

# Introduction to JAVA



Image source: [https://upload.wikimedia.org/Java\\_programming\\_language](https://upload.wikimedia.org/Java_programming_language)

Copyright © Edunet Foundation. All rights reserved.

Disclaimer: The content is curated for educational purposes only.

# Overview

- In this presentation, we will be revisiting Java Object Oriented Paradigms, where the fundamental concepts of object-oriented programming are discussed.

# Contents

- Introduction to JAVA Programming
- Java Fundamentals
- Java Control Statements
- Java Arrays
- Java Class and Objects
- Important Topics of OOPS Concept
- Java Exception Handling
- Additional Topics

# Learning Objectives

- Core JAVA (J2SE) Programming and Coding Skills

# Revisiting Java Object Oriented Paradigms

- After 27 years of existence, Java is still doing well.
- Programmers who know it are still in high demand.
- They will continue to be sought after for a long time to come as over 90% of the Fortune 500 companies still rely on Java for their development projects.

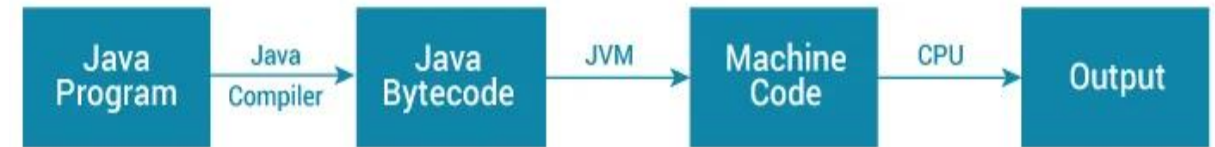


Image source: [https://dev.to/tech\\_sam/let-s-revisit-java-in-2019-3o8c](https://dev.to/tech_sam/let-s-revisit-java-in-2019-3o8c)

# Java JDK, JRE and JVM

What is JVM?

- JVM (Java Virtual Machine) is an abstract machine that enables your computer to run a Java program.



Working of Java Program

Image source: <https://www.programiz.com/java-programming/jvm-jre-jdk>

# Java JDK, JRE and JVM

What is JRE?

- JRE (Java Runtime Environment) is a software package.
- It provides Java class libraries, Java Virtual Machine (JVM), and other components that are required to run Java applications.



Java Runtime Environment

Image source: <https://www.programiz.com/java-programming/jvm-jre-jdk>

# Java JDK, JRE and JVM

What is JDK?

- JDK (Java Development Kit) is a software development kit required to develop applications in Java.



Java Development Kit

Image source: <https://www.programiz.com/java-programming/jvm-jre-jdk>



# Java Variables

- A variable is a location in memory (storage area) to hold data.
- To indicate the storage area, each variable should be given a unique name (identifier).

# Java literals

- Literals are data used for representing fixed values.
- They can be used directly in the code.
- Different types of literals are:
  1. Boolean Literals
  2. Integer Literals
  3. Floating-point Literals
  4. Character Literals
  5. String literals

# Java Data Types

- Data types specify the type of data that can be stored inside variables in Java.
- There are 8 Primitive Data Types:
  1. Boolean
  2. Byte
  3. Short
  4. Int
  5. Long
  6. Double
  7. Float
  8. Char

# Java Operators

- Operators are symbols that perform operations on variables and values.
- Operators in Java can be classified into 6 types:
  1. Arithmetic Operators
  2. Assignment Operators
  3. Relational Operators
  4. Logical Operators
  5. Unary Operators
  6. Bitwise Operators

# Java Basic Input and Output

Java Output: In Java, we simply use

- `System.out.println();` or `System.out.print();` or `System.out.printf();`

Java Input: Java provides different ways to get input from the user. One of them is by using object of Scanner class.

- First, we need to import `java.util.Scanner` package.
- Then, we need to create an object of the Scanner class.
- We can use the object to take input from the user.

# Java Expressions, Statements and Blocks

- Java Expressions: A Java expression consists of variables, operators, literals, and method calls.
- Java Statements: In Java, each statement is a complete unit of execution.
- Expression statements: We can convert an expression into a statement by terminating the expression with a ;.
- Declaration Statements: In Java, declaration statements are used for declaring variables.
- Java Blocks: A block is a group of statements (zero or more) that is enclosed in curly braces { }. A block may not have any statements.

# Java Comments


- Comments are a portion of the program that are completely ignored by Java compilers. They are mainly used to help programmers to understand the code.
- In Java, there are two types of comments:
  1. Single-line Comment: A single-line comment starts and ends in the same line. To write a single-line comment, we can use the `//` symbol.
  2. Multi-line Comment: When we want to write comments in multiple lines, we can use the multi-line comment. To write multi-line comments, we can use the `/*....*/` symbol.

# Java if...else Statement

- We use the if..else statement to run a block of code among more than one alternative.
- The if statement executes a certain section of code if the test expression is evaluated to true.
- However, if the test expression is evaluated too false, it does nothing.
- Statements inside the body of else block are executed if the test expression is evaluated too false.


## Condition is true

```
int number = 5;  
  
if (number > 0) {  
    // code  
}  
else {  
    // code  
}  
// code after if...else
```



## Condition is false

```
int number = 5;  
  
if (number < 0) {  
    // code  
}  
else {  
    // code  
}  
// code after if...else
```



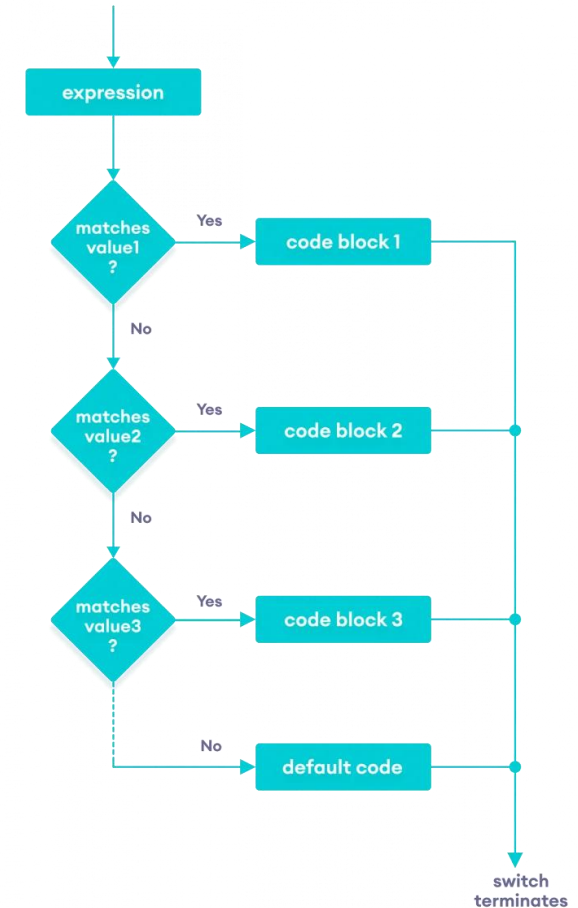
How the if...else statement works?

Image source: <https://www.programiz.com/java-programming/if-else-statement>



# Java switch Statement

- The switch statement allows us to execute a block of code among many alternatives.
- The expression is evaluated once and compared with the values of each case.
- The working of the switch-case statement is like the Java if...else...if ladder.
- However, the syntax of the switch statement is cleaner and much easier to read and write.

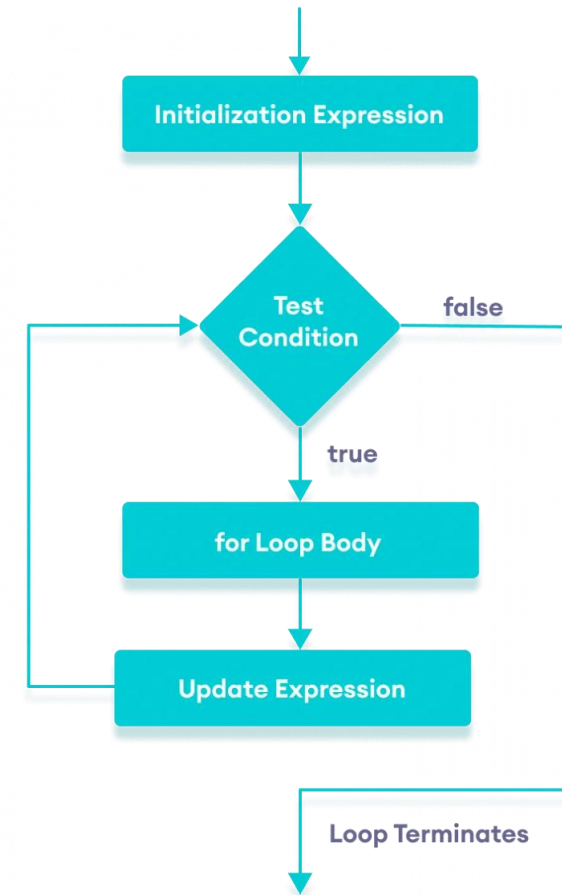


Flowchart of switch Statement

Image source: <https://www.programiz.com/java-programming/switch-statement>

# Java for Loop

- In computer programming, loops are used to repeat a block of code.
- Java for loop is used to run a block of code for a certain number of times.



Flowchart of Java for loop

Image source: <https://www.programiz.com/java-programming/for-loop>

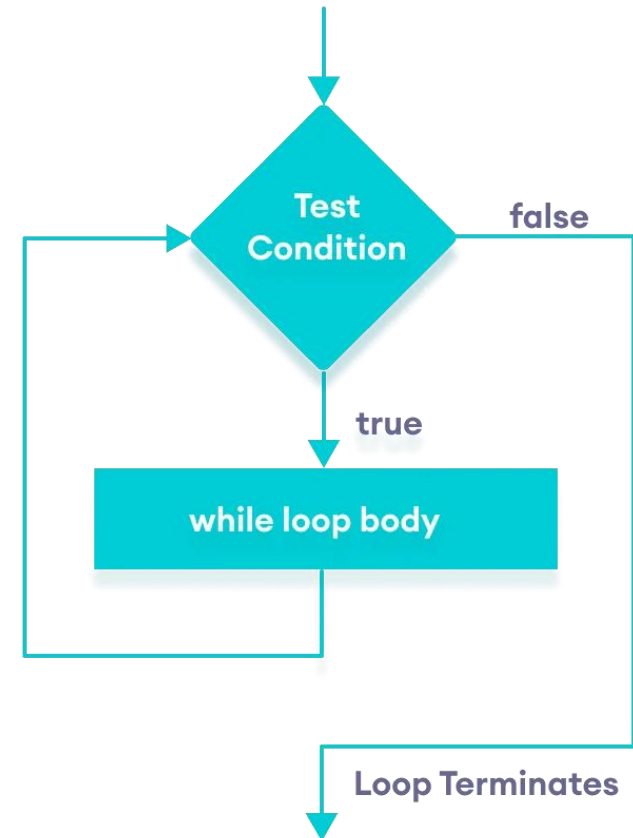
# Java for-each Loop

- In Java, the for-each loop is used to iterate through elements of arrays and collections (like ArrayList).
- It is also known as the enhanced for loop.
- The syntax of the Java for-each loop is:

```
for(dataType item : array) {  
    ...  
}
```

# Java while loop

- Java while loop is used to run a specific code until a certain condition is met.

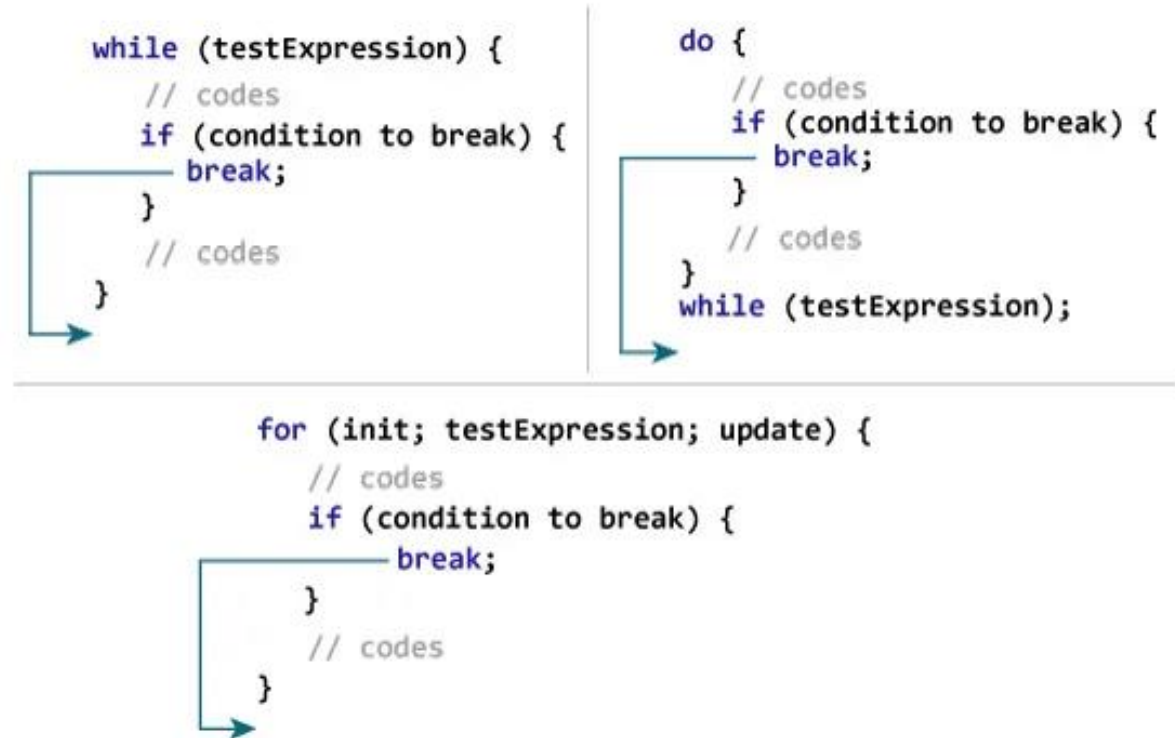


Flowchart of while loop

Image source: <https://www.programiz.com/java-programming/do-while-loop>

# Java break Statement

- While working with loops, it is sometimes desirable to skip some statements inside the loop or terminate the loop immediately without checking the test expression.
- The break statement in Java terminates the loop immediately, and the control of the program moves to the next statement following the loop.
- It is almost always used with decision-making statements (Java if...else Statement).

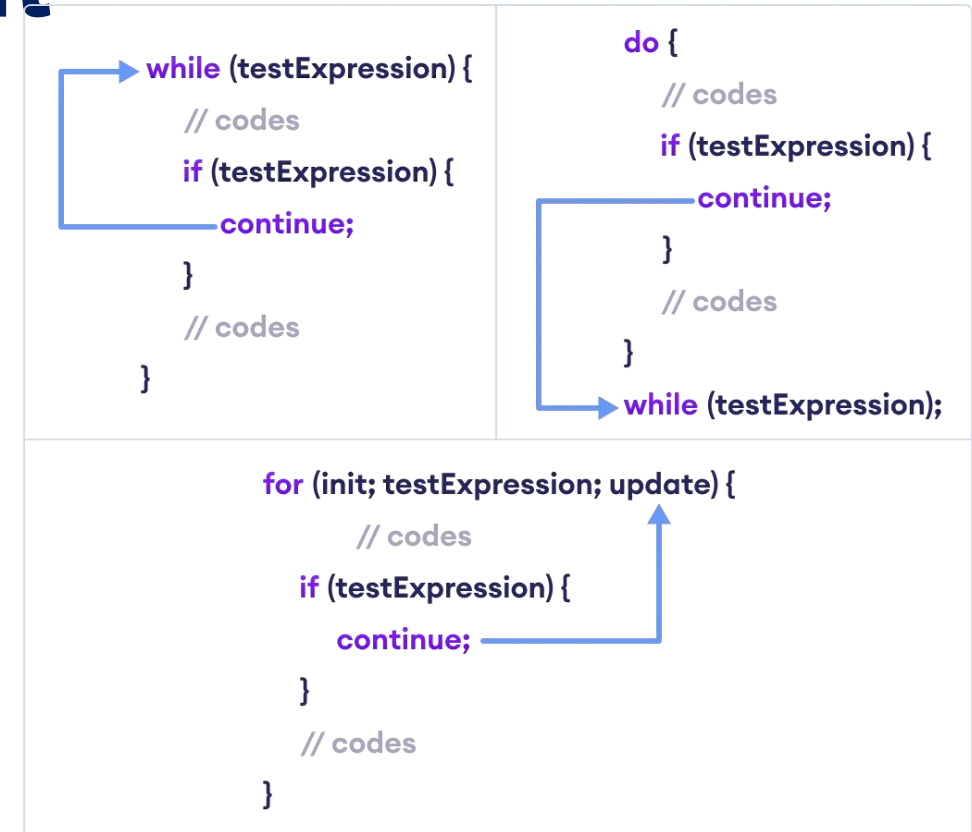


Working of Java break Statement

Image source: <https://www.programiz.com/java-programming/break-statement>

# Java continue Statement

- The continue statement skips the current iteration of a loop (for, while, do...while, etc).
- After the continue statement, the program moves to the end of the loop.
- And test expression is evaluated (update statement is evaluated in case of the for loop).



Working of Java continue Statement

Image source: <https://www.programiz.com/java-programming/continue-statement>

# Java Arrays

- An array is a collection of similar types of data.
- Array indices always start from 0. That is, the first element of an array is at index 0.
- If the size of an array is  $n$ , then the last element of the array will be at index  $n-1$ .

age[0]	age[1]	age[2]	age[3]	age[4]
12	4	5	2	5

Java Arrays initialization

Image source: <https://www.programiz.com/java-programming/arrays>

# Java Multidimensional Arrays

- A multidimensional array is an array of arrays.
- Each element of a multidimensional array is an array itself.
- Each element of the multidimensional array is an array itself.
- unlike C/C++, each row of the multidimensional array in Java can be of different lengths.

	Column 1	Column 2	Column 3	Column 4
Row 1	<code>a[0][0]</code>	<code>a[0][1]</code>	<code>a[0][2]</code>	<code>a[0][3]</code>
Row 2	<code>a[1][0]</code>	<code>a[1][1]</code>	<code>a[1][2]</code>	<code>a[1][3]</code>
Row 3	<code>a[2][0]</code>	<code>a[2][1]</code>	<code>a[2][2]</code>	<code>a[2][3]</code>

2-dimensional Array

Image source: <https://www.programiz.com/java-programming/multidimensional-array>



# Java Class and Objects

- Java Class
  - A class is a blueprint for the object. Before we create an object, we first need to define the class.
- Java Objects
  - An object is called an instance of a class.

# Java Methods

- A method is a block of code that performs a specific task.
- Dividing a complex problem into smaller chunks makes your program easy to understand and reusable.
- In Java, there are two types of methods:
  1. User-defined Methods
  2. Standard Library Methods

# Java Constructors

- What is a Constructor?
- A constructor in Java is like a method that is invoked when an object of the class is created.
- Unlike Java methods, a constructor has the same name as that of the class and does not have any return type.

# Java Constructors

- Types of Constructor
- In Java, constructors can be divided into 3 types:
  1. No-Arg Constructor
  2. Parameterized Constructor
  3. Default Constructor

# Java Strings

- In Java, a string is a sequence of characters.
- We use double quotes to represent a string in Java.
- Strings in Java are not primitive types (like int, char, etc). Instead, all strings are objects of a predefined class named String.
- All string variables are instances of the String class.
- In Java, strings are immutable.

# this Keyword

- In Java, this keyword is used to refer to the current object inside a method or a constructor.
- There are various situations where this keyword is commonly used:
  - Using this for Ambiguity Variable Names
  - Using this in Constructor Overloading
  - Passing this as an Argument

# Java final keyword

- In Java, the final keyword is used to denote constants. It can be used with variables, methods, and classes.
- The final variable cannot be reinitialized with another value
- The final method cannot be overridden
- The final class cannot be extended

# Understanding OOP concepts through core Java

OOP concepts include

- Abstraction,
- Encapsulation,
- Inheritance and
- Polymorphism.



Image source: <https://emergetechnology.org/tag/java/>



# Abstraction

- In Java, abstraction means simple things like objects, classes and variables represent more complex underlying code and data.
- It lets you avoid repeating the same work multiple times.

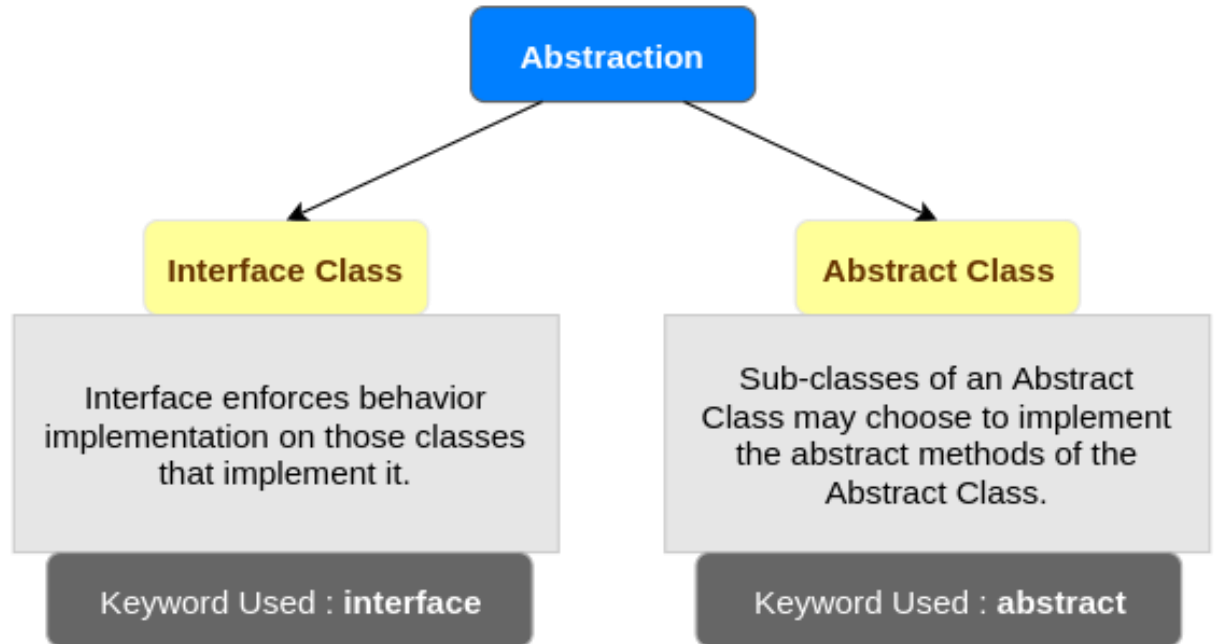


Image source: <https://www.tutorialkart.com/java/abstraction-in-java>

# How Abstraction Works?

- Abstraction lets programmers create useful and reusable tools.
- For example, a programmer can create several different types of **objects**, which can be variables, functions or data structures. Programmers can also create different **classes** of objects as ways to define the objects.

## Example for Java Abstraction



	Owner
	<ul style="list-style-type: none"> <li>• Car Description</li> <li>• Service History</li> <li>• Petrol Mileage History</li> </ul>

	Registration
	<ul style="list-style-type: none"> <li>• Vehicle Identification Number</li> <li>• License plate</li> <li>• Current Owner</li> <li>• Tax due, date</li> </ul>

	Garage
	<ul style="list-style-type: none"> <li>• License plate</li> <li>• Work Description</li> <li>• Billing Info</li> <li>• Owner</li> </ul>

Image source: <https://techvidvan.com/tutorials/abstraction-in-java>

# Java Abstract Class

- The abstract class in Java cannot be instantiated (we cannot create objects of abstract classes).
- We use the abstract keyword to declare an abstract class.
- An abstract class can have both the regular methods and abstract methods.
- If a class contains an abstract method, then the class should be declared abstract.
- An abstract class can have constructors like the regular class.

# Java Interface

- An interface is a fully abstract class. It includes a group of abstract methods .
- We use the interface keyword to create an interface in Java.
- Like abstract classes, we cannot create objects of interfaces.
- To use an interface, other classes must implement it.
- We use the implements keyword to implement an interface.
- In Java, a class can also implement multiple interfaces.
- Similar to classes, interfaces can extend other interfaces.

# Encapsulation

- Encapsulation is a protective barrier that keeps the data and code safe within the class itself.
- We can then reuse objects like code components or variables without allowing open access to the data system-wide.

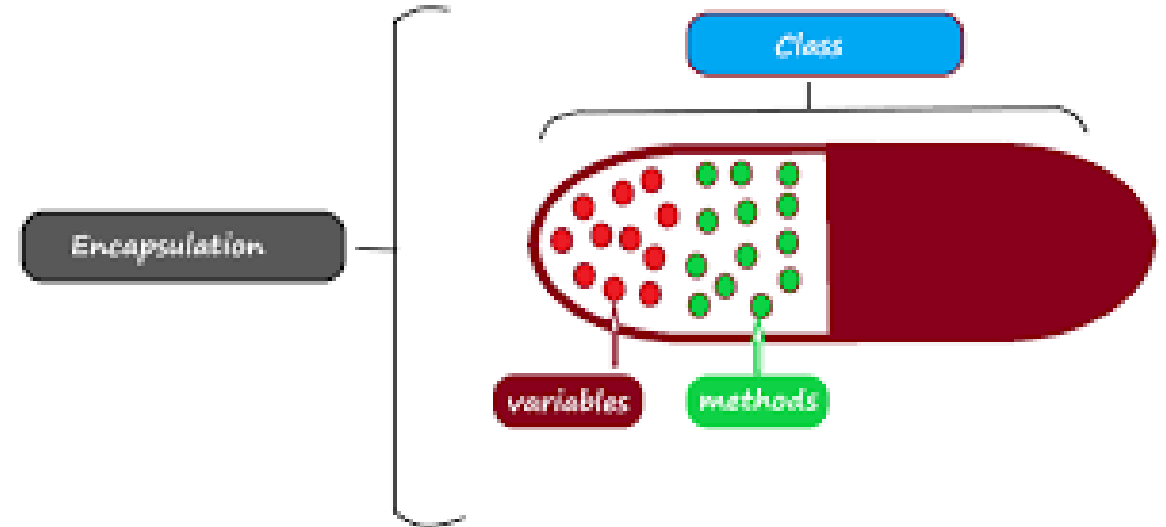


Image source: <https://techblogstation.com/java/oops-concepts-in-java>

# How Encapsulation Works?

- Encapsulation lets us reuse functionality without compromising security.
- It's a powerful, time-saving OOP concept in Java.
- For example, we may create a piece of code that calls specific data from a database.

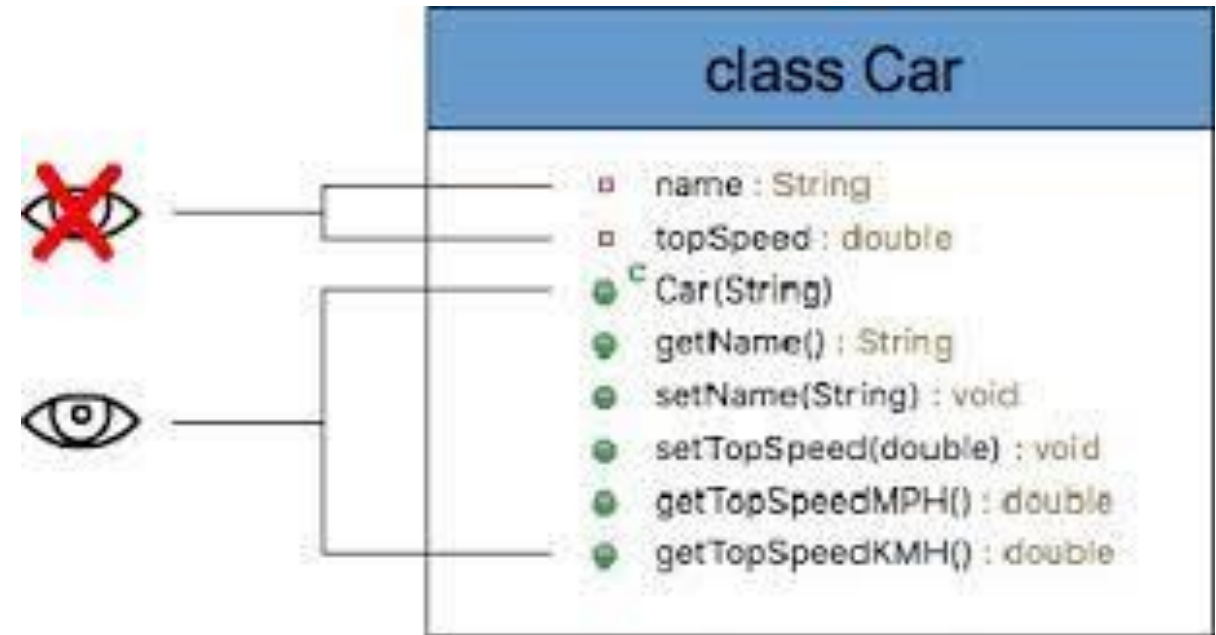
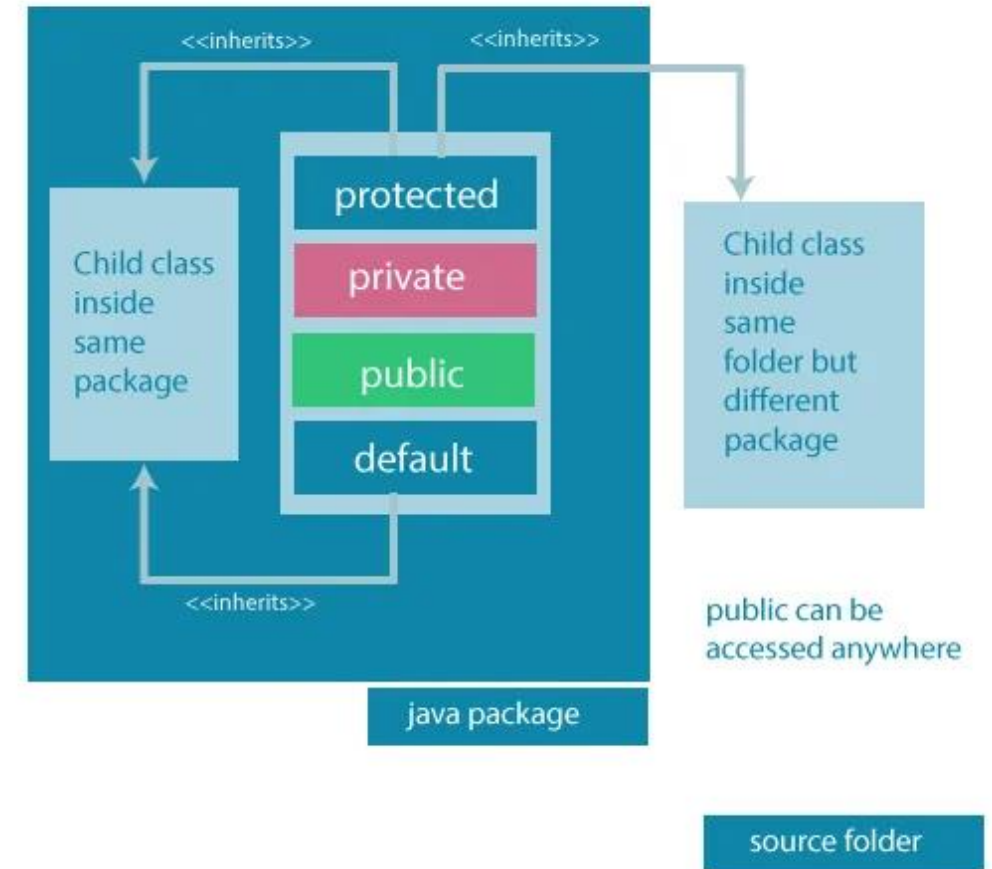


Image source: <https://javatutorial.net/java-encapsulation-example>

# Java Access Modifiers

- In Java, access modifiers are used to set the accessibility (visibility) of classes, interfaces, variables, methods, constructors, data members, and the setter methods.
- There are four access modifiers keywords in Java:
  1. Default
  2. Private
  3. Protected
  4. Public



Accessibility of all access modifiers in java

Image source: <https://www.programiz.com/java-programming/access-modifiers>

# Inheritance

- Inheritance lets programmers create new classes that share some of the attributes of existing classes.
- Using Inheritance lets us build on previous work without reinventing the wheel.

## Inheritance in Java

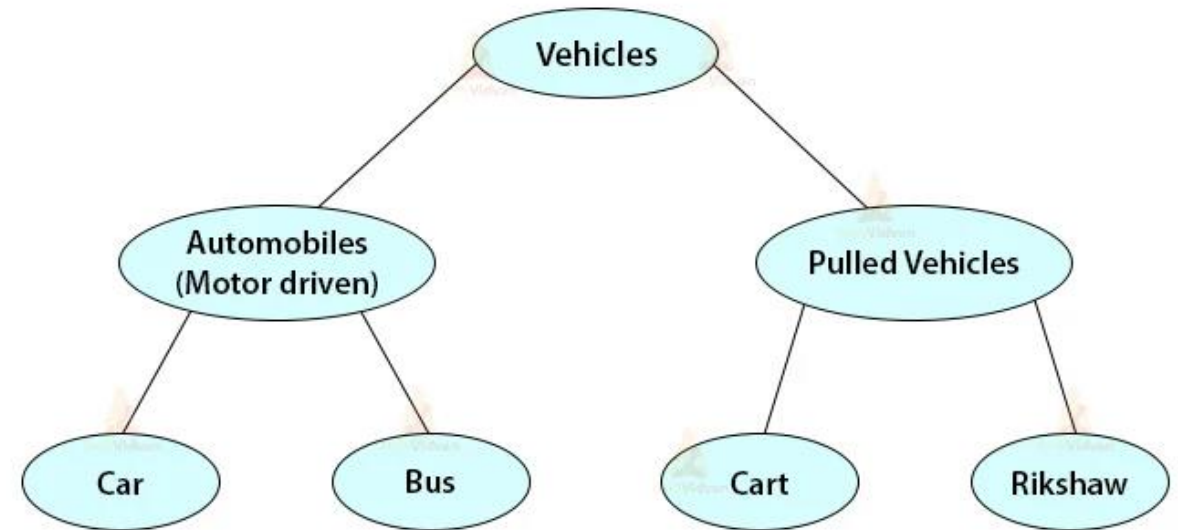


Image source: <https://techvidvan.com/tutorials/java-inheritance>



# How Inheritance Works?

- Inheritance lets a new class adopt the properties of another.
- Inheriting class is called a subclass or a child class.
- The original class is often called the parent.
- Keyword **extends** is used to define a new class that inherits properties from an old class.

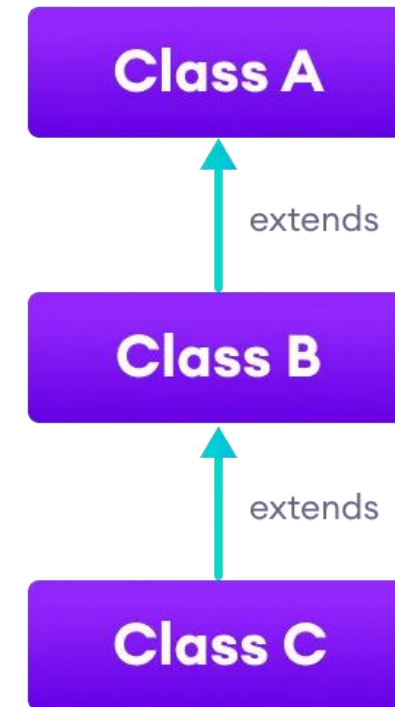


Image source: <https://www.programiz.com/java-programming/inheritance>

# Types of Inheritance

There are **five** types of inheritance.

- Single
- Multiple
- Multilevel
- Hierarchical
- Hybrid

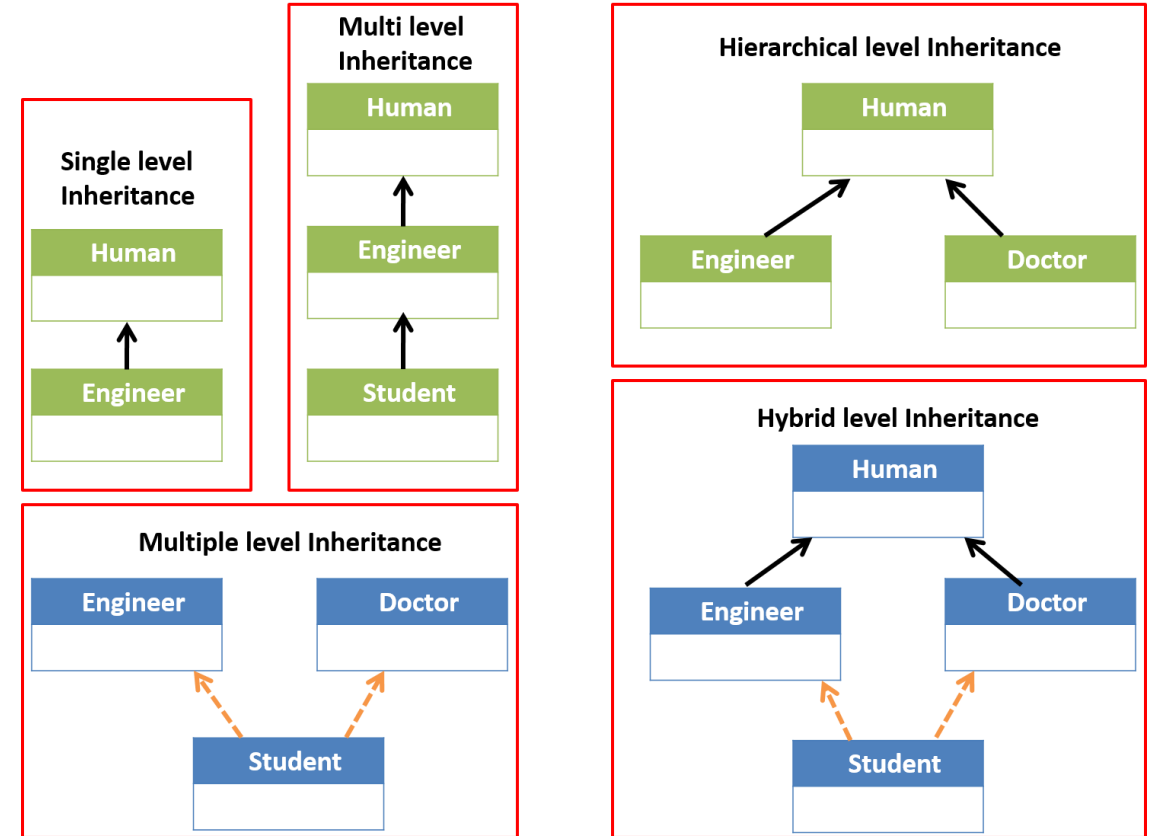


Image source: <https://bytesofgigabytes.com/java/inheritance-in-java>

# Java Inheritance

- Inheritance is one of the key features of OOP that allows us to create a new class from an existing class. Java does not support multiple inheritance
- The extends keyword is used to perform inheritance in Java.
- There are five types of inheritance:
  1. Single inheritance
  2. Multilevel inheritance
  3. Hierarchical inheritance
  4. Multiple inheritance
  5. Hybrid inheritance

# Java super

- The super keyword in Java is used in subclasses to access superclass members (attributes, constructors and methods).
- Uses of super keyword:
  1. To call methods of the superclass that is overridden in the subclass.
  2. To access attributes (fields) of the superclass if both superclass and subclass have attributes with the same name.
  3. To explicitly call superclass no-arg (default) or parameterized constructor from the subclass constructor.

# Polymorphism

- Allows programmers to use the same Java to mean different things in contexts.
- One form of polymorphism is **overloading**.
- The other form is method **overriding**.

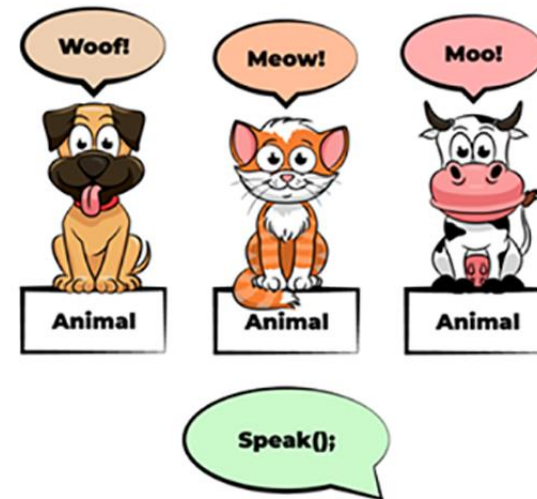


Image source: <https://codegym.cc/groups/posts/polymorphism-in-java>

# How Polymorphism Works?

- We might create a class called “**horse**” by extending the “**animal**” class.
- That class might also implement the “**professional racing**” class.
- The “**horse**” class is “**polymorphic**,” since it inherits attributes of both the “**animal**” and “**professional racing**” class.



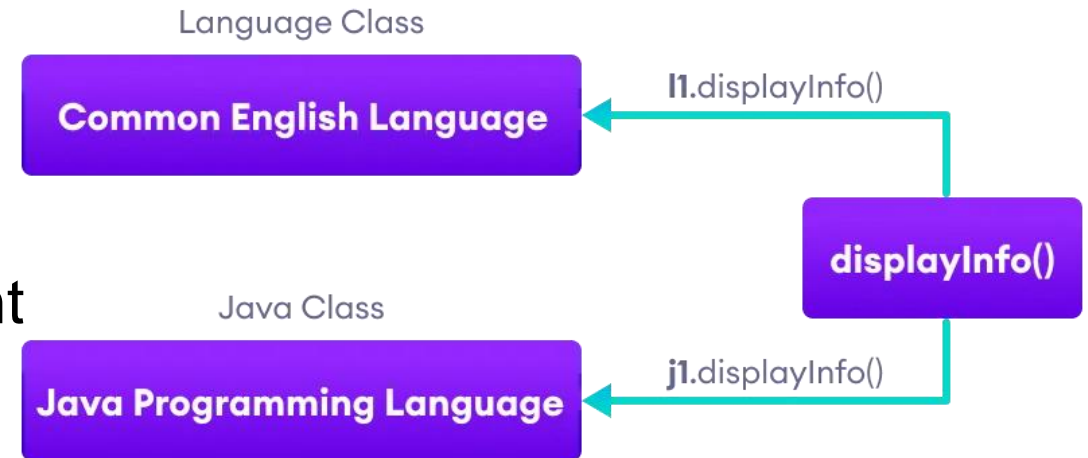
- > A Boy behave like a Student in a School
- > A Boy behave like a Customer in Market or Shopping Mall
- > A Boy behave like a Passenger in a Bus
- > A Boy behave like a Son in Home

Image source: <https://www.javaguides.net/2018/08/polymorphism-in-java-with-example.html>

# Java Polymorphism

- Polymorphism is an important concept of object-oriented programming.
- It simply means more than one form.
- Polymorphism allows us to create consistent code.
- We can achieve polymorphism in Java using the following ways:

1. Method Overriding
2. Method Overloading



Working of Java polymorphism

Image source: <https://www.programiz.com/java-programming/polymorphism>

# Method Overloading

- A single method may perform different functions depending on the context in which it's called.
- This means a single method name might work in different ways depending on what arguments are passed to it.

## Method Overloading in Java

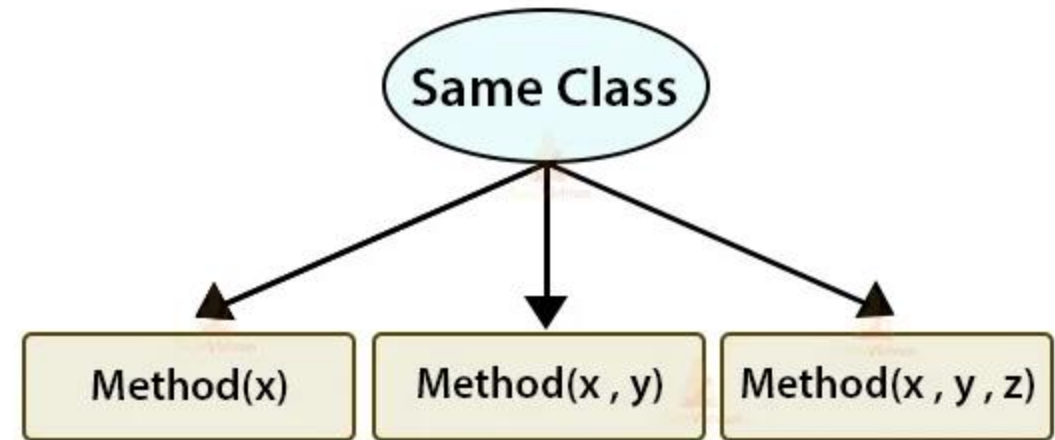


Image source: <https://techvidvan.com/tutorials/method-overloading-and-overriding>



# Method Overriding

- The child class can override a method of its parent class.
- That allows a programmer to use one method in different ways depending on whether it's invoked by an object of the parent class or an object of the child class.

## Method Overriding in Java

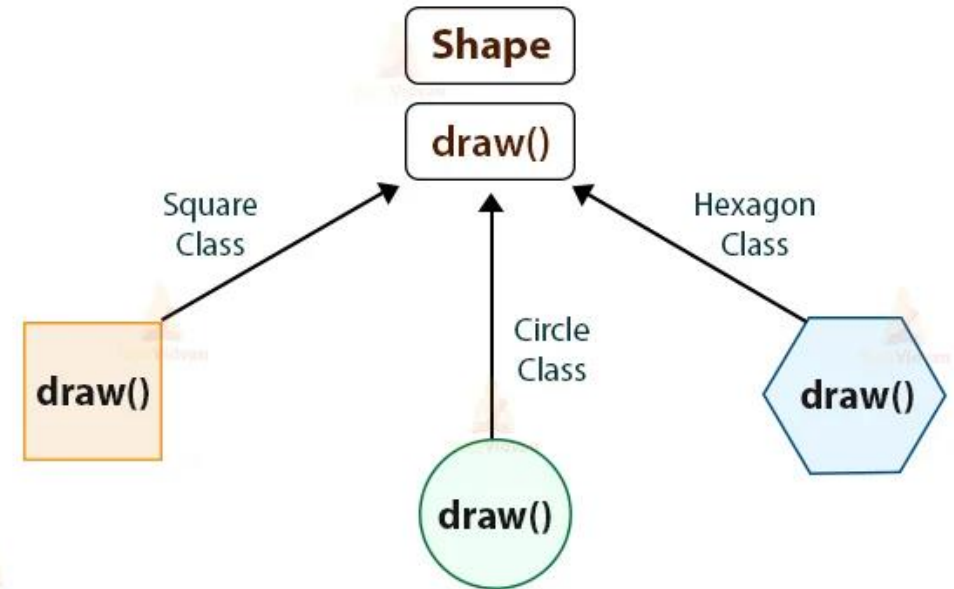


Image source: <https://techvidvan.com/tutorials/method-overloading-and-overriding>

# Java Method Overriding

- If the same method is defined in both the superclass and the subclass, then the method of the subclass class overrides the method of the superclass.
- This is known as method overriding.

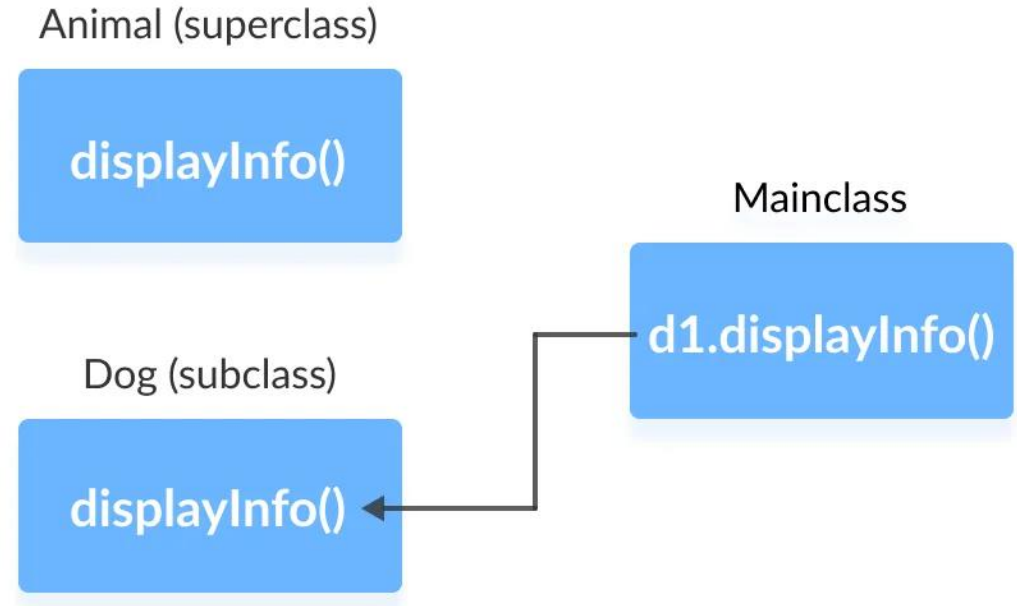


Image source: <https://www.programiz.com/java-programming/method-overriding>

# Java Method Overriding

## Java Overriding Rules:

- Both the superclass and the subclass must have the same method name, the same return type and the same parameter list.
- We cannot override the method declared as final and static.
- We should always override abstract methods of the superclass

# Java Exceptions

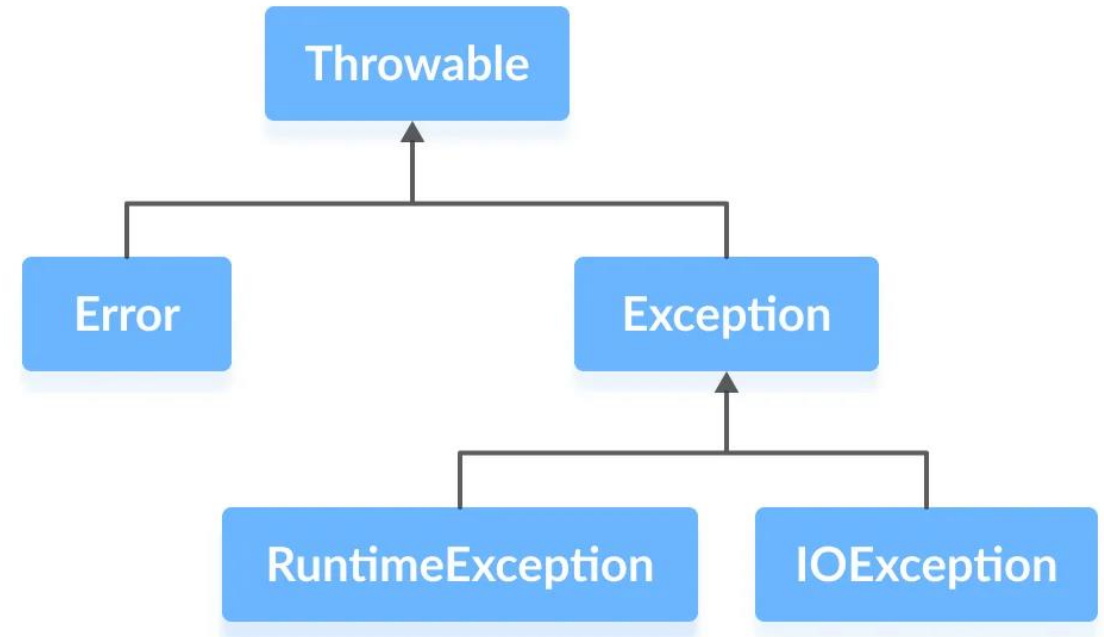
- An exception is an unexpected event that occurs during program execution.

## Errors

- Errors represent irrecoverable conditions
- Errors are usually beyond the control of the programmer, and we should not try to handle errors.

## Exceptions

- Exceptions can be caught and handled by the program.



Java Exception Hierarchy

Image source: <https://www.programiz.com/java-programming/exceptions>

# Java try...catch

- The try...catch block in Java is used to handle exceptions and prevents the abnormal termination of the program.
- The try block includes the code that might generate an exception.
- The catch block includes the code that is executed when there occurs an exception inside the try block.
- In Java, we can use a try block without a catch block. However, we cannot use a catch block without a try block.

# Multiple Catch blocks

- For each try block, there can be zero or more catch blocks.
- Multiple catch blocks allow us to handle each exception differently.
- The argument type of each catch block indicates the type of exception that can be handled by it.

# finally block

- The finally block is always executed whether there is an exception inside the try block or not.
- The code inside the finally block is executed irrespective of the exception.
- It is a good practice to use finally block to include important cleanup code like closing a file or connection.

# Java throw and throws

Java throws keyword

- The throws keyword in the method declaration to declare the type of exceptions that might occur within it.

Java throw keyword

- The throw keyword is used to explicitly throw a single exception.



# Java Scanner Class

- The Scanner class of the java.util package is used to read input data from different sources like input streams, users, files, etc.
- The Scanner class provides various methods that allow us to read inputs of different types.
- Examples are nextInt(), nextFloat(), nextBoolean(), nextLine(), etc.

# Java Wrapper Class

- The wrapper classes in Java are used to convert primitive types (int, char, float, etc) into corresponding objects.
- Each of the 8 primitive types has corresponding wrapper classes.
- In Java, sometimes we might need to use objects instead of primitive data types. For example, while working with collections. In such cases, wrapper classes help us to use primitive data types as objects.
- We can store the null value in wrapper objects.

# Java Type Casting

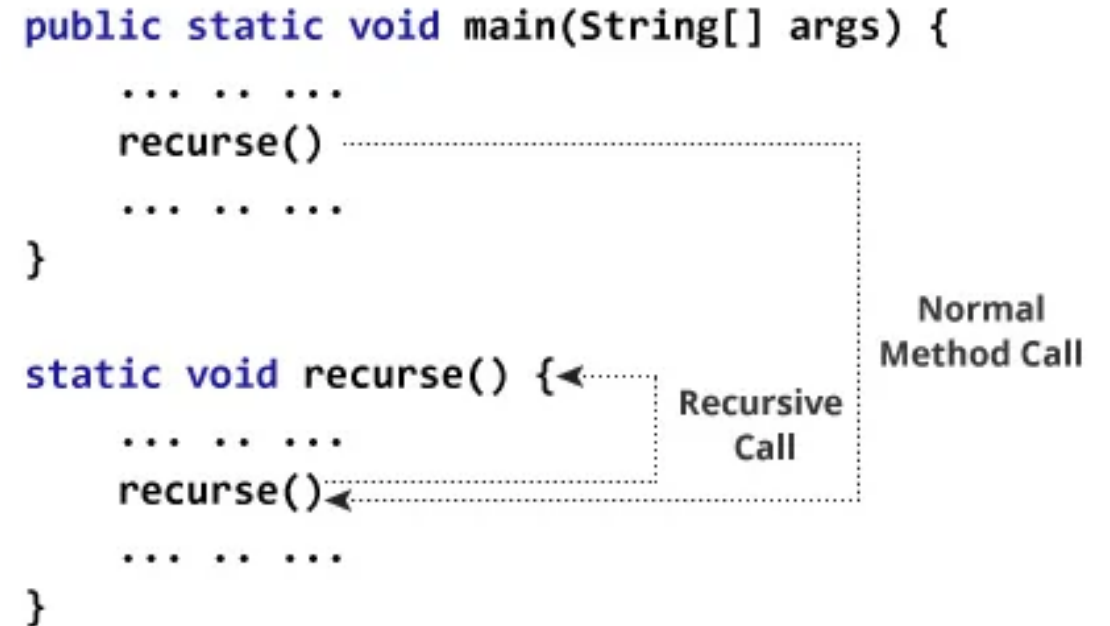
- The process of converting the value of one data type (int, float, double, etc.) to another data type is known as typecasting.
- In Widening Type Casting, Java automatically converts one data type to another data type.
- The lower data type (having smaller size) is converted into the higher data type (having larger size). Hence there is no loss in data.
- We manually convert one data type into another using the parenthesis.
- This is also known as Explicit Type Casting.

# Java Command-Line Arguments

- The command-line arguments in Java allow us to pass arguments during the execution of the program.
- `public static void main(String[] args) {...}`
- The String array stores all the arguments passed through the command line.

# Java Recursion

- A method that calls itself is known as a recursive method. And this process is known as recursion.
- if...else statement is used to terminate the recursive call inside the method.
- A recursive solution is much simpler and takes less time to write, debug and maintain.
- Recursion generally uses more memory and is generally slow.



Working of Java Recursion

Image source: <https://www.programiz.com/java-programming/recursion>

—thank you—