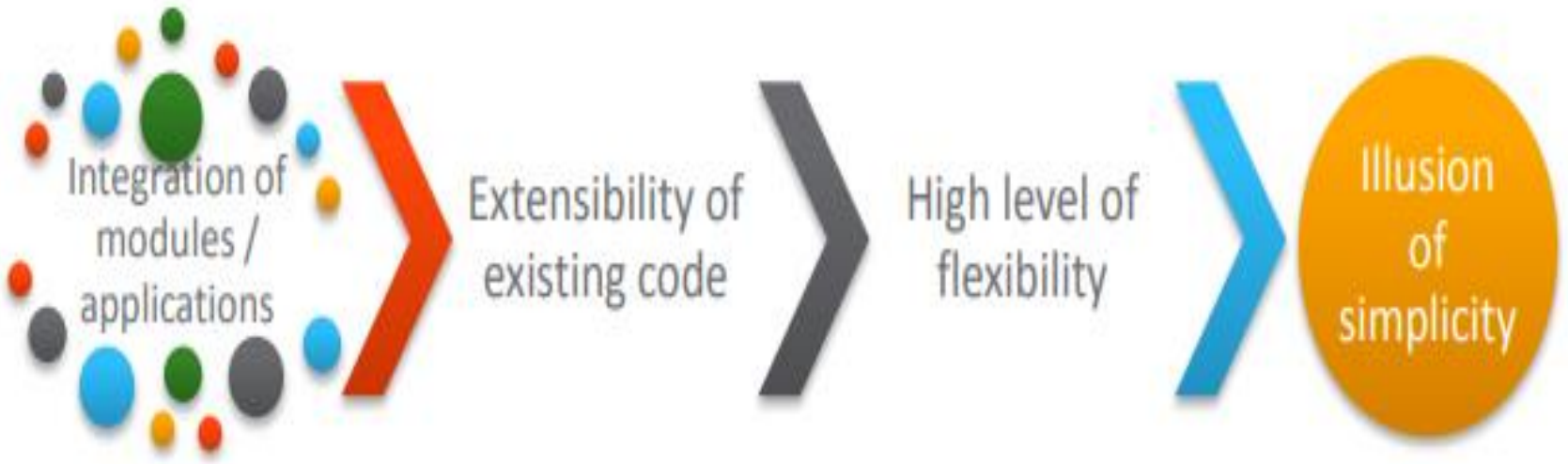
• Data is hidden and not accessible by external functions.

• New data and functions can be easily added as and when required.

• Follows a bottom-up approach for problem solving.

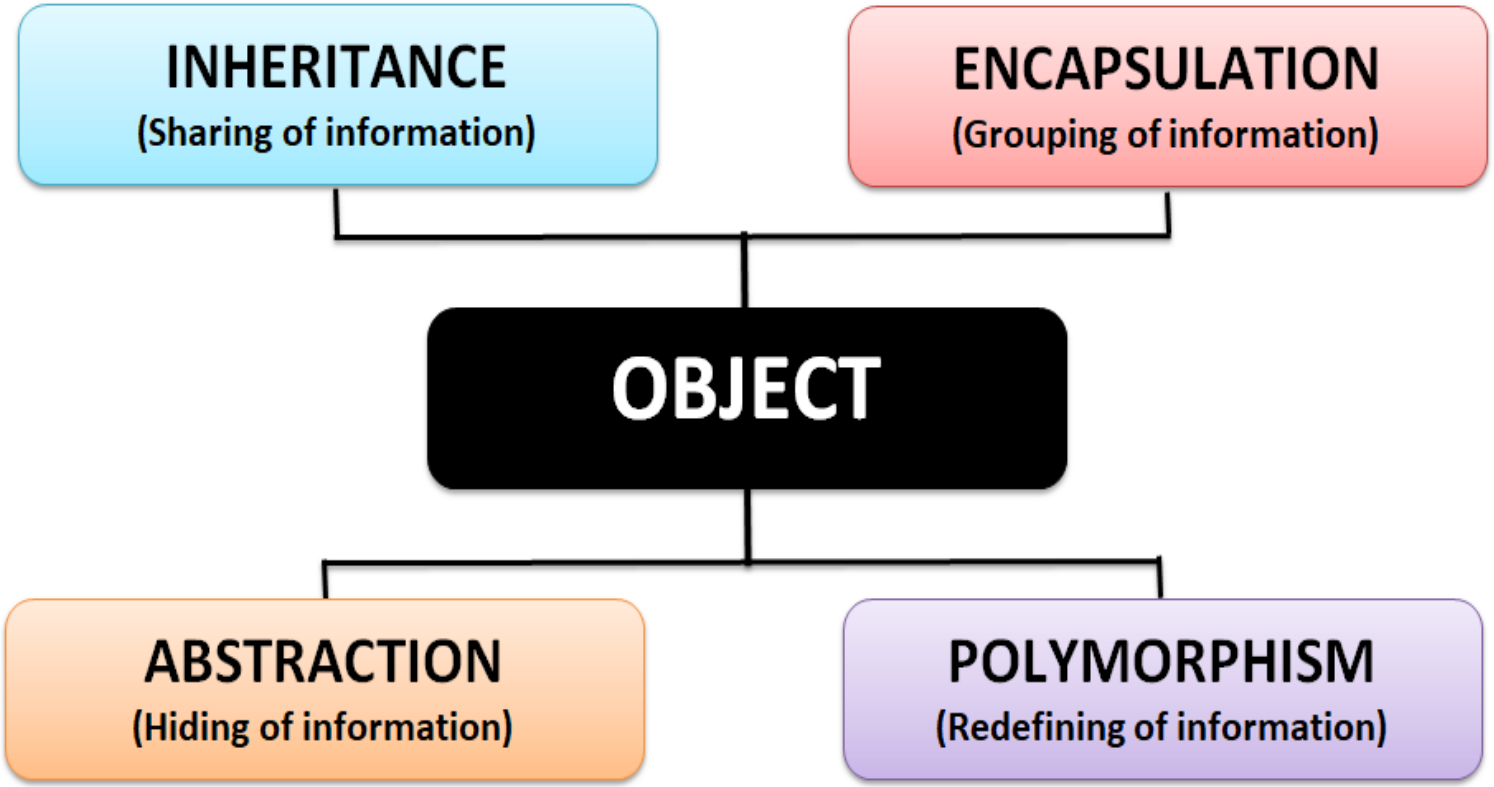
Need for Object Oriented Approach

- Challenges in developing a business application.



- If these challenges are not addressed, it may lead to software crisis.

Pillars of OOPS



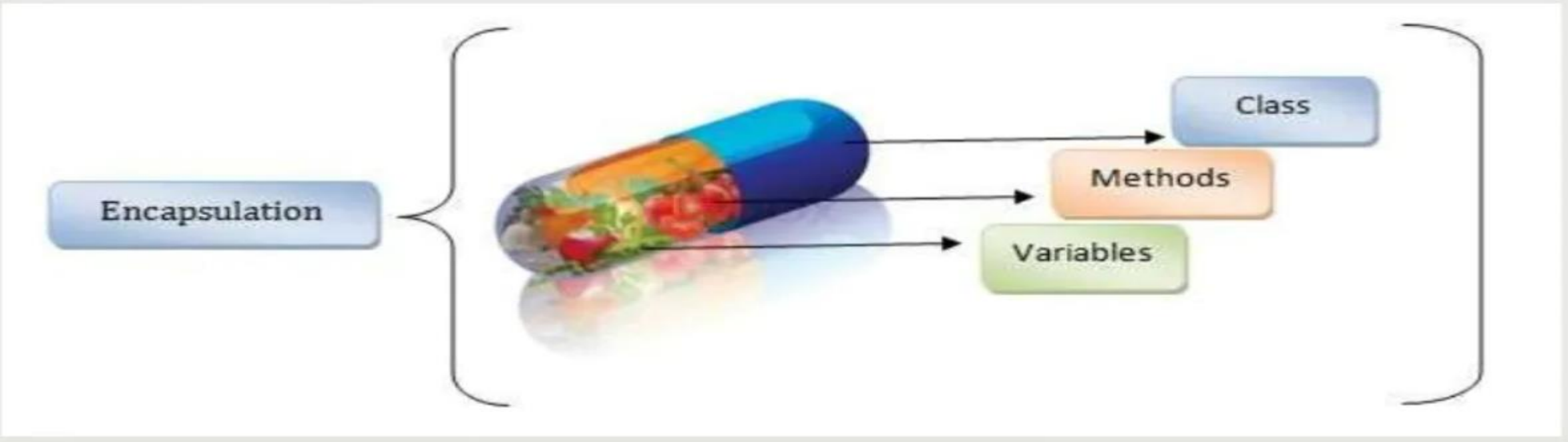
Encapsulation

- Binding the data with the code that manipulates it.
- It keeps the data and the code safe from external interference

Example :-

- A power steering mechanism of a car. Power steering of a car is a complex system, which internally have lots of components tightly coupled together, they work synchronously to turn the car in the desired direction. It even controls the power delivered by the engine to the steering wheel. But to the external world there is only one interface is available and rest of the complexity is hidden. Moreover, the steering unit in itself is complete and independent. It does not affect the functioning of any other mechanism.

Encapsulation



Abstraction

- Abstraction is a process of hiding the implementation details from the user, only the functionality will be provided to the user. In other words, the user will have the information on what the object does instead of how it does.
- **Example 1**, when you login to your Amazon account online, you enter your user_id and password and press login, what happens when you press login, how the input data sent to amazon server, how it gets verified is all abstracted away from the you.
- **Example 2**: A car in itself is a well-defined object, which is composed of several other smaller objects like a gearing system, steering mechanism, engine, which are again have their own subsystems. But for humans car is a one single object, which can be managed by the help of its subsystems, even if their inner details are unknown.

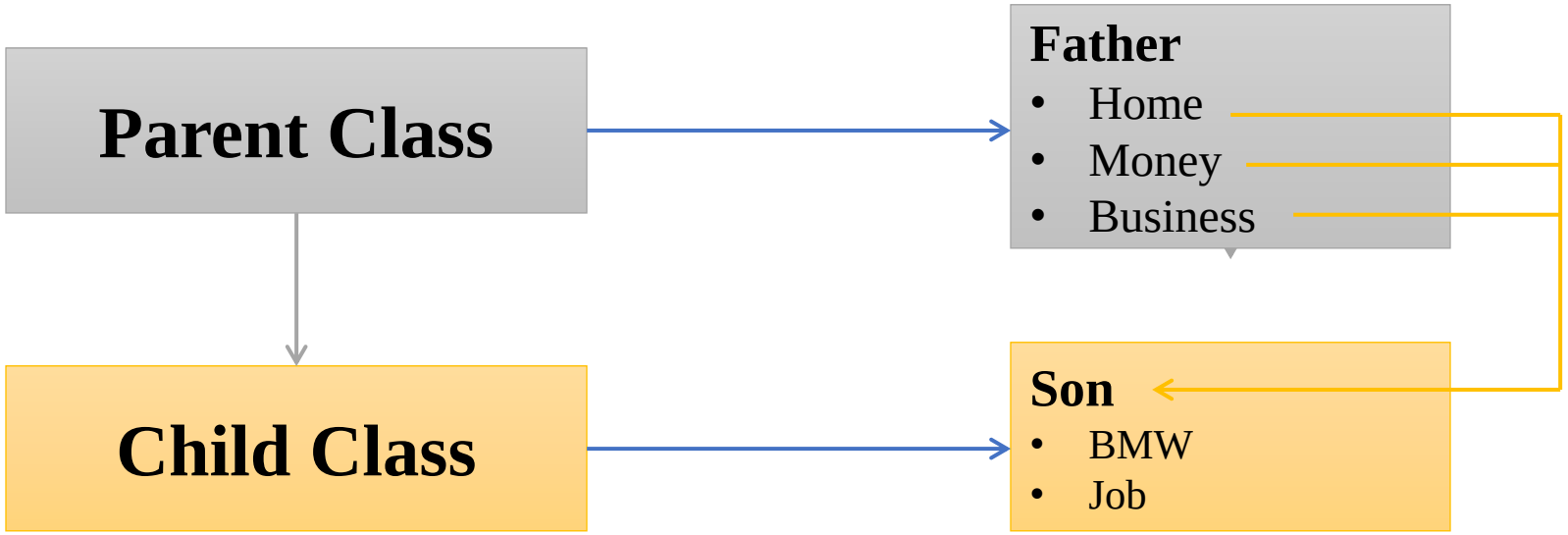
Inheritance

- The mechanism of deriving a new class from an old one (existing class) such that the new class inherit all the members (variables and methods) of old class is called inheritance or derivation.
- A class that is derived from another class is called a **subclass** (also a derived class, extended class or child class)
- The class from which the subclass is derived is called a **super class** or **base class** or **parent class**

Inheritance

The old class is referred to as the Super class and the new one is called the Sub class.

- Parent Class - Base Class or Super Class
- Child Class - Derived Class or Sub Class



Inheritance

For example:

- Car is a four wheeler vehicle so assume that we have a class FourWheeler and a sub class of it named Car. Here Car acquires the properties of a class FourWheeler. Other classifications could be a jeep, van etc.

- FourWheeler defines a class of vehicles that have four wheels, and specific range of engine power, load carrying capacity etc.

- Car (termed as a sub-class) acquires these properties from FourWheeler, and has some specific properties, which are different from other classifications of FourWheeler, such as luxury, comfort, shape, size, usage etc.

- A car can have further classification such as an open car, small car, big car etc, which will acquire the properties from both Four Wheeler and Car, but will still have some specific properties.

Advantage of Inheritance

Inheritance has a lot of benefits :

- Extensibility
- Reusability
- Provide abstraction
- Eliminates redundant code

Polymorphism

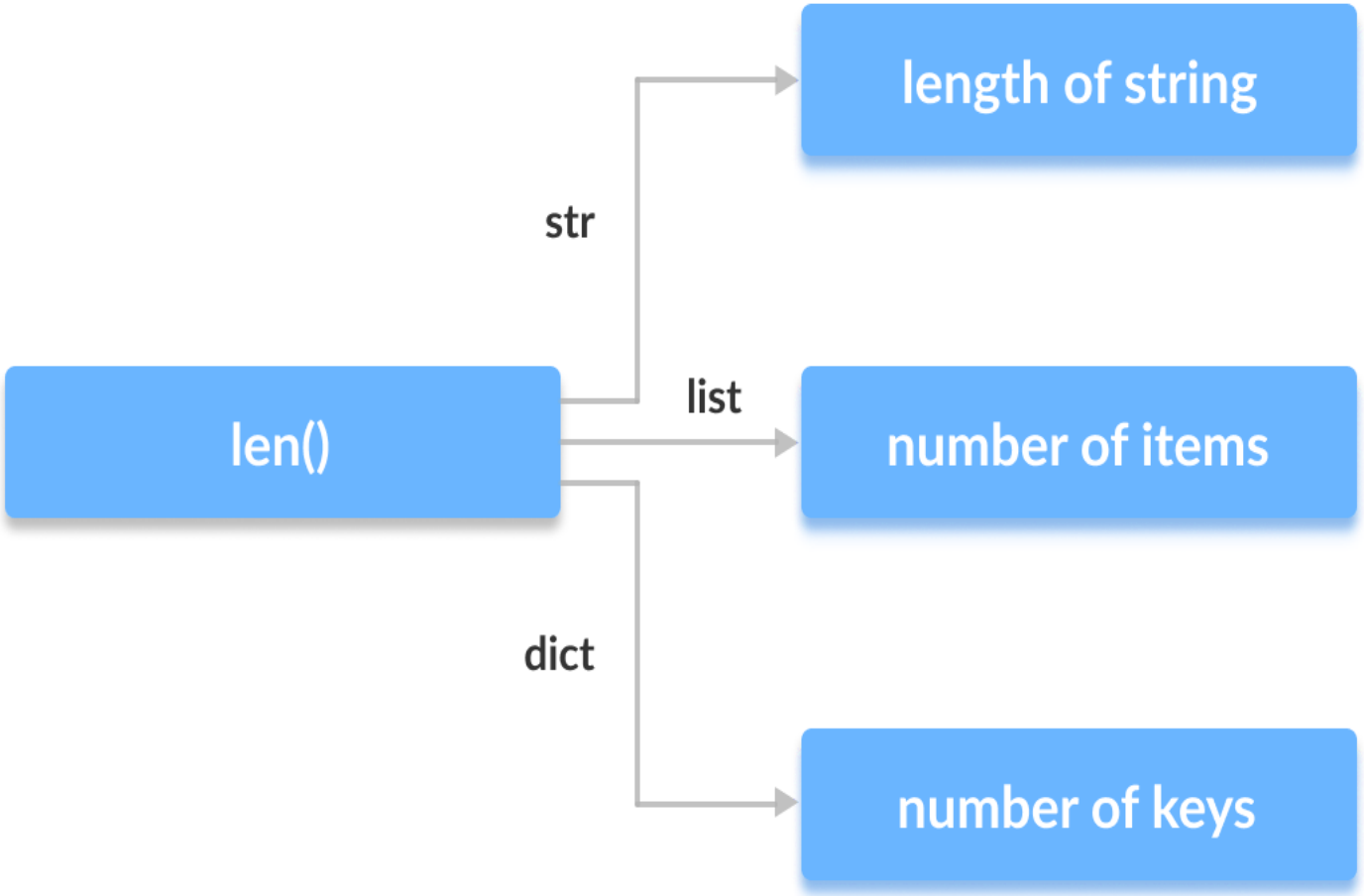
- Polymorphism is a word that came from two greek words, poly means many and morphos means forms.

- If a variable, object or method perform different behavior according to situation, it is called **polymorphism**.

Example of a car

- A car have a gear transmission system. It has four front gears and one backward gear. When the engine is accelerated then depending upon which gear is engaged different amount power and movement is delivered to the car. The action is same applying gear but based on the type of gear the action behaves differently or you can say that it shows many forms (polymorphism means many forms)

Polymorphism



Modeling Concepts

- Object-oriented modeling and design is a way of thinking about problems using models organized around real world concepts.

- The fundamental construct is the object, which combines both data structure and behavior.

Class Diagram

- Class diagram is a static diagram. It represents the static view of an application.
- Class diagram is not only used for visualizing, describing, and documenting different aspects of a system but also for constructing executable code of the software application.
- It is the only diagram that is widely used for construction, and it can be mapped with object-oriented languages.

Following are the purpose of class diagrams given below:

- It analyses and designs a static view of an application.
- It describes the major responsibilities of a system.
- It is a base for component and deployment diagrams.
- It incorporates forward and reverse engineering.

Class Diagram

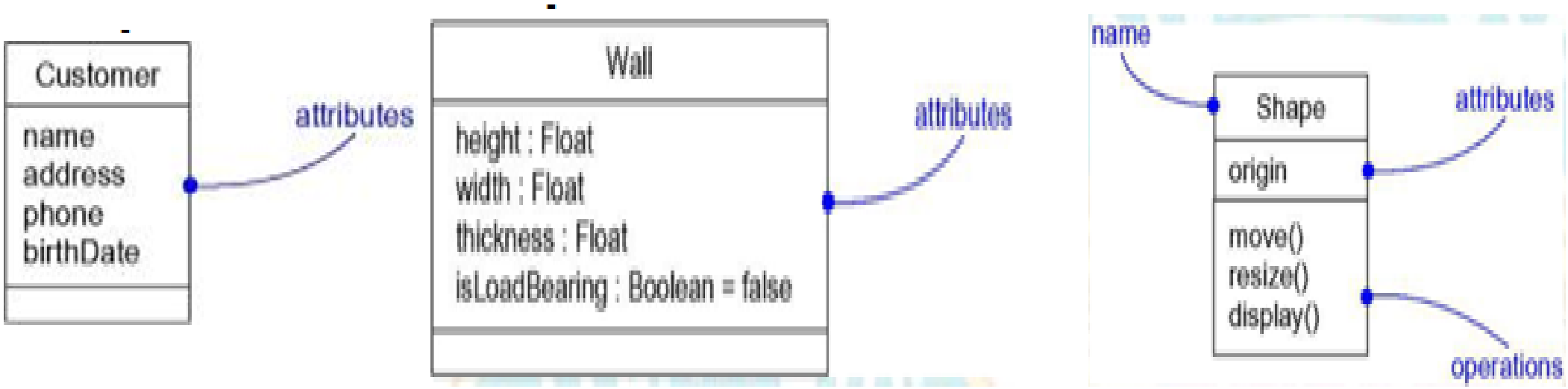
The following points should be remembered while drawing a class diagram −

- The name of the class diagram should be meaningful to describe the aspect of the system.
- Each element and their relationships should be identified in advance.
- Responsibility (attributes and methods) of each class should be clearly identified
- For each class, minimum number of properties should be specified, as unnecessary properties will make the diagram complicated.
- Use notes whenever required to describe some aspect of the diagram. At the end of the drawing it should be understandable to the developer/coder.
- Finally, before making the final version, the diagram should be drawn on plain paper and reworked as many times as possible to make it correct.

Class Diagram

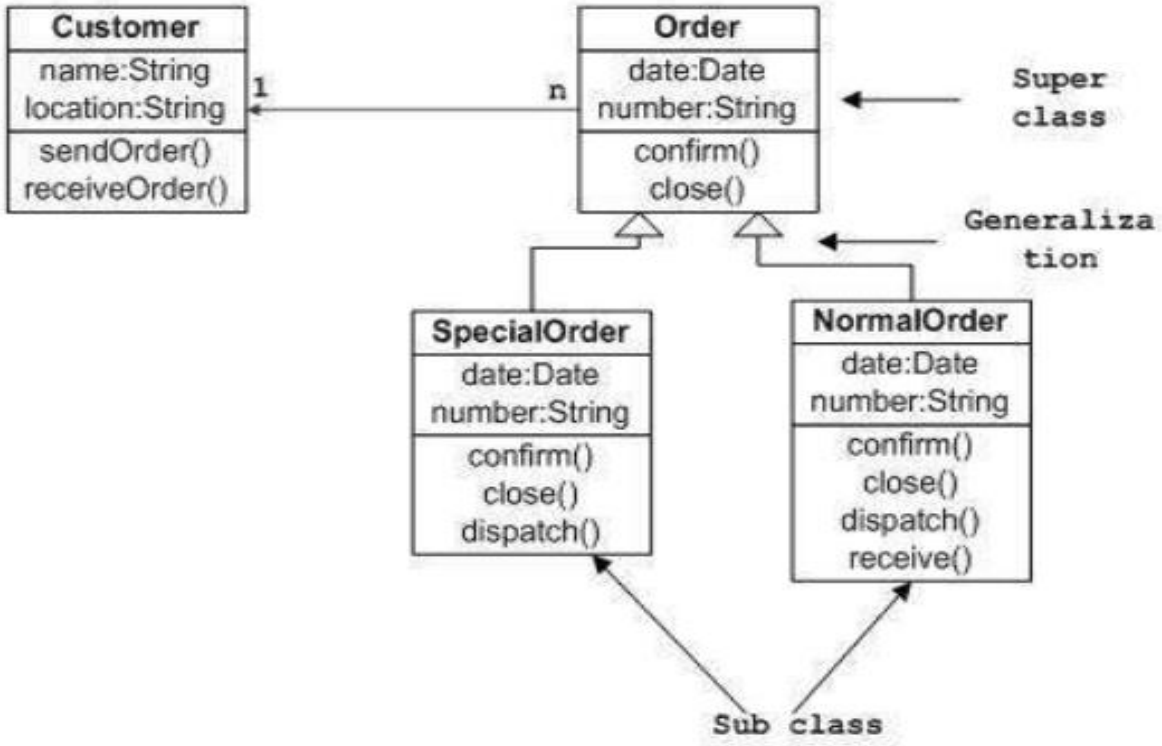
Attributes

- An **attribute** is a named property of a class that describes a range of values that instances of the property may hold.
- A class may have any number of attributes or no attributes at all.
- An attribute represents some property of the thing you are modeling that is shared by all objects of that class.



Class Diagram

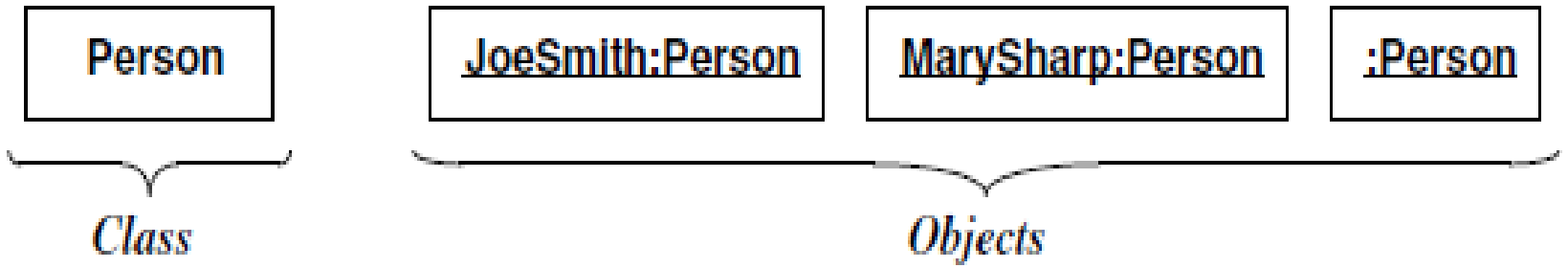
- The following diagram is an example of an Order System of an application. It describes a particular aspect of the entire application.



Object Diagram

Object

- An object is a concept, abstraction, or thing with identity that has meaning for an application.
- Objects often appear as proper nouns or specific references in problem descriptions and discussions with users.
- Some objects have real-world entity like table, tree

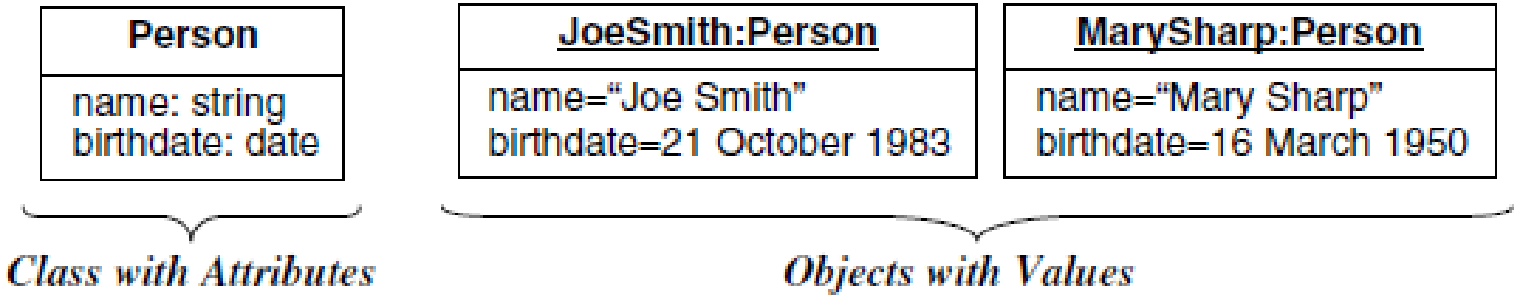


A class and objects. Objects and classes are the focus of class modeling.

Object Diagram

Attribute and Value of an object

- Each attribute is associated a specific value.



Attributes and values. Attributes elaborate classes.

Object Diagram

- Object diagrams are derived from class diagrams so object diagrams are dependent upon class diagrams.
- Object diagrams also represent the static view of a system but this static view is a snapshot of the system at a particular moment.

The purpose of the object diagram can be summarized as

- Forward and reverse engineering.
- Object relationships of a system
- Static view of an interaction.
- Understand object behavior and their relationship from practical perspective

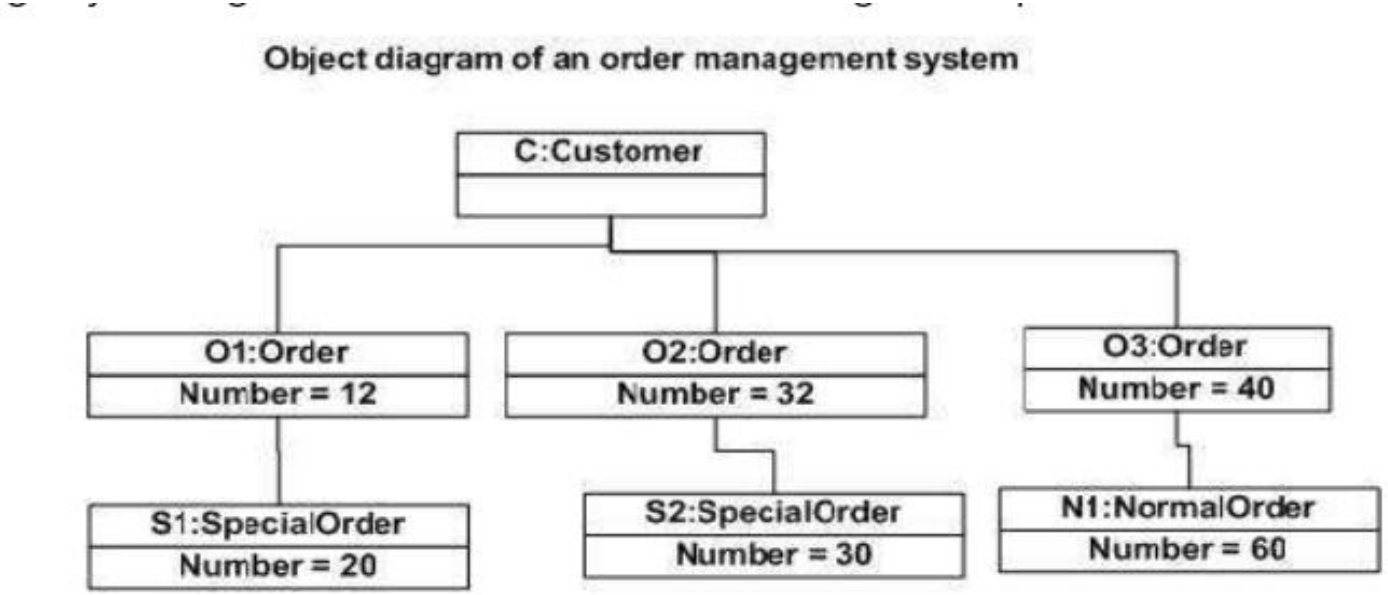
Object Diagram

The following things should be remembered and understood clearly −

- Object diagrams consist of objects.
- The link in object diagram is used to connect objects.
- Objects and links are the two elements used to construct an object diagram.
- The object diagram should have a meaningful name to indicate its purpose.
- The most important elements are to be identified.
- The association among objects should be clarified.
- Values of different elements need to be captured to include in the object diagram.
- Add proper notes at points where more clarity is required.

Object Diagram

- The following diagram is an example of an object diagram. It represents the Order management system



Control statements

- Java compiler executes the code from top to bottom.
- The statements in the code are executed according to the order in which they appear. However, java provides statements that can be used to control the flow of Java code. Such statements are called control flow statements.
- Java provides three types of control flow statements:

I. Decision Making statements

II. Loop statements

III. Jump statements

Decision Making statements

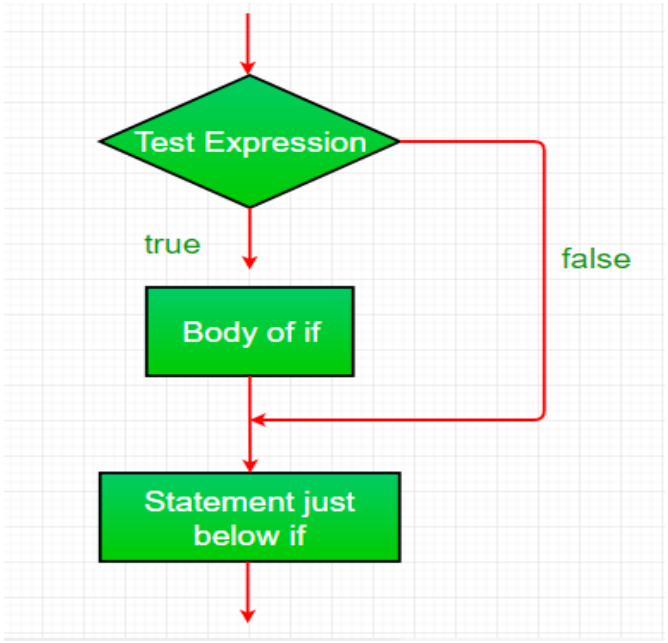
- Decision-making statements decide which statement to execute and when.

- Decision-making statements evaluate the Boolean expression and control the program flow depending upon the result of the condition provided.

Decision Making statements

1.If Statement:

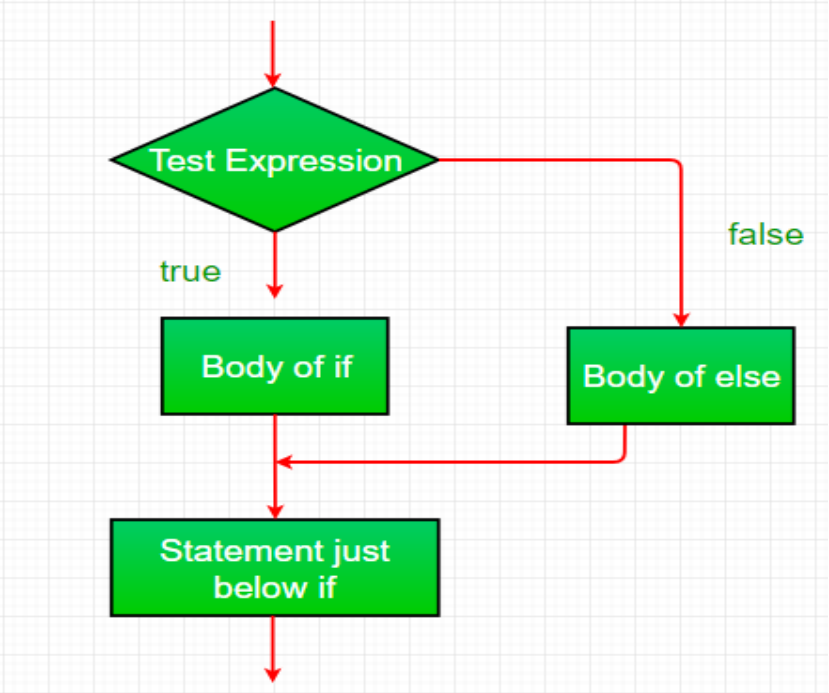
- In Java, the "if" statement is used to evaluate a condition. The control of the program is diverted depending upon the specific condition.



Decision Making statements

If-else:

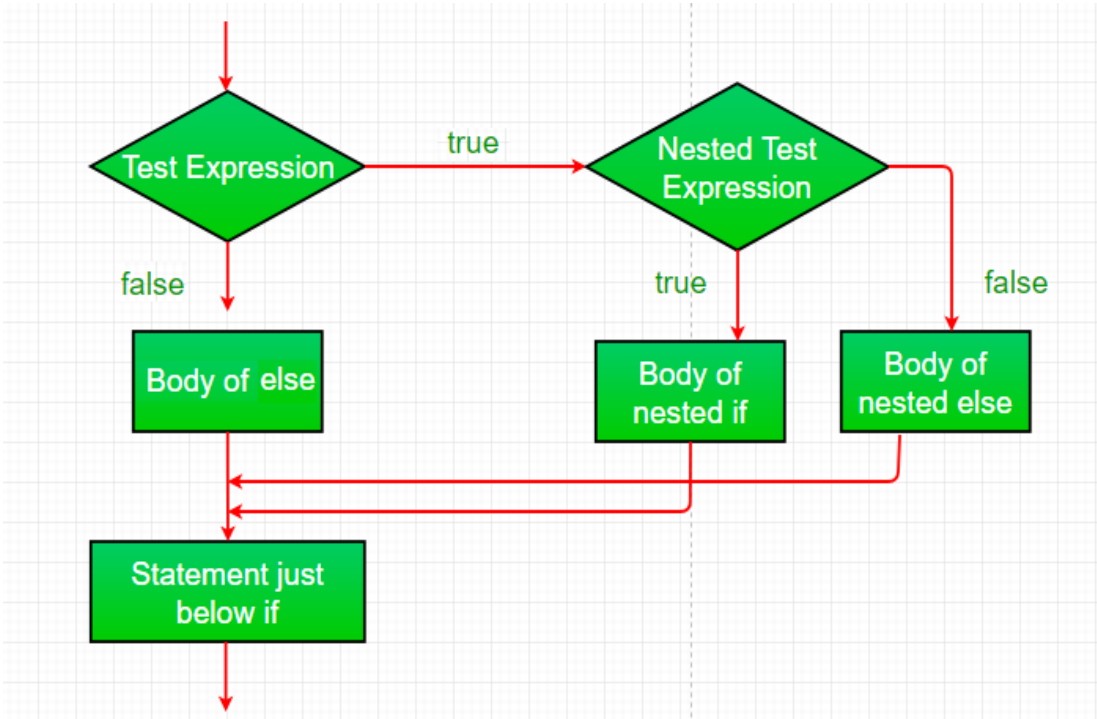
- We can use the else statement with if statement to execute a block of code when the condition is false.



Decision Making statements

Nested-if:

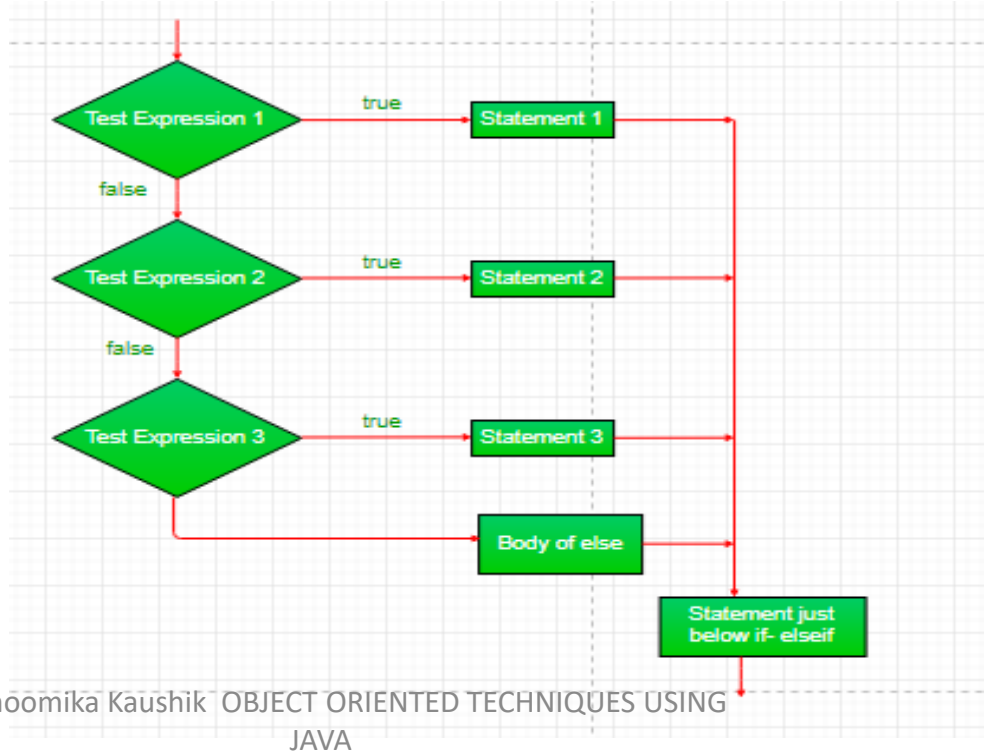
A nested if is an if statement that is the target of another if or else. Nested if statements means an if statement inside an if statement.



Decision Making statements

if-else-if-ladder:

- if statements are executed from the top down.
- As soon as one of the conditions controlling the if is true, the statement associated with that if is executed, and the rest of the ladder is bypassed.

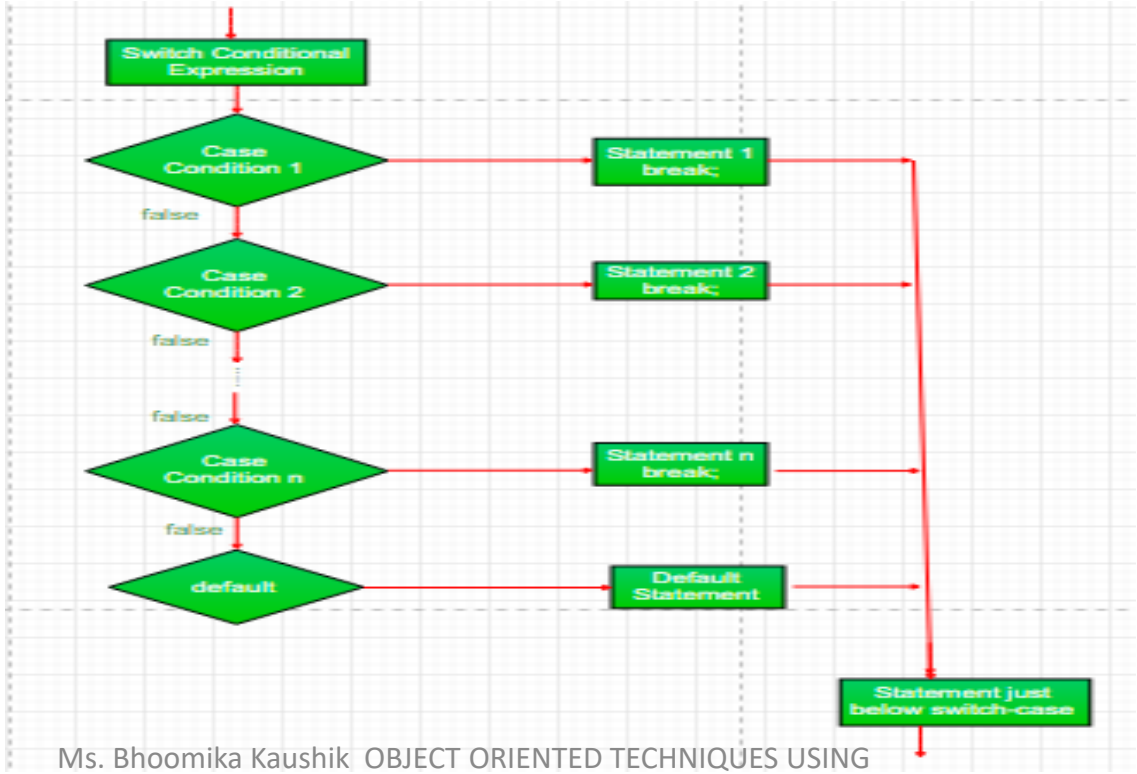


Decision Making statements

switch statement

The switch statement is a multiway branch statement. It provides an easy way to dispatch execution to different parts of code based on the value of the expression.

Syntax:



Looping

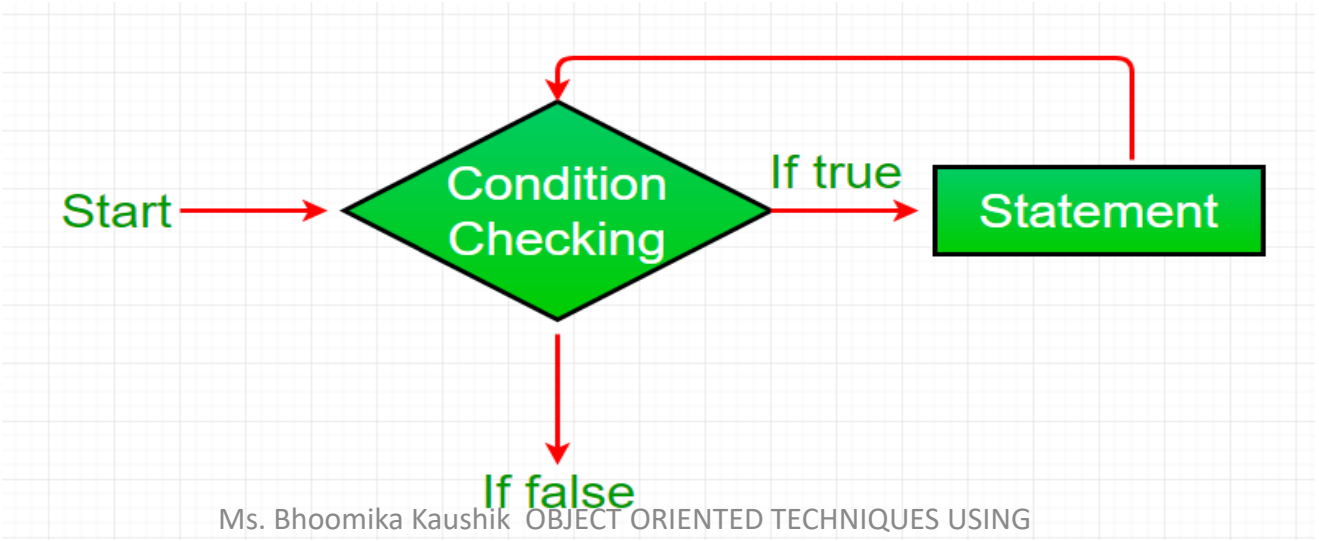
- The Java *for loop* is used to iterate a part of the program several times

- Java provides three ways for executing the loops

Looping

While loop:

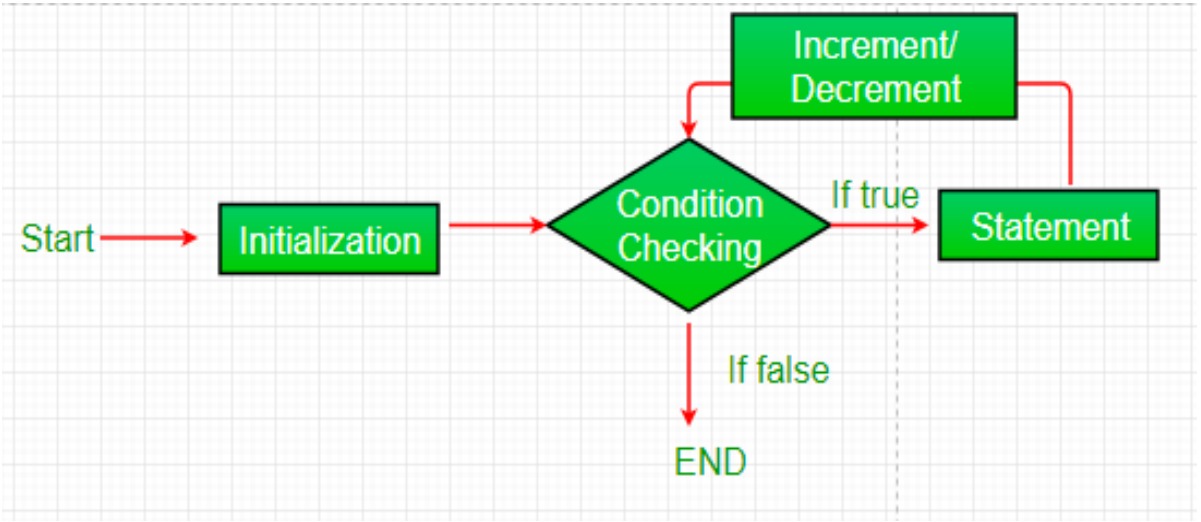
- A while loop is a control flow statement that allows code to be executed repeatedly based on a given Boolean condition. The while loop can be thought of as a repeating if statement.
- While loop starts with the checking of condition. If it evaluated to true, then the loop body statements are executed otherwise first statement following the loop is executed. For this reason it is also called **Entry control loop**



Looping

for loop:

- for loop provides a concise way of writing the loop structure.
- Unlike a while loop, a for statement consumes the initialization, condition and increment/decrement in one line thereby providing a shorter, easy to debug structure of looping.



Looping

- **Initialization condition:** Here, we initialize the variable in use. It marks the start of a for loop. An already declared variable can be used or a variable can be declared, local to loop only.

- **Testing Condition:** It is used for testing the exit condition for a loop. It must return a boolean value. It is also an **Entry Control Loop** as the condition is checked prior to the execution of the loop statements.

- **Statement execution:** Once the condition is evaluated to true, the statements in the loop body are executed.

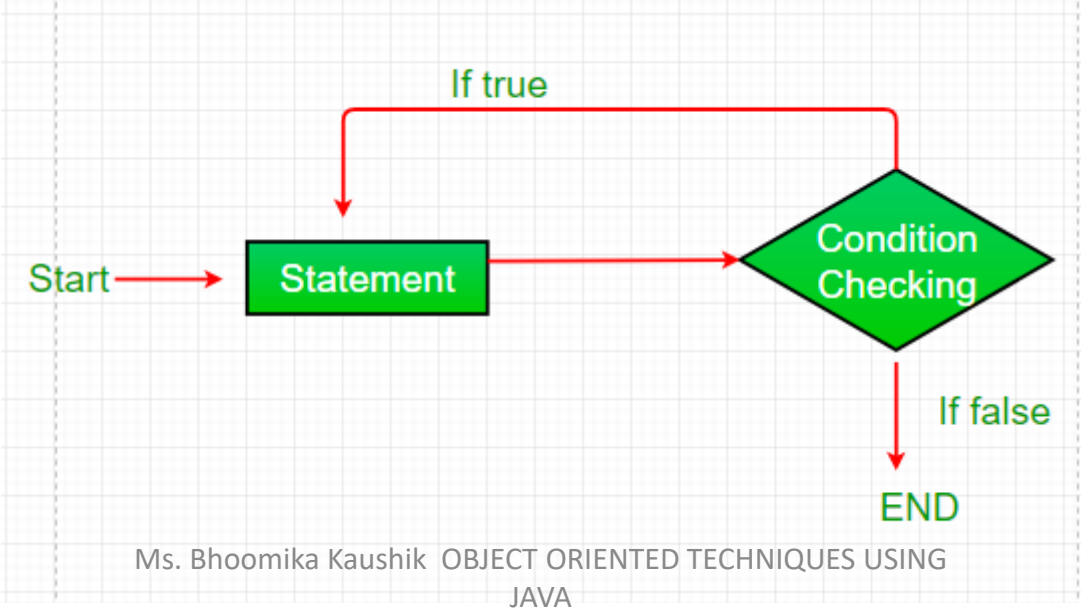
- **Increment/ Decrement:** It is used for updating the variable for next iteration.

- **Loop termination:**When the condition becomes false, the loop terminates marking the end of its life cycle.

Looping

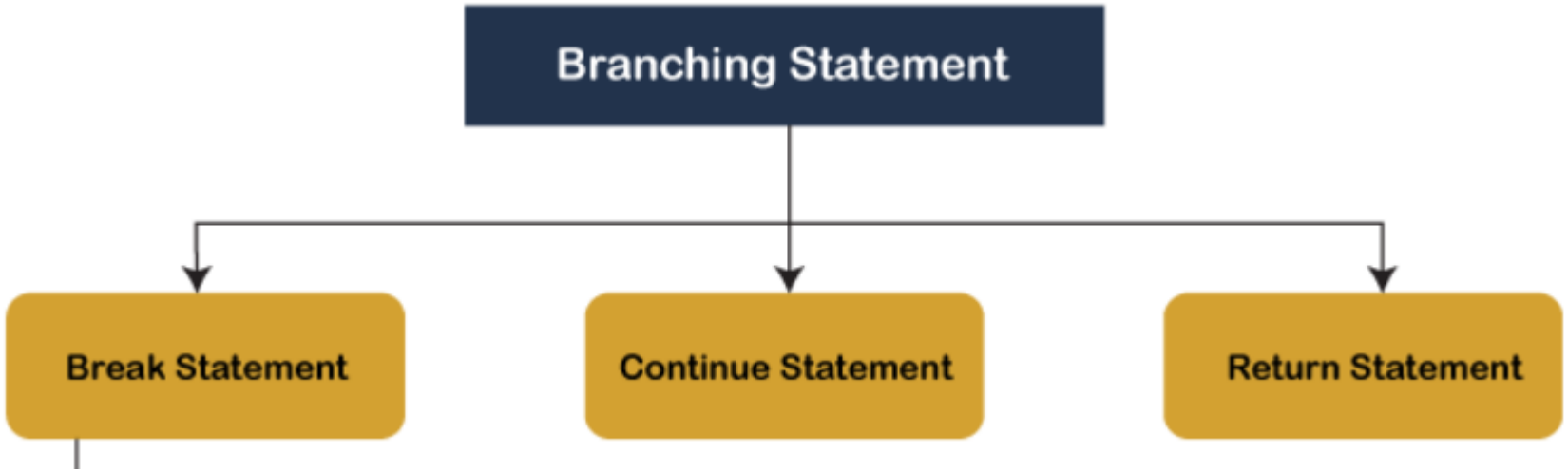
Do-while:

- do while loop is similar to while loop with only difference that it checks for condition after executing the statements, and therefore is an example of **Exit Control Loop.**
- do while loop starts with the execution of the statement(s). There is no checking of any condition for the first time.



Branching statements

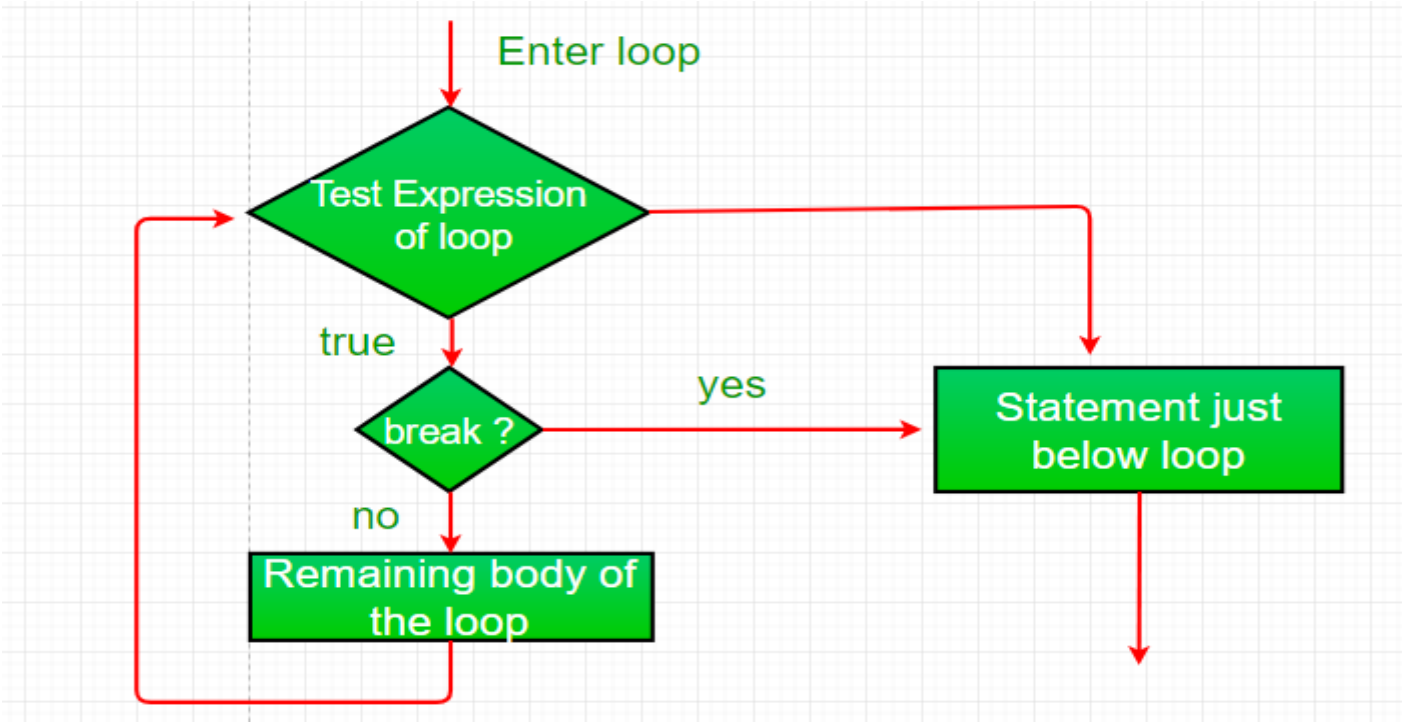
- **Branching statements** are the statements used to jump the flow of execution from one part of a program to another.
- The **branching statements** are mostly used inside the control statements.



Branching statements

The break Statement

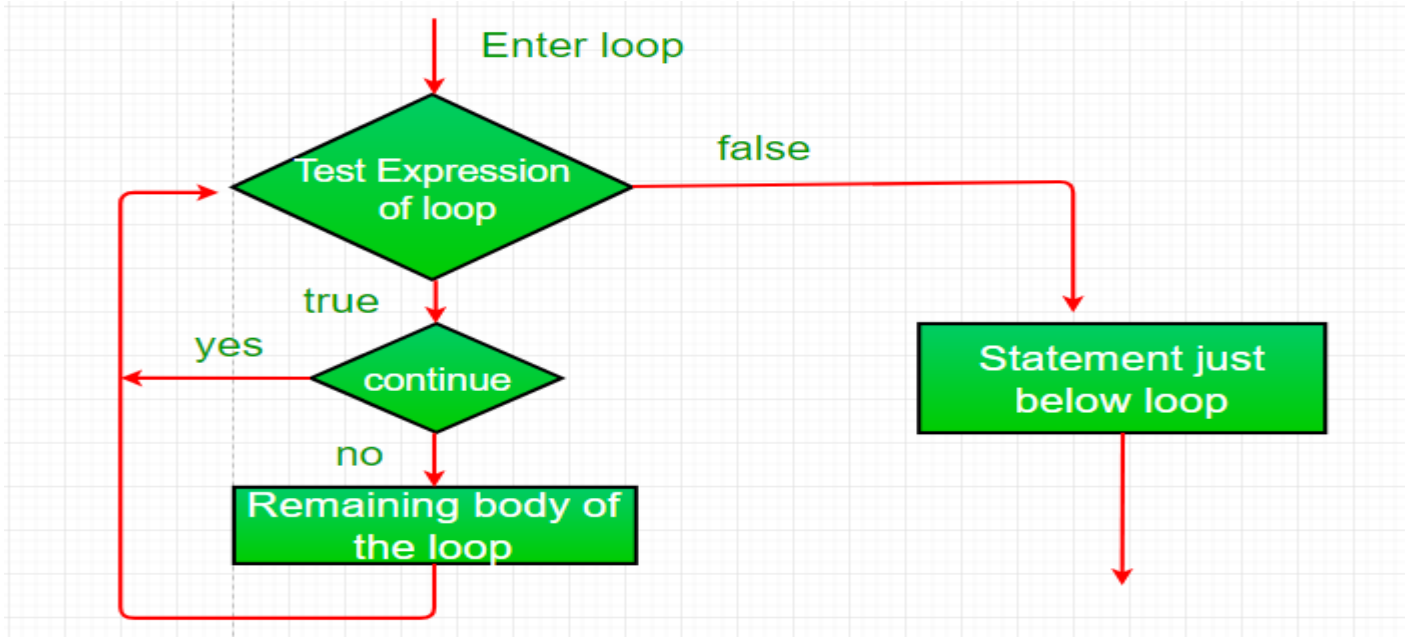
- Using break, we can force immediate termination of a loop, bypassing the conditional expression and any remaining code in the body of the loop.



Branching statements

Continue:

- Sometimes it is useful to force an early iteration of a loop. That is, you might want to continue running the loop but stop processing the remainder of the code in its body for this particular iteration.



Branching statements

Return:

- The return statement is used to explicitly return from a method. That is, it causes a program control to transfer back to the caller of the method.

Argument Passing Mechanism

- Arguments in Java are always **passed-by-value**.

- During method invocation, a copy of each argument, whether its a value or reference, is created in stack memory which is then passed to the method.

- When we pass an object, the reference in stack memory is copied and the new reference is passed to the method.

Command Line Argument

- Command line argument is a parameter supplied to the program when it is invoked.
- Command line argument is an important concept in programming.
- It is mostly used when you need to control your program from outside.
- Command line arguments are passed to the main() method.

Command Line Argument

To pass command line arguments, we typically define main() with two arguments : first argument is the number of command line arguments and second is list of command-line arguments.

```
int main(int argc, char *argv[]) { /* ... */ }
```

- **argc (ARGument Count)** is int and stores number of command-line arguments passed by the user including the name of the program. So if we pass a value to a program, value of argc would be 2 (one for argument and one for program name).The value of argc should be non negative.
- **argv(ARGument Vector)** is array of character pointers listing all the arguments.
- If argc is greater than zero,the array elements from argv[0] to argv[argc-1] will contain pointers to strings.