UNIT-01: Principles of Object-Oriented Programming with Introduction to JAVA

- The Traditional approach, drawback of procedure-oriented languages,
- The three basic constructs of OOPS including abstraction and encapsulation.
- Comparison of various object-oriented languages, Need of java,
- The creation of java, Basic differences of java and C++,
- byte code, difference between JDK, JRE, JVM,
- java applets and applications, java buzzword,
- three basic constructs of oops applicable to java.

Questions to be discussed:

1. Explain the basic concept of object oriented programming.
2. State four features of java.
3. Differentiate procedure-oriented languages and object-oriented languages.
4. What do you mean by byte code?
5. Explain basic differences of java and C++.
6. Write the difference between JDK, JRE and JVM.

CTH EDUCATION Name of Faith Education

Introduction to Java:

- Java is an object-oriented, class-based, secured and general-purpose computer programming language.
- It was developed by Sun Microsystems in the year 1995.
- James Gosling is known as the father of Java.
- Before Java, its name was Oak.
- Java runs on a variety of platforms, such as Windows, Mac OS, and UNIX OS.

An Overview/History of Java:

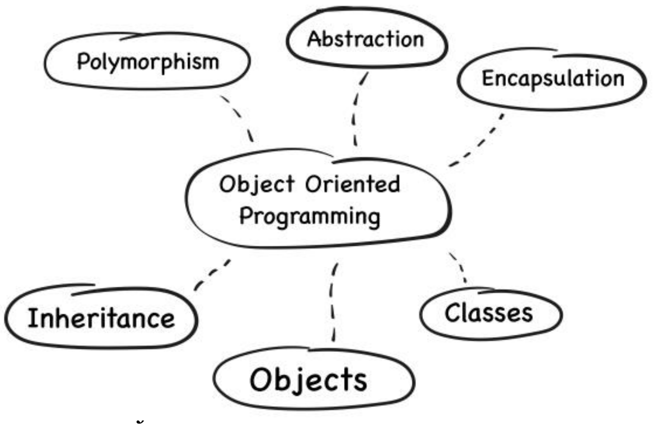
- The history of Java starts with the Green Team. Java team members (also known as Green Team), initiated this project to develop a language for digital devices such as set-top boxes, televisions, etc.
- It was suited for internet programming.
- Currently, Java is used in internet programming, mobile devices, games, e-business solutions, etc.
- JDK 1.0 released in (January 23, 1996).
- Now Java is being used in Windows applications, Web applications, enterprise applications, mobile applications etc.

Difference between C and JAVA:

C	JAVA
C is a Procedural Programming Language.	Java is Object-Oriented language.
C was developed by Dennis M. Ritchie between 1969 and 1973.	Java was developed by James Gosling in 1995.
It is a compiled language.	It is an interpreted language.
It does not follow OOPs concepts.	It follows OOPs concepts.
The file is saved with the extension **.c**.	The file is saved with the extension **.java**.
It is not secure**.**	It is fully secured language.
It translates the code into machine language so that the machine can understand the code.	It translates the code into a bytecode that is executed by the JVM.
It does not support inheritance that is useful for code reusability.	It supports inheritance that provides code reusability.
It generates .exe file.	It generates .class file.
It directly executes the code.	It executes code with the help of JVM.

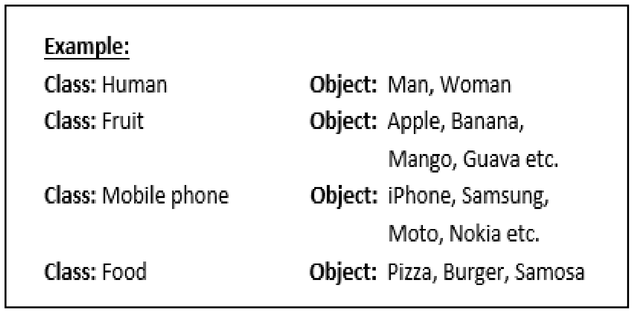
Basic Concepts of Object-Oriented Programming:

- OOP is a methodology or paradigm to design a program using classes and objects.
- Alan Kay coined the term “object oriented programming” at grad school in 1966 or 1967.
- It simplifies software development and maintenance by providing some concepts:
  1. Object
  2. Class
  3. Inheritance
  4. Polymorphism
  5. Abstraction
  6. Encapsulation



Objects

- Any entity that has state and behavior is kn
- An object is an instance of class.
- For, example a chair, pen, table, keyboard, bike, etc.



Example:

Class: Human	**Object:** Man, Woman
Class: Fruit	**Object:** Apple, Banana, Mango, Guava etc.
Class: Mobile phone	**Object:** iPhone, Samsung, Moto, Nokia etc.
Class: Food	**Object:** Pizza, Burger, Samosa

Class

- Collection of objects is called class.
- Class is a user-defined data-type
- It is the basic building block of OOP.

Inheritance

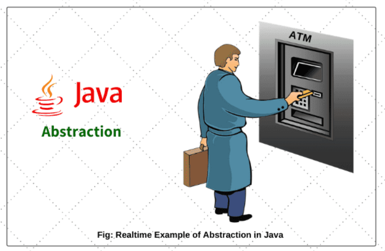
- When one object acquires all the properties and behaviors of a parent object, it is known as inheritance.

Polymorphism

- Existing in multiple forms.
- If one task is performed by different ways, it is known as polymorphism.

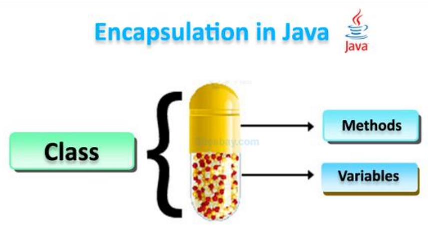
Abstraction

- Hiding internal details and showing functionality is known as abstraction.
- For, example phone call, we don't know the internal processing.



Encapsulation

- Binding (or wrapping) code and data together into a single unit are known as encapsulation.
- For, example capsule, it is wrapped with different medicines.



Features of Java:

- The primary objective was to make it portable, simple and secure programming language.
- The features of Java are also known as Java buzzwords.
- Java supports dynamic compilation and automatic memory management (garbage collection).
- A list of the most important features of the Java language is given below.
  1. Simple
  2. Object-Oriented
  3. Portable
  4. Platform independent
  5. Secured
  6. Dynamic etc.

Simple:

- It is very easy to learn, and its syntax is simple, clean and easy to understand.
- According to Sun Microsystem, Java language is a simple programming language because:
- Java syntax is based on C++.

Object-oriented:

- Java is an object-oriented programming language. Everything in Java is an object.
- OOPs is a methodology that simplifies software development and maintenance by providing some rules.

Platform Independent:

- Java code can be executed on multiple platforms, for example, Windows, Linux, Mac/OS, etc.
- Java code is compiled by the compiler and converted into bytecode.
- This bytecode is a platform-independent code because it can be run on multiple platforms, i.e., Write Once and Run Anywhere (WORA).

Portable

- Java is portable because it facilitates you to carry the Java bytecode to any platform.
- It doesn't require any implementation.

Secured:

- Java is best known for its security.
- With Java, we can develop virus-free systems.

Dynamic:

- Java is a dynamic language & it supports the dynamic loading of classes.
- It means classes are loaded on demand.

Explain the difference between POP & OOP.

POP	OOP
POP Stands for Procedural Oriented Programming.	OOP Stands for Object Oriented Programming.
Entire program is divided into functions.	Entire program is divided into objects.
It follows Top-down approach.	It follows Bottom-up approach.
No access specifiers are supported.	Access specifiers are supported.
Here, no concept of overloading.	It overloads functions, constructors & operators.
Inheritance is not supported.	Inheritance is supported.
No data hiding is present, so data is insecure	Encapsulation is used to hide in data.
C, VB, FORTRAN, Pascal etc.	C++, JAVA, VB.NET, Python etc.

Write the difference between JDK, JRE and JVM.

- JDK is the development platform, while JRE is for execution.
- JVM is the foundation, or the heart of the Java programming language.
- JVM is included in both JDK and JRE—Java programs won't run without it.

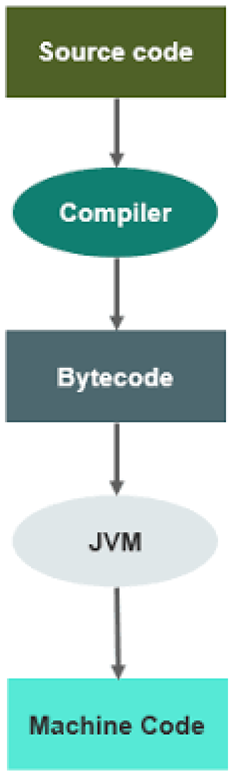
JDK	JRE	JVM
JDK stands for Java Development Kit.	JRE stands for Java Runtime Environment.	JVM stands for Java Virtual Machine.
The JDK is a software development kit that develops applications in Java.	The JRE is an implementation of JVM. It is a type of software package that provides class libraries of Java.	JVM is a platform-independent abstract machine that has three notions in the form of specifications.
The JDK is platform-dependent.	JRE, just like JDK, is also platform-dependent.	The JVM is platform-independent.
JDK = Development Tools + JRE (Java Runtime Environment)	**JRE** = Libraries for running the application + JVM (Java Virtual Machine)	**JVM** = Only the runtime environment that helps in executing the Java bytecode.
It consists of various tools required for writing Java programs.	If a user wants to run the Java applets, then they must install JRE on their system.	It provides its users with a platform-independent way for executing the Java source code.

What do you mean by Byte Code?

- Byte Code can be defined as an intermediate code generated by the compiler after the compilation of source code(JAVA Program).
- This intermediate code makes Java a platform-independent language.

How is Byte Code generated?

- Compiler converts the source code or the Java program into the Byte Code(machine code), and secondly, the Interpreter executes the byte code on the system.
- The Interpreter can also be called JVM(Java Virtual Machine).
- The byte code is the common piece between the compiler(which creates it) and the Interpreter (which runs it).



CTH EDUCATION Name of Faith Education

Unit – 02: Data types, variables, and Arrays

- Classification of various data types used in JAVA (including Integer, float, characters, Boolean), closer look at the literals used in java,
- Defining and initialization of variables, type conversion and casting, automatic type promotions in expressions,
- Arrays (one dimensional and multidimensional).

Questions to be discussed:

1. What do you mean by variable in Java? Also explain its type.
2. Discuss various data types used in java with proper examples.
3. Discuss type conversion and type casting with suitable example.
4. What is array? Explain types of array in java.

CTH EDUCATION Name of Faith Education

Variable:

- A variable is the name of a reserved area allocated in memory.
- In other words, it is a name of the memory location.
- It is a combination of "vary + able" which means its value can be changed.
- A variable is a container which holds the value while the Java program is executed.
- A variable is assigned with a data type.
- Variable is a name of memory location.

Declaration of variable:

Syntax:

Data_type variable_Name;

Example:

```
int a;
```

Initialization of variable:

Syntax:

Data_type variable_Name = value;

Example:

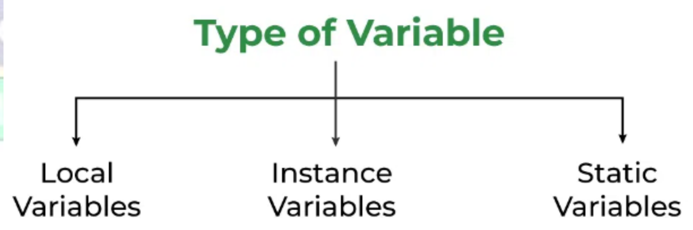
```
int a = 10;
```

CTH EDUCATION

Types of Variables:

There are three types of variables in Java:

1. local variable
2. instance variable
3. static variable



Local Variable:

- A variable declared inside the body of the method is called local variable.
- You can use this variable only within that method.
- A local variable cannot be defined with "static" keyword.

Instance Variable

- A variable declared inside the class but outside the body of the method, is called an instance variable.
- It is not declared as static.
- It is called an instance variable because its value is instance-specific and is not shared among instances.

Static variable

- A variable that is declared as static is called a static variable.
- It cannot be local.
- You can create a single copy of the static variable and share it among all the instances of the class.
- Memory allocation for static variables happens only once when the class is loaded in the memory.

Example to understand the types of variables in java:

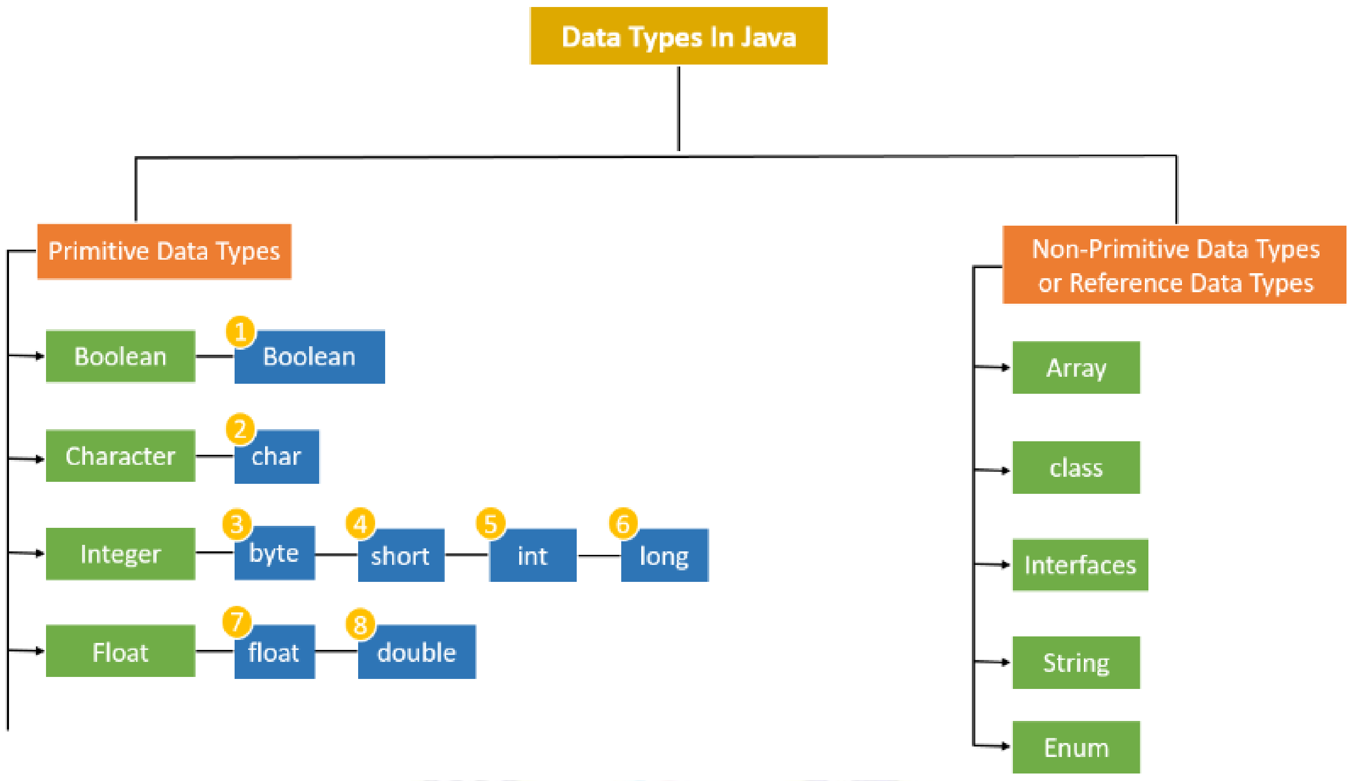
```
public class A
{
   static int m=100;//static variable
   void method( )
   {
      int n=90;//local variable
   }
   public static void main(String args[])
   {
      int data=50;//instance variable
   }
}
```

Data Types in Java:

- Data types specify the different sizes and values that can be stored in the variable.
- There are two types of data types in Java:
  1. **Primitive data types**
  2. **Non-primitive data types:** The non-primitive data types include Classes, Interfaces, and Arrays.

Primitive Data Types:

- Primitive data types specify the size and type of variable values.
- They are the building blocks of data manipulation and cannot be further divided into simpler data types.
- These are boolean, char, byte, short, int, long, float and double.



Primitive Data Types Table – Default Value, Size, and Range:

Data Type	Default Value	Default size	Range
byte	0	1 byte	-128 to 127
short	0	2 bytes	-32,768 to 32,767
int	0	4 bytes	2,147,483,648 to 2,147,483,647
long	0	8 bytes	9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
float	0.0f	4 bytes	1.4e-045 to 3.4e+038
double	0.0d	8 bytes	4.9e-324 to 1.8e+308
char	‘\u0000’	2 bytes	0 to 65536
boolean	FALSE	1 byte	0 or 1

Non-Primitive Data Types:

- Non-primitive data types or reference data types refer to instances or objects.
- They cannot store the value of a variable directly in memory.
- They store a memory address of the variable.
- All non-primitive data types are of equal size.

Type Casting:

- In type casting, a data type is converted into another data type by the programmer using the casting operator during the program design.
- In type casting, the destination data type may be smaller than the source data type when converting the data type to another data type, that's why it is also called narrowing conversion.

Syntax:

```
destination_datatype = (target_datatype)variable;
```

Example:

```
float x;
byte y;
y=(byte)x;
```

Note: ( ) - is a casting operator.

Type conversion:

- In type conversion, a data type is automatically converted into another data type by a compiler at the compiler time.
- In type conversion, the destination data type cannot be smaller than the source data type, that's why it is also called widening conversion.

Type Conversion example –

```
int x=30;
float y;
y=x;
```

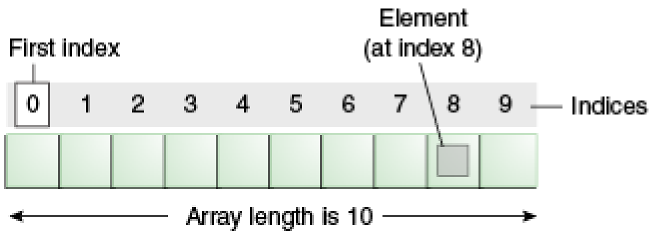
CTH EDUCATION Name of Faith Education

Difference Between Type Casting and Type Conversion:

TYPE CASTING	TYPE CONVERSION
Here, a data type is converted into another data type by a programmer using casting operator.	Whereas in type conversion, a data type is converted into another data type by a compiler.
Type casting can be applied to **compatible data types** as well as **incompatible data types**.	Whereas type conversion can only be applied to **compatible datatypes**.
In type casting, casting operator is needed in order to cast a data type to another data type.	Whereas in type conversion, there is no need for a casting operator.
In typing casting, the destination data type may be smaller than the source data type.	Whereas in type conversion, the destination data type can't be smaller than source data type.
Type casting takes place during the program design by programmer.	Whereas type conversion is done at the compile time.

Java Arrays:

- Java array is an object which contains elements of a similar data type.
- The elements of an array are stored in a contiguous memory location.
- It is a data structure where we store similar elements.
- We can store only a fixed set of elements in a Java array.
- In Java, array is an object of a dynamically generated class.



Types of Array in java:

- There are two types of array.
  1. Single Dimensional Array
  2. Multidimensional Array

Single Dimensional Array:

- An array having only one subscript variable is called One-Dimensional array.
- Single Dimensional array is used to represent and store data in a linear form.
- It is also known as linear Array.

Syntax to Declare an Array in Java:

Data_Type array_name[ ] = **new** datatype[size];

int a[ ] = new int[3];

Multidimensional Array:

- An array having two subscript variable is called two-dimensional array.
- Two-dimensional array is used to repersents Matrix.
- One Subscript Variable denotes the "**Row**" of a matrix.
- Another Subscript Variable denotes the "**Column**" of a matrix.
- The two-dimensional array can be defined as an array of arrays.
- Two-dimensional array is the simplest form of a multidimensional array.

Syntax to Declare Multidimensional Array in Java

Data_Type array_name[ ][ ] = **new** data_type[size][size];

int a[][] =**new int**[3][3];

Java Program to illustrate how to declare, instantiate, initialize:

```
class Testarray
{
        public static void main(String args[])
        {
                int a[]=new int[5];
                a[0]=10;
                a[1]=20;
                a[2]=70;
                a[3]=40;
                a[4]=50;
        for(int i=0;i<a.length;i++)
        System.out.println(a[i]);
        }
}
```

Java Program to illustrate the use of multidimensional array

```
class Testarray3
{
        public static void main(String args[])
        {
        //declaring and initializing 2D array
                int arr[][]={{1,2,3},{2,4,5},{4,4,5}};
                //printing 2D array
                for(int i=0;i<3;i++)
                {
                        for(int j=0;j<3;j++)
                        {
                                System.out.print(arr[i][j]+" ");
                        }
                 System.out.println();
                }
        }
}
```

Unit – 03: Operators and control statement

- Operators
  - Arithmetic operators,
  - Bitwise operators,
  - Relational operators,
  - Logical operators,
  - Assignment operator?
- Operator precedence,
- Java's selection statement
  - (if, switch statement),
  - iteration statement (while, do-while, for, nested loops)
  - Jump statement (break, continue).

CTH EDUCATION Name of Faith Education

Questions to be discussed:

1. Write a program to find out whether the given number is palindrome or not.
2. Discuss arithmetic, bitwise and relational operators with proper example.
3. What do you mean by bitwise logical operator and Boolean logical operator? Define it.
4. Explain working of break and continue with java code.
5. Write any program to check leap year in Java.
6. Write a program in Java to find the roots of quadratic equation.
7. Write a program in java to explain recursion.

Operators:

- Operators are the symbols which perform operations on various data items known as operands.
- There are many types of operators in Java which are given below:
  1. Unary Operator,
  2. Arithmetic Operator,
  3. Shift Operator,
  4. Relational Operator,
  5. Bitwise Operator,
  6. Logical Operator,
  7. Ternary Operator and
  8. Assignment Operator.

Java Unary Operator:

- Unary operators require only one operand.
- Unary operators are used to perform various operations i.e.:
  - ➢ incrementing/decrementing a value by one
  - ➢ negating an expression
  - ➢ inverting the value of a boolean

Java Unary Operator Example: ++ and --

Example:

```
public class OperatorExample
{
public static void main(String args[])
   {
      int x=10;
      System.out.println(x++);//10 (11)
      System.out.println(++x);//12
      System.out.println(x--);//12 (11)
      System.out.println(--x);//10
   }
}
```

```
public class OperatorExample
{
public static void main(String args[])
  {
      int a=10;
      int b=10;
      System.out.println(a++ + ++a);//10+12=22
      System.out.println(b++ + b++);//10+11=21
  }
}
```

CTH EDUCATION Name of Faith Education

Arithmetic Operators:

- Arithmetic operators are used to perform addition, subtraction, multiplication and division.
- They act as basic mathematical operations.

Example:

```
public class OperatorExample
{
public static void main(String args[])
        {
                int a=10;
                int b=5;
                System.out.println(a+b);//15
                System.out.println(a-b);//5
                System.out.println(a*b);//50
                System.out.println(a/b);//2
                System.out.println(a%b);//0
        }
}
```

Left Shift Operator(<<):

- Left shift operator is used to shift all of the bits in a value to the left side of a specified number of times.

Example:

```
public class OperatorExample
{
public static void main(String args[])
        {
                System.out.println(10<<2);//10*2^2=10*4=40
                System.out.println(10<<3);//10*2^3=10*8=80
                System.out.println(20<<2);//20*2^2=20*4=80
                System.out.println(15<<4);//15*2^4=15*16=240
        }
}
```

Java Right Shift Operator(>>):

- The Java right shift operator is used to move the value of the left operand to right by the number of bits specified by the right operand.

  Example:

```
public OperatorExample
{
        public static void main(String args[])
        {
                System.out.println(10>>2);//10/2^2=10/4=2
                System.out.println(20>>2);//20/2^2=20/4=5
                System.out.println(20>>3);//20/2^3=20/8=2
        }
}
```

Relational operators:

- It is used to establish a relationship between two operands.
- They compare the values and return a boolean result, either true or false.

Operator Type	**Symbol**	**Syntax**	**Use Case**
Equal to	==	*a == b*	Checks if two values are equal
Not Equal to	!=	*a != b*	Checks if two values are not equal
Greater than	>	*a > b*	Checks if the left operand is greater than the right operand
Less than	<	*a < b*	Checks if the left operand is less than the right operand
Greater than or equal to	>=	*a >= b*	Checks if the left operand is greater than or equal to the right operand
Less than or equal to	<=	*a <= b*	Checks if the left operand is less than or equal to the right operand

```
public class AgeVerification
{
   public static void main(String[] args)
     {
                int age = 20;
                if (age >= 18)
                {
                            System.out.println("Eligible to vote.");
                }
                else
                {
                            System.out.println("Not eligible to vote.");
                }
     }
 }
```

Logical Operators:

- Logical operators are used to check whether an expression is true or false.
- They are used in decision making.
- It is used to perform logical "AND", "OR" and "NOT" operations.

- **AND Operator** (**&&**) – if(a && b) [if true execute else don't]
- **OR Operator** (||) – if(a || b) [if one of them is true to execute else don't]
- **NOT Operator** (**!**) – !(a<b) [returns false if a is smaller than b]

```
class Main {
 public static void main(String[] args) {

  // && operator
  System.out.println((5 > 3) && (8 > 5));  // true
  System.out.println((5 > 3) && (8 < 5));  // false

  // || operator
  System.out.println((5 < 3) || (8 > 5));  // true
  System.out.println((5 > 3) || (8 < 5));  // true
  System.out.println((5 < 3) || (8 < 5));  // false

  // ! operator
  System.out.println(!(5 == 3));  // true
  System.out.println(!(5 > 3));  // false
 }
```

Assignment Operator:

- Java assignment operator is one of the most common operators.
- It is used to assign the value on its right to the operand on its left.

```
public class OperatorExample
{
    public static void main(String args[])
    {
        int a=10;
        int b=20;
        a+=4;//a=a+4 (a=10+4)
        b-=4;//b=b-4 (b=20-4)
        System.out.println(a);
        System.out.println(b);
    }
}
```

Ternary Operator:

- Java Ternary operator is used as one line replacement for if-then-else statement.
- It is the only conditional operator which takes three operands.

Example:

```
public class OperatorExample
{
    public static void main(String args[])
    {
        int a=2;
        int b=5;
        int min=(a<b)?a:b;
        System.out.println(min);
    }
}
```

CTH EDUCATION Name of Faith Education

Operator Precedence:

Operator Type	Category	Precedence
Unary	postfix	*expr*++ *expr*--
	prefix	++*expr* --*expr*
Arithmetic	multiplicative	* / %
	additive	+ -
Shift	shift	<< >>
Relational	comparison	< > <= >=
	equality	== !=
Bitwise	bitwise AND	&
	bitwise exclusive OR	^
	bitwise inclusive OR	|
Logical	logical AND	&&
	logical OR	||
Ternary	ternary	? :
Assignment	assignment	= += -= *= /= %= &= ^= |=

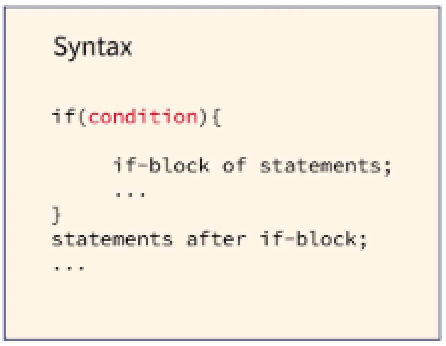
Selection Statement in Java:

- Selection statements are also known as branching or conditional or decision-making statements.
- It is used to select part of a program to be executed based on condition.
- These are statements that control programs in java.
- They are used for selecting a path of execution if a certain condition is met.
  1. (if, switch statement),
  2. iteration statement (while, do-while, for, nested loops)
  3. Jump statement (break, continue).

if Statement:

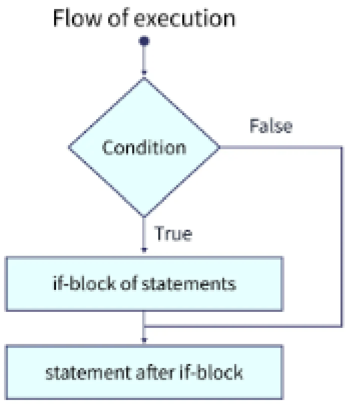
Syntax:

```
if(test expression)
statement
```



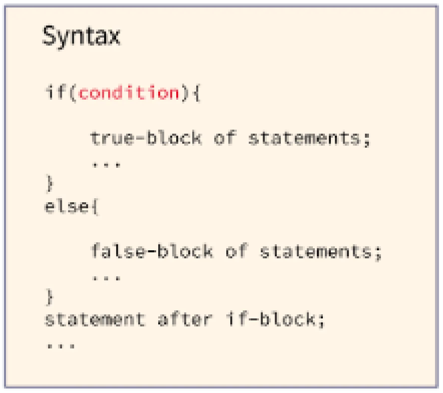
Syntax

```
if(condition){
     if-block of statements;
     ...
}
statements after if-block;
...
```



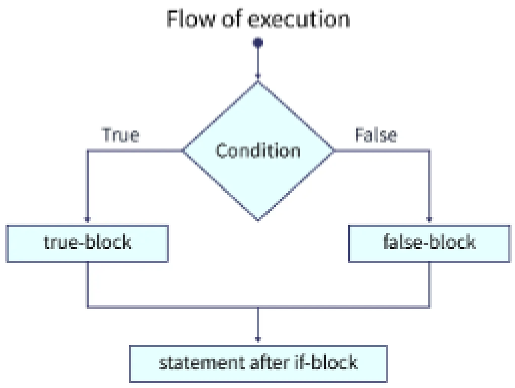
if-else Statement

```
if(test expression)
          Statement 1
else
          Statement 2
```



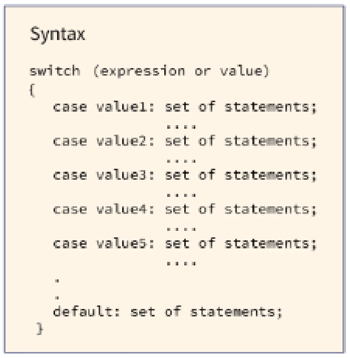
Syntax

```
if(condition){
     true-block of statements;
     ...
}
else{
     false-block of statements;
     ...
}
statement after if-block;
...
```



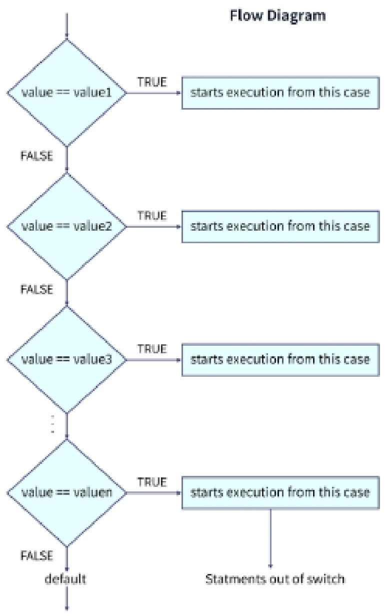
Switch Statement:

```
switch(expression)
{
    case value1: statements
    break;
    case value2: statements
    break;
    .........
    .........
    .........
    .........
    case value: statements
    break;
    default: statements
}
```



Syntax

```
switch (expression or value)
{
   case value1: set of statements;
                ....
   case value2: set of statements;
                ....
   case value3: set of statements;
                ....
   case value4: set of statements;
                ....
   case value5: set of statements;
                ....
   .
   .
   default: set of statements;
 }
```



CTH EDUCATION Name of Faith Education

Java Program to demonstrate the Switch statement where we are printing month name

```
public class SwitchMonthExample
{
  public static void main(String[] args)
     {
          int month=7;
          String monthString="";
          switch(month)
          {
               case 1: monthString="1 - January";
               break;
               case 2: monthString="2 - February";
               break;
               case 3: monthString="3 - March";
               break;
               case 4: monthString="4 - April";
               break;
               case 5: monthString="5 - May";
               break;
```

```
                    case 6: monthString="6 - June";
                    break;
                    case 7: monthString="7 - July";
                    break;
                    case 8: monthString="8 - August";
                    break;
                    case 9: monthString="9 - September";
                    break;
                    case 10: monthString="10 - October";
                    break;
                    case 11: monthString="11 - November";
                    break;
                    case 12: monthString="12 - December";
                    break;
                    default:System.out.println("Invalid Month!");
                }
        System.out.println(monthString);
        }
    }
```

Loops in Java:

- Iteration statements are also called as looping statements.
- It is used to repeat the statements until specified condition is met.
- In Java, we have the following looping statements:
    - For
    - while
    - do...while

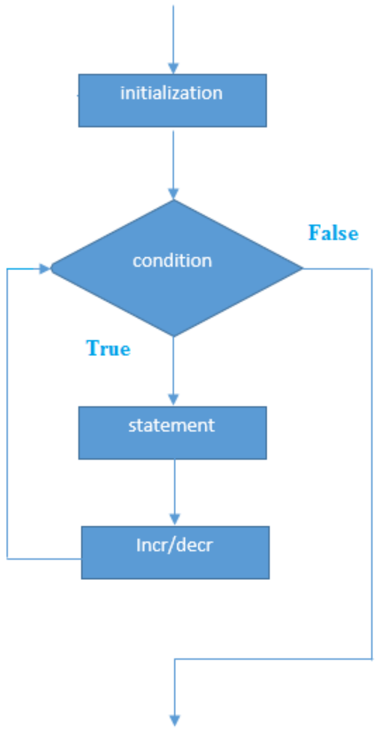
for loop:

- It provides a concise way of writing the loop structure.
- A for loop statement consumes the initialization, condition and increment/decrement in one line.

Syntax:

```
for (initialization; condition; increment/decrement)
{
        statement(s)
}
```

Java Program to demonstrate the example of for loop



```
public class ForExample
  {
       public static void main (String[] args)
        {
              for (int i=0;i<=10;i++)
              {
                 System.out.println(i);
              }
         }
}
```

Java Infinitive for Loop:

- If you use two semicolons ;; in the for loop, it will be infinitive for loop.

Syntax:

```
for( ; ; )
{
   //code to be executed
}
```

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.
- It is also known as entry control loop.

Syntax:

```
initialization;
while (condition)
{
     loop statements...
}
```

do-while loop:

- Loops in Java come into use when we need to repeatedly execute a block of statements.
- Java do-while loop is an Exit control loop.
- Therefore, unlike ***for*** or ***while*** loop, a do-while check for the condition after executing the statements of the loop body.

Syntax:

```
do
{
   ....................
   Update_expression
}while (test_expression);
```

for loop	**while loop**	**do-while loop**
for loop is a control flow statement that iterates a part of the programs multiple times.	It is a control flow statement that executes a part of the programs repeatedly on the basis of given boolean condition.	It is a control flow statement that executes a part of the programs at least once and the further execution depends upon the given boolean condition.
If the number of iteration is fixed, it is recommended to use for loop.	If the number of iteration is not fixed, use while loop.	If the number of iteration is not fixed and you must have to execute the loop at least once, then use the do-while loop.
for(init;condition;incr/decr) { // code to be executed }	while(condition) { //code to be executed }	Do { //code to be executed }while(condition);
for(int i=1; i<=10; i++) { System.out.println(i); }	int i=1; while(i<=10) { System.out.println(i); i++; }	int i=1; do { System.out.println(i); i++; }while(i<=10);
for(;;) { //code to be executed }	while(true) { //code to be executed }	do { //code to be executed }while(true);

Jump statements in Java:

Break Statement

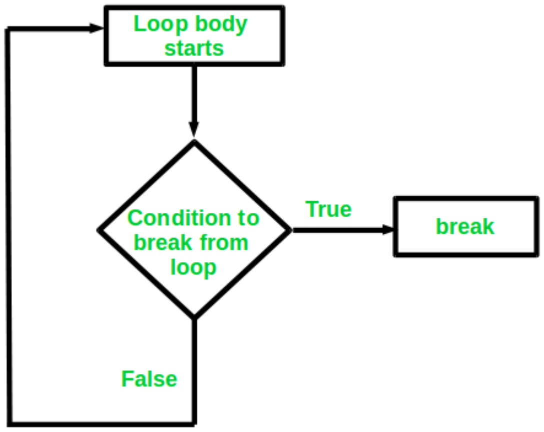
- When a break statement is encountered inside a loop, the loop is immediately terminated.
- The Java *break* statement is used to break loop or switch statement.
- It breaks the current flow of the program at specified condition.
- In case of inner loop, it breaks only inner loop.
- We can use Java break statement in all types of loops such as for loop, while loop and do-while loop.

Syntax:

```
jump-statement;
break;
```

Java Program to demonstrate the use of break statement inside the for loop.

```
public class BreakExample
{
        public static void main(String[] args)
        {
                for(int i=1; i<=100; i++)
                {
                        if(i==5)
                        {
                                break;
                        }
                        System.out.println(i);
                }
        }
}
```



Continue statement:

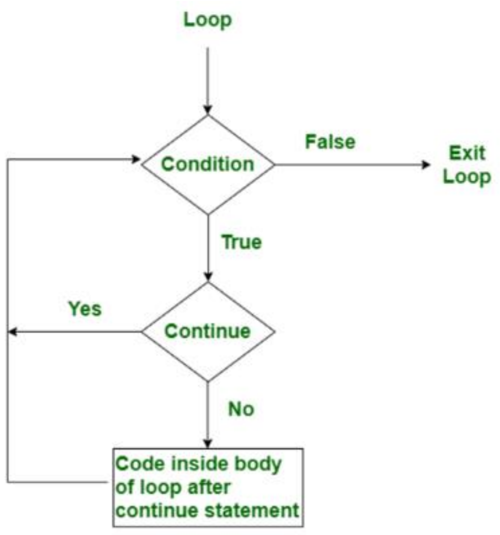
- It is often used inside in programming languages inside loops control structures.
- Inside the loop, when a continue statement is encountered the control directly jumps to the beginning of the loop for the next iteration instead of executing the statements of the current iteration.
- The continue statement is used when we want to skip a particular condition and continue the rest execution.
- Java continue statement is used for all type of loops but it is generally used in for, while, and do-while loops.

Syntax:

```
continue;
```

Java Program to demonstrate the use of continue statement

```
public class ContinueExample
{
        public static void main(String[] args)
        {
                for(int i=1; i<=10; i++)
                {
                        if(i==5)
                        {
                                Continue;
                        }
                        System.out.println(i);
                }
        }
}
```



Write a program to find out whether the given number is palindrome or not.

```
#include<stdio.h>
int main()
{
        int n, r, sum=0, temp;
        printf("enter the number=");
        scanf("%d", &n);
        temp=n;
        while(n>0)
        {
                r=n%10;
                sum=(sum*10) + r;
                n=n/10;
        }
        if(temp==sum)
                printf("palindrome number ");
        else
                printf("not palindrome");
        return 0;
}
```

CTH EDUCATION Name of Faith Education

Write any program to check leap year in Java.

```
class Solution
{
        public static void main(String[] args)
        {
                int year = 2023;
                if (year % 400 == 0)
                {
                        System.out.println(year + " is a Leap Year.");
                }
                else if (year % 100 == 0)
                {
                        System.out.println(year + " is not a Leap Year.");
                }
                 else if (year % 4 == 0)
                {
                        System.out.println(year + " is a Leap Year.");
                }
                else
                {
                        System.out.println(year + " is not a Leap Year.");
                }
        }
}
```

Write a program in Java to find the roots of quadratic equation.

```
public class Main
{
   public static void main(String[] args)
    {
          double a = 7.2, b = 5, c = 9;
          double x1, x2;
          double det = b * b - 4 * a * c;
           if (det > 0)
          {
                 X1 = (-b + Math.sqrt(det)) / (2 * a);
                 X2 = (-b - Math.sqrt(det)) / (2 * a);
                 System.out.format( "First Root = %.2f and Second Root = %.2f", x1, x2);
          }
          else if (det == 0)
          {
                    x1 = x2 = -b / (2 * a);
                    System.out.format("First Root = Second Root = %.2f;", x1);
          }
          else
          {
                    double real = -b / (2 * a);
                    double imaginary = Math.sqrt(-det) / (2 * a);
                    System.out.printf("First Root = %.2f+%.2fi", real, imaginary);
                    System.out.printf("\nSecond Root = %.2f-%.2fi", real, imaginary);
          }
    }
}
```

CTH EDUCATION Name of Faith

Unit – 04: Classes and methods

- Class fundamentals,
- Declaring objects, assigning object to reference variables,
- Constructors (default and parameterized), overloading constructor,
- Understanding this keyword, finalize keyword, static keyword, final keyword,
- garbage collection,
- method introduction and returning a value from a method,
- Overloading method,
- Recursion,
- Introduction to inner and nested classes, command line argument.

Questions to be discussed:

1. What is class in java? How to declare class in java?
2. What is constructor? Explain different types of constructor in java.
3. What do you mean by constructor and destructor?
4. Differentiate between final and finalize in java.
5. What do you mean by method in java? Why we use method in java program?
6. Discuss about method overloading and constructor overloading in java.
7. What do you mean by recursive method? Explain it with example.
8. Write short notes on:
    a. This keyword
    b. Static keyword
    c. Garbage collection

CTH EDUCATION SAHARSA ALOK SIR

What is an object in Java?

- An object is a real-world entity.
- An entity that has state and behavior is known as an object.
- Example: chair, bike, marker, pen, table, car, etc.
- An object is an instance of a class.
- A class is a template or blueprint from which objects are created.
- So, an object is the instance (result) of a class.

How to Create Object in Java?

- The **object** is a basic building block of an OOPs language.
- In **Java**, we cannot execute any program without creating an **object**.
- Using the **new** keyword we create an object of the class.

  Syntax:

  ClassName object = **new** ClassName( );

Assigning object to reference variables:

- When we create an object of class then space is reserved in heap memory.
- Now, the space in the memory is created but how to access that space.
- Then, we create a reference variable which simply points out the Object.
- So simply we can say that the reference variable is used to point an object.
- By default, if no object is passed to a reference variable then it will store a null value.

  Example:

```
Myclass D1= new Myclass( );
int i = 10;
```

CTH EDUCATION SAHARSA ALOK SIR

What is a class in Java?

- A class is a group of objects which have common properties.
- It is a template or blueprint from which objects are created.
- It is a logical entity, it can't be physical.
- A class in Java can contain property and methods.

How to declare class in java?

- Java provides a reserved keyword **class** to define a class.
- The keyword must be followed by the class name.
- Inside the class, we declare methods and variables.

```
class class_name
{
    // member variables
    // class methods
}
```

Difference between object and class:

Object	Class
Object is an instance of a class.	It is a blueprint from which objects are created.
Object is a real world entity such as pen, laptop, mobile, bed, mouse, chair etc.	Class is a group of similar objects.
Object is a physical entity.	Class is a logical entity.
Object is created through new keyword.	Class is declared using class keyword.
Object is created many times as per requirement.	Class is declared once.
Object allocates memory when it is created.	Class doesn't allocated memory when it is created.

Constructor in java:

- Constructor are the special member function.
- It is used to initialize the data members of the class.
- Constructor has the same name as the class name.
- A constructor has no return type not even void.
- Constructors are automatically invoked (call) as soon as the object of its class is created.
- It is called constructor because it constructs the values at the time of object creation.
- It is not necessary to write a constructor for a class, because java compiler creates a default constructor if your class doesn't have any.

How to declare Constructor?

- As we know that constructor has the same name as the class, thus it can be easily identified.
- Syntax:

```
class_name (arguments if any)
{
  ...
  ...
};
```

Types of constructor:

- There are two types of constructors in Java:
  1. Default constructor, and
  2. Parameterized constructor.

Default Constructor:

- A constructor having no arguments is called "Default Constructor.
- The default constructor is used to provide the default values to the object like 0, null, etc.

Parameterized Constructor:

- A constructor having arguments is called a parameterized constructor.
- The parameterized constructor is used to provide different values to distinct objects.
- We can have any number of parameters in the constructor.

Constructor overloading in Java:

- In Java, we can overload constructors like methods.
- It can be defined as the concept of having more than one constructor with different parameters so that every constructor can perform a different task.
- They are differentiated by the compiler by the number of parameters in the list and their types.

  Example:

```
class Student
{
        variables of the class
        Student()
        {
                ...........
        }
        Student(parameters)
        {
                ...........
        } ;
}
```

CTH EDUCATION SAHARSA ALOK SIR

Rules for creating Java constructor:

1. Constructor name must be the same as its class name.
2. A Constructor must have no return type.
3. A Java constructor cannot be abstract, static, final, and synchronized.

What is Destructor?

- A destructor is a member function of a class that deallocates the memory allocated to an object.
- A destructor is also declared and defined with the same name as that of the class.
- A destructor is preceded by a **tilde (~)** symbol.
- A single class has only a single destructor.

How to declare destructor?

Syntax:

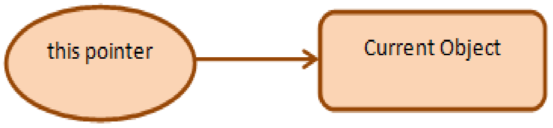
```
~ class_name (no arguments)
{
  …
  …
};
```

Difference between constructor and destructor:

CONSTRUCTOR	DESTRUCTOR
It allocates the memory to an object.	It deallocates the memory of an object.
class_name(arguments if any){ };	~ class_name(no arguments){ };
Constructor accepts argument	Destructor does not accept any argument.
Constructor is called automatically, while the object is created.	Destructor is called automatically, as block of program terminates.
There can be multiple constructors in a class.	There is always a single destructor in the class.
Constructors can be overloaded.	Destructor can not be overloaded.

This keyword:

- It is a keyword that refers to the current object in a method or constructor.
- If the name of instance variable and local variable are same then at the runtime JVM gets confused.
- To avoid this type of problem "this" keyword is used.
- It is used to call default constructor of its own class.
- It is used to called parametrized constructor of its own class.
- "this" can be passed as argument in the constructor call.
- "this" can be used to return the current class instance from the method.



Static keyword:

- It is a keyword which is used to make a variable or method constant.
- If you declare any variable as static, it is known as a static variable.
- The static keyword is used for efficient memory management.
- The static variable gets memory only once in the class area at the time of class loading.
- We can apply static keyword with variables, methods, blocks and nested classes.

Final Keyword:

- The final keyword in java is used to restrict the user.
- Using the final keyword means that the value can't be modified in future.
- The **final** keyword can be used with class method and variable.
- A final class cannot be inherited, a final method cannot be overridden and a final variable cannot be reassigned.

Difference between final, finally and finalize:

Final	finally	finalize
final is the keyword which is used to apply restrictions on a class, method or variable.	finally is the block in Java Exception Handling to execute the important code whether the exception occurs or not.	finalize is the method in Java which is used to perform clean up processing just before object is garbage collected.
Final keyword is used with the classes, methods and variables.	Finally block is always related to the try and catch block in exception handling.	finalize() method is used with the objects.
Once declared, final variable then it cannot be modified, overridden & inherited.	finally block runs the important code even if exception occurs or not & cleans up all the resources used in try block	Finalize method performs the cleaning activities with respect to the object before its destruction.
Final method is executed only when we call it.	Finally block is executed as soon as the try-catch block is executed.	finalize method is executed just before the object is destroyed.

What are Methods in Java?

- A method in Java is a block of code that performs specific actions mentioned in it.
- Method runs only when it is called.
- You can insert values or parameters into methods, and they will only be executed when called.
- They are also referred to as functions.

Why we use methods in Java?

- It allows code reusability (define once and use multiple times)
- You can break a complex program into smaller chunks of code
- It increases code readability

How to Declare Methods in Java?

- You can only create a method within a class.

Syntax:

```
Return_type function_name (parameters)
 {
         //method body
 }
```

Return Type:

- It defines the return type of the method.
- In the above syntax, "int" is the return type.
- We can mention void as the return type if the method returns no value.

Method Overloading in Java:

- Method Overloading is a feature that allows a class to have multiple methods with the same name but with different number, sequence or type of parameters.
- In short multiple methods with same name but with different signatures**.**
- For example:

```
add(int a, int b)          //having two int parameters
add(int a, int b, int c)          //having three int parameters
```

- Method overloading increases the readability of the program.
- There are different ways to overload the method:
  1. By changing number of arguments
  2. By changing the data type of parameters
  3. Changing the order sequence of parameters

Recursion in Java:

- The process in which a method calls itself directly or indirectly is called recursion.
- A method in java that calls itself is called recursive method.
- It makes the code compact but complex to understand.
- Using recursive algorithm, certain problems can be solved quite easily.

```
factorial(n) = n * factorial(n - 1) *..........1        where n ≥ 1
Exp:            factorial(5) = 5 * 4 * 3 * 2* 1 = 120
```

Syntax:

```
return_type method_name( )
{
        //code to be executed
        method_name();//calling same method
}
```

Write a Java program to find the factorial of a number?

```
public class Main
{
        static int factorial(int n)
        {
                if(n==0 || n=1)
                {
                        return 1;
                }
                else
                {
                        return n * factorial(n-1);
                }
        }
        public static void main(String[] args)
        {
                Int x = 5;
                System.out.println("The value of factorial x is:" + factorial(x));
        }
}
```

CTH EDUCATION SAHARSA ALOK SIR

Garbage collection:

- When Java programs run on the JVM, objects are created on the heap, which is a portion of memory dedicated to the program.
- Eventually, some objects will no longer be needed.
- The garbage collector finds these unused objects and deletes them to free up memory.
- It is the process by which Java programs perform automatic memory management.
- In other words, it is a way to destroy the unused objects.

Java Inner Classes (Nested Classes):

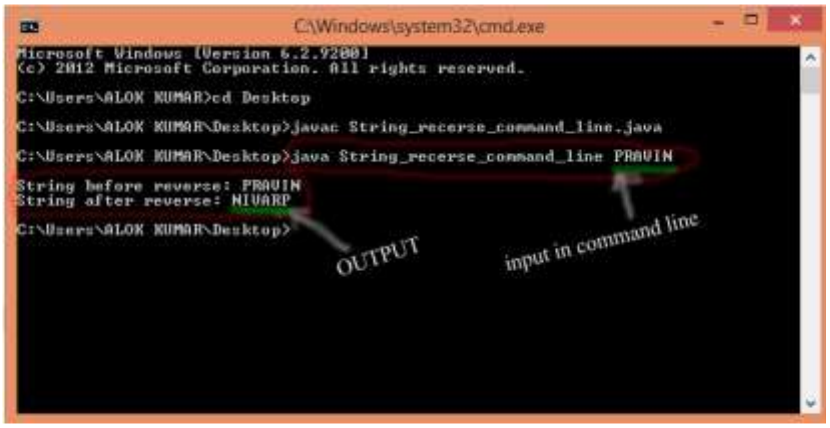
- In java, just like data member and member function we can also create a class as its member.
- The class is written within the class is called inner class.
- The class that holds the inner class is called outer class.
- The inner class is also known as nested class.

  Example:

```
Class outer
{
        int a;
        void add()
        {       }
        class inner
        {
        }
}
```

Command-line argument:

- It is an argument i.e. passed at the time of running the Java program.
- In the command line, the arguments passed from the console can be received in the java program and they can be used as input.
- The users can pass the arguments during the execution bypassing the command-line arguments inside the main( ) method.
- We need to pass the arguments as space-separated values.
- We can pass both strings and primitive data types (int, double, float, char, etc) as command-line arguments.



```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.2.9200]
(c) 2012 Microsoft Corporation. All rights reserved.

C:\Users\ALOK KUMAR>cd Desktop

C:\Users\ALOK KUMAR\Desktop>javac String_recerse_command_line.java

C:\Users\ALOK KUMAR\Desktop>java String_recerse_command_line PRAVIN

String before reverse: PRAVIN
String after reverse: NIVARP

C:\Users\ALOK KUMAR\Desktop>
```

Unit – 05: Inheritance and package introduction and Exception Handling

- Inheritance basic,
- Method overriding,
- abstract class, Object class,
- Defining a package, access protection, importing a package,
- Introduction to interface, variables in interface, extension of interface,
- Fundamentals of Exception handling, types of exception, creating your own exception.
- Use of try and catch, nested try block, throw, throws, finally keywords.

Questions to be discussed:

1. What is inheritance in java? Explain its different types.
2. What do you mean by multi-level inheritance? Explain it with example.
3. Differentiate between method overloading and method overriding.
4. What do you mean by interface? Define it with example.
5. What do you mean by class modifier? Define it with example.
6. What is Exception in Java? Explain its type in brief.
7. What do you mean by un-checked exceptions? Explain it.
8. Explain various available Exception keywords in java.
9. Write short notes on:
    a. Abstract class
    b. Object class
    c. Try & catch keyword

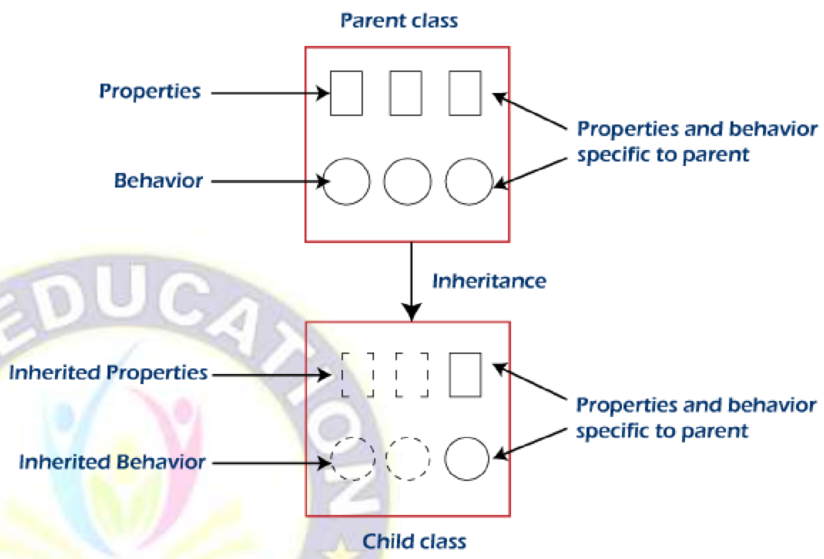
CTH EDUCATION SAHARSA ALOK SIR

Inheritance in Java:

- Inheritance is an important concept of OOPs.
- It is a mechanism in which a child object acquires all the properties and behaviors of a parent object.
- Inheritance means driving a new class from an existing class.
- The existing class is known as **base class** or **super class** or **parent class**.
- The new class is known as a **derived class** or **sub class** or **child class**.
- Inheritance provides the **reusability** of code especially when there is a large scale of code to reuse.

Syntax:

```
Class A
{
        .........
        .........
}
class B extends A
{
        .........
        .........
}
```



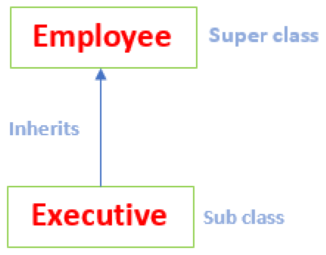
CTH EDUCATION SAHARSA ALOK SIR

- The **extends keyword** indicates that you are making a new class that derives from an existing class.
- The meaning of "extends" is to increase the functionality.

Types of inheritance in java:

- There are four types of inheritance in java:
  1. Single Inheritance
  2. Multilevel Inheritance
  3. Hierarchical Inheritance
  4. Hybrid Inheritance

Single Inheritance:

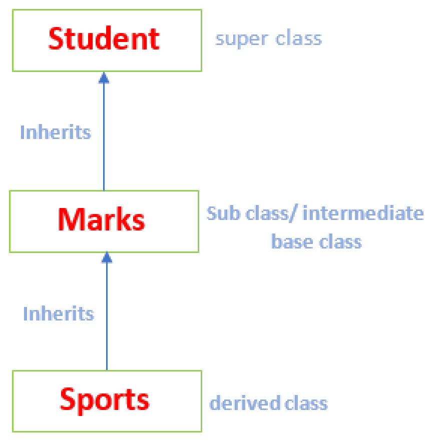


Single Inheritance

- In single inheritance, a sub-class is derived from only one super class.
- It inherits the properties and behavior of a single-parent class.
- Sometimes it is also known as **simple inheritance**.

Multi-level Inheritance:

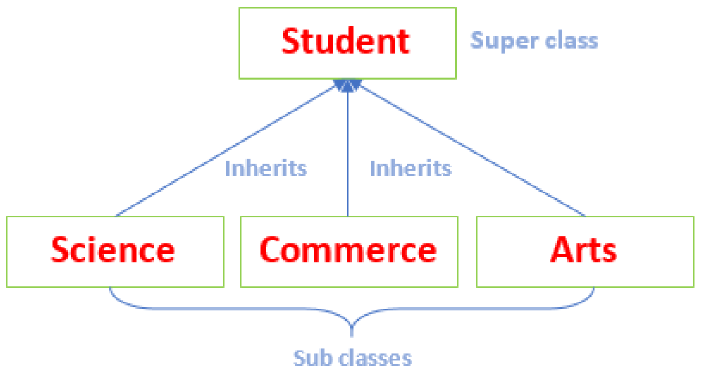
- In multi-level inheritance, a class is derived from a sub class which is also derived from another class.
- Note that the classes must be at different levels.
- It has a single base class and single derived class but multiple intermediate base classes.



Multi-level Inheritance

Hierarchical Inheritance:

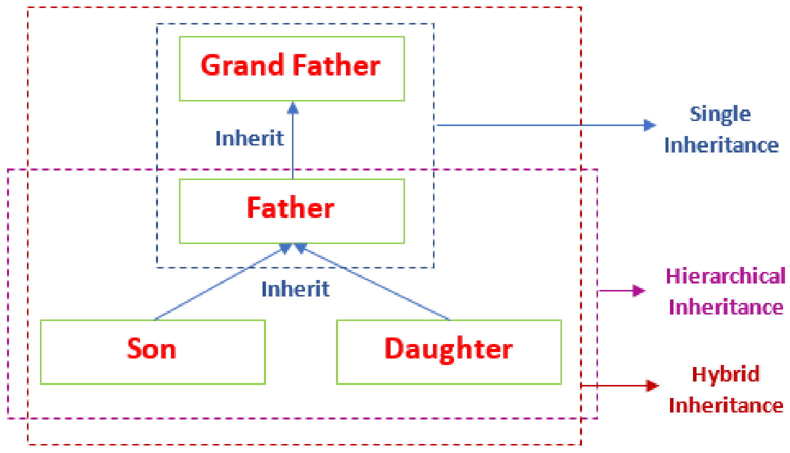
- If a number of classes are derived from a single base class, it is called hierarchical inheritance.



Hierarchical Inheritance

Hybrid Inheritance:

- Hybrid means consist of more than one.
- Hybrid inheritance is the combination of two or more types of inheritance.



Hybrid Inheritance

Method Overriding in Java:

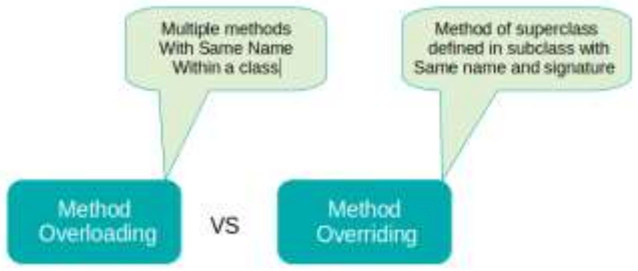
- If child class has the same method as declared in the parent class, it is known as method overriding.
- Method overriding is used for runtime polymorphism.
- Method overriding can't be achieved without inheritance.

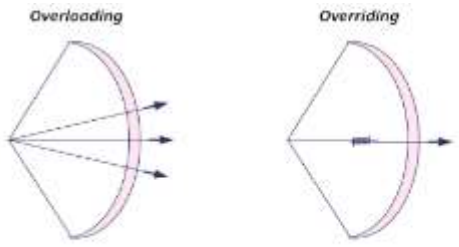
Syntax:

```
Class A
{
        void run()
        {

        }
        Class B extends A
        {
                void run()
                {

                }
        }
}
```


Multiple methods With Same Name Within a class
Method of superclass defined in subclass with Same name and signature
Method Overloading
VS
Method Overriding



Rules for Java Method Overriding

1. The method must have the same name as in the parent class
2. The method must have the same parameter as in the parent class.
3. There must be an IS-A relationship (inheritance).

Difference between method overloading and method overriding in java:

Method Overloading	Method Overriding
Overloading is a compile-time polymorphism.	Overriding is a run-time polymorphism.
It helps to increase the readability of the program.	It is used to specific implementation of the method which is already provided by its parent class.
It occurs within the class.	It is performed in two classes.
Method overloading may or may not require inheritance.	Method overriding always needs inheritance.
In method overloading, methods must have the same name and different signatures.	In method overriding, methods must have the same name and same signature.
Poor Performance due to compile time polymorphism.	It gives better performance.

CTH EDUCATION SAHARSA ALOK SIR

Abstract class in Java:

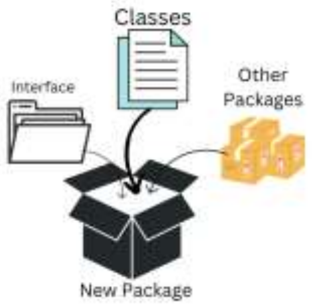
- Those class which has hiding the implementation and showing the function definition to the user is known as Abstract class.
- An abstract class must be declared with an abstract keyword.
- It can have abstract and non-abstract methods.
- We can't create an object of an abstract class.
- But we can create a reference variable of an abstract class because it refer to the objects of derived classes.

Object class in Java:

- The Object class is the parent class of all the classes in java by default.
- In other words, it is the topmost class of java.
- The Object class provides some common behaviors to all the objects such as object can be compared, object can be cloned, object can be notified etc.
- Object class is present in **java.lang** package.
- Every class in Java is directly or indirectly derived from the Object class.
- Therefore the Object class methods are available to all Java classes.
- Hence object class acts as a root of the inheritance hierarchy in any Java Program.

What is Package in Java?

- A java package is a collection of similar types of classes, interfaces and other packages.
- It works as containers for classes and other sub-packages.
- In other words we can say that the package is a way to group different classes or interfaces together.
- The grouping is done on the basis of functionality of classes and interfaces.
- There are two types of package in Java:
  1) Built-in Packages (packages from the Java API)
  2) User-defined Packages (create your own packages)



There are many built-in packages such as java, lang, awt, net, io, util, applet etc.

Import Package:

- We import the java package suing the keyword import.
- Suppose we want to use the student class stored in the java.util package then we can write the import statement at the beginning of the program like:

```
import.java.util.student
```

CTH EDUCATION SAHARSA ALOK SIR

Access Protection in Packages:

- Packages adds another dimension to the access control.
- Java provides many levels of security that provides the visibility of members within the classes, subclasses, and packages.
- Access modifiers define the scope of the class and its members.
- In java, the accessibility of the members of a class or interface depends on its access specifiers.

There are four types of Java access modifiers:

1. Public
2. Private
3. Protected
4. Default

Private:

- The access level of a private modifier is only within the class.
- It cannot be accessed from outside the class.

Public:

- The access level of a public modifier is everywhere.
- It can be accessed from within the class, outside the class, within the package and outside the package.

Protected:

- The access level of a protected modifier is within the package and outside the package through child class.
- If you do not make the child class, it cannot be accessed from outside the package.

Default:

- The access level of a default modifier is only within the package.
- It cannot be accessed from outside the package.
- If you do not specify any access level, it will be the default.

Interface in Java:

- Interface is just like a class in java.
- An interface is a fully abstract class.
- It includes a group of abstract methods without a body.
- We use the interface keyword to create an interface in Java.
- Interface variables are by default public, static and final.
- 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.

Extending an Interface:

- Similar to classes, interfaces can extend other interfaces.
- The extends keyword is used for extending interfaces.

Example:

```
Interface Line
{
        ...............// members of Polygon interface
}
Interface Polygon extends Line
{
        ...............// members of Polygon interface
        ...............// members of Line interface
}
```

What do you mean by Exception in Java?

- Exception is an abnormal condition of the program execution.
- It is an unexpected event, which occurs during the execution of a program.
- When exception occur then it disrupts the normal flow of the program at run time.
- When an exception occurs within a method, it creates an object, this object is called the exception object.
- It contains information about the exception, such as the name and description of the exception and the state of the program when the exception occurred.

Major reasons why an exception Occurs:

- Invalid user input
- Device failure
- Loss of network connection
- Physical limitations (out of disk memory)
- Code errors

CTH EDUCATION SAHARSA ALOK SIR

Exception Handling in Java:

- In Java, it is one of the powerful mechanism to handle the runtime errors so that the normal flow of the application can be maintained.

Types of Exception in Java?

There are two types of Exceptions:

1. **Checked Exceptions:**

- Checked exceptions are called compile-time exceptions because these exceptions are checked at compile-time by the compiler.

2. **Unchecked Exceptions:**

- The unchecked exceptions are just opposite to the checked exceptions.
- The compiler will not check these exceptions at compile time.
- In simple words, if a program throws an unchecked exception, and even if we didn't handle or declare it, the program would not give a compilation error.

Java Exception Keywords:

- Java provides five keywords that are used to handle the exception.
- The following table describes each.

Keyword	Description
try	▪ The "try" keyword is used to specify a block where we should place an exception code. ▪ It means we can't use try block alone. ▪ The try block must be followed by either catch or finally.
catch	▪ The "catch" block is used to handle the exception. ▪ It must be preceded by try block which means we can't use catch block alone. ▪ It can be followed by finally block later.
finally	▪ The "finally" block is used to execute the necessary code of the program. ▪ It is executed whether an exception is handled or not.
throw	▪ The "throw" keyword is used to throw an exception.
throws	▪ The "throws" keyword is used to declare exceptions. ▪ It specifies that there may occur an exception in the method. ▪ It doesn't throw an exception. ▪ It is always used with method signature.

Unit – 06: Multithreaded Programming and I/O

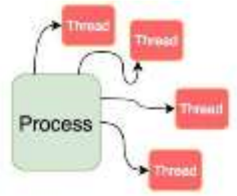
- The java thread model, thread priorities, synchronization, creating a thread, creating multiple thread,
- using is Alive() and join(),
- Synchronization in multiple thread,
- I/O basics, streams(byte and character),
- Reading and writing console input and output, Reading and writing files.

Questions to be discussed:

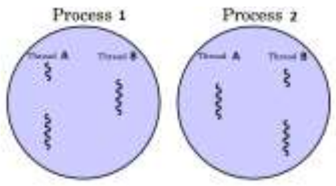
1. What is thread? Explain life cycle of thread in Java.
2. Differentiate between multi-tasking and multi-threading.
3. What do you mean by resuming and stoping of threads? Explain it.
4. What do you mean by stream in Java? Discuss its type in brief.
5. Write the difference between character stream and byte stream.
6. Explain input and output console in Java.
7. Write short notes on:
   a. Thread synchronization
   b. Alive() and join()

CTH EDUCATION SAHARSA ALOK SIR

What is a Thread in Java?



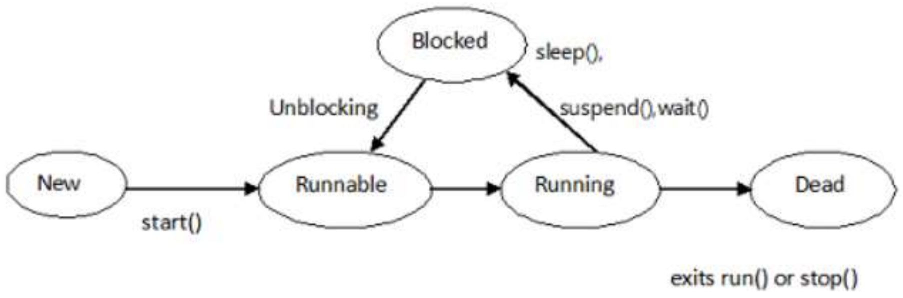
- Thread is a sub part of process.
- It is also known as light weight process.
- Multiple thread within a process share process state memory etc.
- Each thread in the program has its own program counter, stack, and local variable.
- All the programs have at least one thread, known as main thread, that is provided by the JVM at the starting of the program's execution.
- Multiple threads of execution can be run concurrently on the Java Virtual Machine.
- The priority of each thread varies, higher priority threads are executed first.



Lifecycle of a Thread in Java:

There are basically 5 stages in the lifecycle of a thread, as given below:

1. New
2. Runnable
3. Running
4. Blocked
5. Dead



New:

- A new thread is created but not working.
- A thread after creation and before invocation of start() method will be in new state.

Runnable:

- A thread after invocation of start() method will be in runnable state.
- A thread in runnable state will be available for thread scheduler.

Running:

- A thread in execution after thread scheduler select it, it will be in running state.

Blocked:

- A thread which is alive but not in runnable or running state will be in blocked state.
- A thread can be in blocked state because of suspend(), sleep(), wait() methods or implicitly by JVM to perform I/O operations.

Dead:

- A thread after exiting from run() method will be in dead state.
- We can use stop() method to forcefully killed a thread.

Multithreading in Java:

- In Java, multithreading is the method of running two or more threads at the same time to maximize CPU utilization.
- As a result, it is often referred to as Concurrency in Java.
- Each thread runs in parallel with the others.
- Since several threads do not assign different memory areas, they conserve memory.
- Furthermore, switching between threads takes less time.
- In Java, multithreading enhances program structure by making it simpler and easier to navigate.

Difference between Multi-tasking and Multi-threading:

Multitasking	Multithreading
In multitasking, users are allowed to perform many tasks by CPU.	In multithreading, many threads are created from a process.
Here, the processes share separate memory.	Here, processes are allocated the same memory.
It involves in multiprocessing.	It does not involve in multiprocessing.
In multitasking, the CPU is provided in order to execute many tasks at a time.	In multithreading, a CPU is provided in order to execute many threads of a process at a time.
In multitasking, processes don't share the same resources.	While in multithreading, each process shares the same resources.
Multitasking is slow compared to multithreading.	While multithreading is faster.
It helps in developing efficient programs.	It helps in developing efficient operating systems.

Java Thread Priorities:

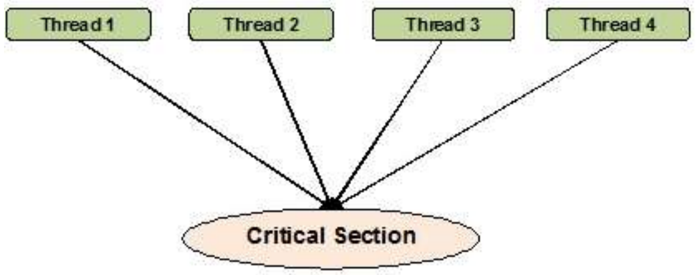
- The number of services assigned to a given thread is referred to as its priority.
- Any thread generated in the JVM is given a priority.
- The priority scale runs from 1 to 10.
- 1 is known as the lowest priority and 10 represents the highest level of priority.
- The main thread's priority is set to 5 by default, and each child thread will have the same priority as its parent thread.

Synchronization in multiple thread:

- Synchronization is a process of handling resource accessibility by multiple thread requests.
- The main purpose of synchronization is to avoid thread interference.
- At times when more than one thread try to access a shared resource, we need to ensure that resource will be used by only one thread at a time.
- The process by which this is achieved is called synchronization.
- The synchronization keyword in java creates a block of code referred to as critical section.

Syntax:

```
synchronized (object)
{
  //statement to be synchronized
}
```



Java Thread class:

- Java provides **Thread class** to achieve thread programming.
- Thread class provides methods to create and perform operations on a thread.
- Some important thread methods are:
  - start()
  - stop()
  - run()
  - suspend()
  - resume() etc.

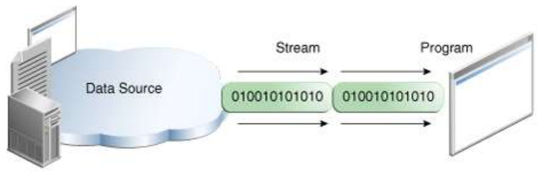
CTH EDUCATION SAHARSA ALOK SIR

Method	Description
start()	It is used to start the execution of the thread.
run()	It is used to do an action for a thread.
isAlive()	It tests if the thread is alive.
yield()	It causes the currently executing thread to pause & allow other threads to execute temporarily.
suspend()	It is used to suspend the thread.
resume()	It is used to resume the suspended thread.
stop()	It is used to stop the thread.

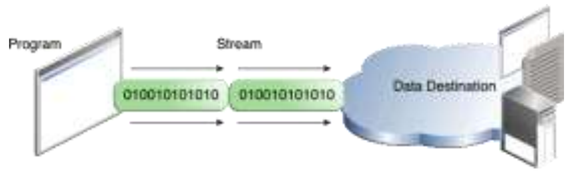
isAlive() and join():

- Sometimes one thread needs to know when other thread is terminating.
- In java, **isAlive()** and **join()** are two different methods that are used to check whether a thread has finished its execution or not.
- The **isAlive()** method returns **true** if the thread is still running otherwise it returns **false**.
- But, **join()** method is used more commonly than **isAlive()**.
- This method waits until the thread on which it is called terminates.
- Using **join()** method, we tell our thread to wait until the specified thread completes its execution.
- If we run a thread without using join() method then the execution of thread cannot be predict.

What is Stream?



- A stream is a sequence of data.
- In Java, a stream is composed of bytes.
- Java programs perform I/O operations using streams.
- It's called a stream because it is like a stream of water that continues to flow.
- An **input stream** is used to read data from the source.
- An **output stream** is used to write data to the destination.



Types of Streams:

Depending upon the data a stream can be classified into two types:

1. Byte Stream
2. Character Stream

Difference between character stream and byte stream:

Character stream	**Byte stream**
Character stream is used to read and write a single character of data.	Byte stream is used to read and write a single byte (8 bits) of data.
Reading or writing a character or text based I/O such as text file, text documents, XML, HTML etc.	Reading or writing to binary data such as exe file, image file, LLL formats file like .zip, .class, .exe etc.
Input and output character streams are called readers and writers respectively.	Input and output byte streams are simply called input streams and output streams respectively.
The abstract class reader and writer and their derived classes in java.io package provide support for character streams.	The abstract class InputStream and OutputStream and their derived classes in java.io package provide support for byte streams.

What is a Console?

- To run a program, we might need input from the user according to the requirement.
- We cannot always take input just from the program.
- Sometimes, we can take the input from the console or terminal too.
- The process of taking input from the console is introduced by the concept of Java Console.
- Java programming language provides several ways in order to take input from the console and provide its corresponding output on the same console.
- There are three ways to take the input from the Java console. They are:
  1. Using Java Scanner class (Basic level)
  2. Using Java BufferedReader class (Intermediate level)
  3. Using Java Console class

Scanner class in Java:

- A class that is used to scan inputs within the program is known as the Scanner class.

BufferedReader class in Java:

- It is one of the classical methods of taking input from the user.
- A new statement, " InputStreamReader " is also introduced along with the " BufferedReader " in order to scan the input values using BufferedReader.

Java Console class:

- The class " Console " in Java was introduced from Java version 1.5.
- The class " Console " can be accessed through the package " Java.io " which is the basic package that is used in all programs constructed in Java.
- There are several methods that are embedded within the " Console " class.

Methods in Console class:

readLine()

- The method that is used to read a single line of text or String from the console is the " readLine() " method.
- This method can be accessed or used when an object is created within the String class.

readPassword()

- The method that is used to read or scan a password as an input in a way such that the input taken is shown or visible at any time on the screen is the " readPassword() " method.
- **reader()**

- The method is called by an object created in the class " Reader() " which handles and relates to the console.
- This method also helps in the creation of the object in the class " Reader() ".

Console printf(String, Object)

- The method that is used in order to print a string within the output of the console is " printf() ".
- It can be accessed using the class " Console() ".

flush()

- The method that is used to clear all the statements or the matter present on the console is " flush() ".
- It is a general method which is neither creates an object nor will be accessed by an object created within a particular class.
- It just removes the matter on the console screen. Its return data type is generally " void ".

CTH EDUCATION SAHARSA ALOK SIR

Unit – 07: Database connectivity using JDBC driver Interface

- JDBC – JDVC Architecture – classes interfaces and drivers related to JDBC – connectivity to database using JDBC.

Questions to be discussed:

1. What do you mean by JDBC? Explain it.
2. Discuss the steps required to connecting database using JDBC.

What is an API?

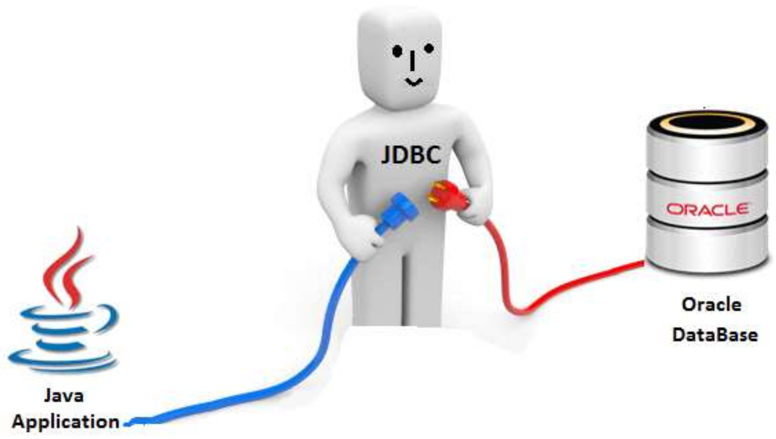
- API stands for Application Program Interface.
- It is a software intermediary that allows two applications to talk to each other.
- Each time you use an app like Facebook, send an instant message, or check the weather on your phone, you're using an API.
- There are three types of API.
  1. ODBC(Open Database Connectivity)
  2. OLE DB(Object Linking and Embedding)
  3. JDBC(Java Database Connectivity)

These APIs allow to connecting the application to a database in order to access data.

- The main difference between ODBC, OLEDB and JDBC is that:
- The ODBC is an API developed by Microsoft to access relational databases and
- OLEDB is an API developed by Microsoft to access both relational and non-relational databases.
- While JDBC is an API developed by Oracle to access the relational and non-relational database.

Java JDBC Tutorial

- JDBC stands for Java Database Connectivity.
- JDBC is a Java API to connect and execute the query with the database.
- Before JDBC, ODBC API was the database API to connect and execute the query with the database.
- But, ODBC API uses ODBC driver which is written in C language (platform dependent and unsecured).
- That is why Java has defined its own API (JDBC API) that uses JDBC drivers (written in Java language).
- JDBC API uses JDBC drivers to connect with the database.
- It acts as a bridge from your code to the database.
- JDBC Released as part of JDK 1.1 in 1997.



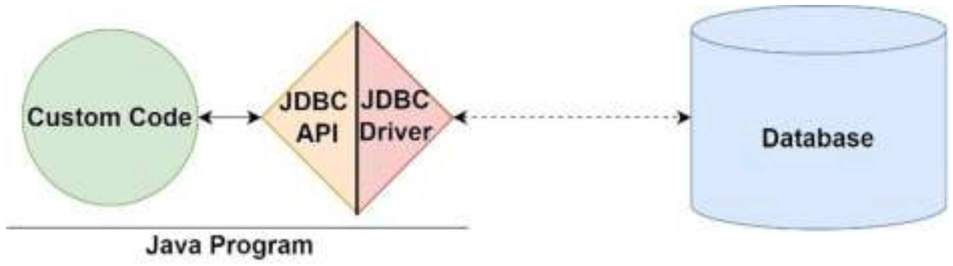
Uses of JDBC:

We can use JDBC API to handle database using Java program and can perform the following activities:

1. Connect to the database
2. Execute queries and update statements to the database
3. Retrieve the result received from the database.

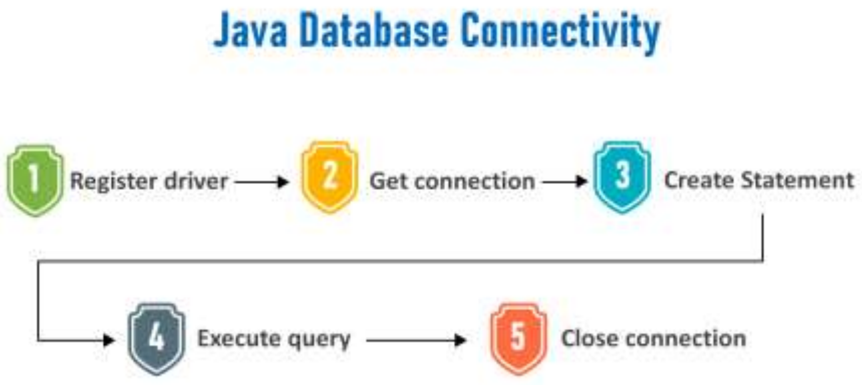
JDBC's architecture:

- The JDBC interface consists of two layers:
  1. The JDBC API supports communication between the Java application and the JDBC manager.
  2. The JDBC driver supports communication between the JDBC manager and the database driver.



JDBC – connectivity to database using JDBC.

- The steps for connecting to a database with JDBC are as follows:
  1. Install or locate the database you want to access.
  2. Include the JDBC library.
  3. Ensure the JDBC driver you need is on your classpath.
  4. Use the JDBC library to obtain a connection to the database.
  5. Use the connection to issue SQL commands.
  6. Close the connection when you are finished.



OOP Through Java

1. How to create a package? Explain with example.
2. What do you mean by interface? Define it with example.
3. What do you mean by un-checked exceptions? Explain it.
4. Explain java thread model. Discuss the thread priorities.
5. What do you mean by resuming and stopping of threads? Explain it.
6. Write any program of thread including suspend () and stop ().
7. Write the difference between character stream and byte stream.
8. Write short notes on:
    a. Thread synchronization
    b. Alive( ) and join( )