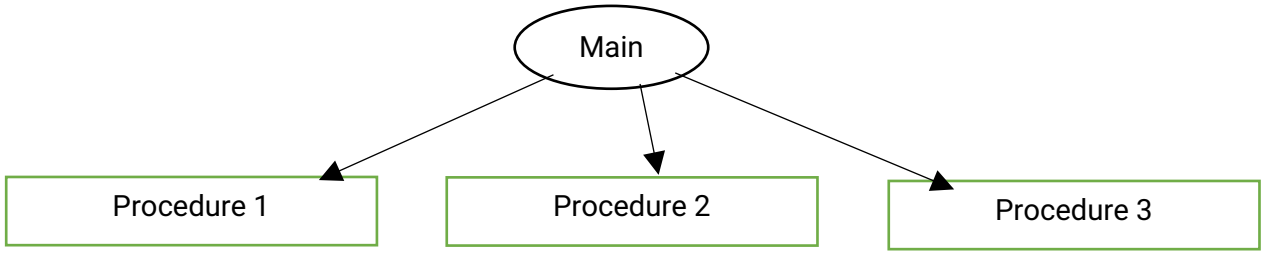
Object Oriented Programming with Java

Procedure-oriented programming: -

A type of programming where a structured method of creating programs is used. With procedure-oriented programming, a problem is broken up into parts and each part is then broken up into further parts. All these parts are known as procedures . They are separate but work together when needed. A main program centrally controls them all. Some procedure-oriented languages are COBOL, FORTRAN, and C.

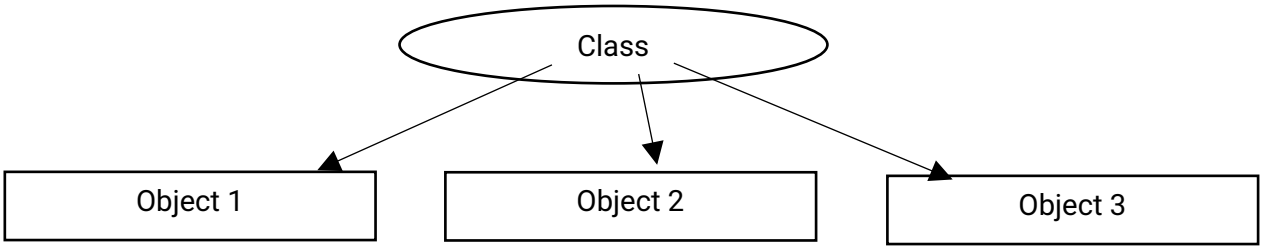


Object oriented programming: -

A type of programming where data types representing data structures are defined by the programmer as well as their properties and the things that can be done with them. With object-oriented programming, programmers can also create relationships between data structures and create new data types based on existing ones by having one data type inherit characteristics from another one.

In object-oriented programming, data types defined by the programmer are called classes (templates for a real world object to be used in a program). For example, a programmer can create a data type that represents a car - a car class. This class can contain the properties of a car (color, model, year, etc.) and functions that specify what the car does (drive, reverse, stop, etc.)

Some object-oriented languages are C++, Java, and PHP.



Edit with WPS Office

Difference between Object Oriented Programming and Procedure Oriented Programming

Object Oriented Programming	Procedure Oriented Programming
Data and functions of each individual object act like single unit.	Every function contains different data
Data gets more importance than functions in Program.	Functions get more importance than data in Program.
Data hiding is possible in Object Oriented Programming	. There is no perfect way for data hiding.

Concepts of Object Oriented Programming

1. class: - Class = data + methods. Everything (data and methods) in Java is contained in classes.

*class is a user defined data type that behave like built in types of an OOP language.

*class is a collection of fields (instance and class variable) and methods.

*class is a blue print or prototype that defines the variable and methods common to all objects of same kind.

Variables:

An instance variable *is a variable that is defined in a class, but outside of a method. There is one copy of the variable for every instance (object) created from that class. Instance variables can be accessed from inside any method, constructor or blocks of that particular class.*

Edit with WPS Office

A class variable or static variable *is defined in a class with static keyword, but there is only*
one copy regardless of how many objects are created from that class. Like: static int x;

A local variable *defined inside methods, constructors or blocks. The variable will be declared and initialized within the method and the variable will be destroyed when the method has completed.*

Final variables *are nothing but constants. We cannot change the value of a final variable once it is initialized. It's declare with final keyword. Like: final int x;*

Methods:
Instance method: *which invoke directly from the class by the object of related class*
Class / static method: *which invoke directly from the class by the class name, without object.*
A ***final method*** *cannot be overridden. Which means even though a sub class can call the final method of parent class without any issues but it cannot override it.*
We cannot extend a final class.
Access Specifiers / Modifiers:
Java provides a number of access modifiers to set access levels for classes, variables, methods and constructors. The four access levels are:

- *Visible to the package. The default. No modifiers are needed.*
- *Visible to the class only (private).*
- *Visible to the world (outside package and all classes) (public).*
- *Visible to the package and all subclasses (protected).*

2. Object: (Object is an instance of a class.)
Object is variable of class by which we can access the functions and variables of related class.
Object is an identifiable runtime entity with some characteristics and behavior. For example: Orange is an object of Fruit class and its characters are shape, colour and behavior is taste.
An object has three characteristics:

Edit with WPS Office

- ***state:*** It is represented by attributes of an object. It also reflects the properties of an object. *represents data (value) of an object.*
- ***behavior:*** It is represented by methods of an object. It also reflects the response of an object with other objects. *represents the behavior (functionality) of an object such as deposit, withdraw etc.*
- ***identity:*** It gives a unique name to an object and enables one object to interact with other objects. *Object identity is typically implemented via a unique ID. The value of the ID is not visible to the external user. But, it is used internally by the JVM to identify each object uniquely.*

  Example of an object: dog

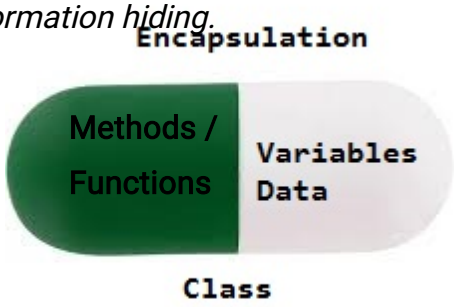
Identity	State/Attributes	Behaviors
Name of dog	*Breed* *Age* *Color*	*Bark* *Sleep* *Eat*

Objects correspond to things found in the real world. For example, a graphics program may have objects such as “circle”, “square”, “menu”. An online shopping system might have objects such as “shopping cart”, “customer”, and “product”.

3. Encapsulation: (To manage complexity)

Encapsulation is a process of binding data and functions together into a single unit. The wrapping up of data (characters) and function (behavior) into a single unit (class) is known as encapsulation.

Encapsulation leads to the concepts of data hiding but the concept of encapsulation must not be restricted to information hiding.



School bag is real life example of encapsulation.

4. Data Hiding: It is the implementation details of a class that are hidden from the user.

Edit with WPS Office

The concept of restricted access level programmers to write specialized functions or methods for performing operations on hidden members of class.

5. Abstraction: Abstraction refers to representing essential features without including background details or explanation.

Let us consider a car. In order to drive in uses are only aware of some essential features of a car. Users can apply break to stop the car, press acceleration to speed the car and press the clutch to change the gears. User's don't know what changes are going on in the machinery part of the engine.

The classes use the concept of data abstraction, they are known as abstract data type.

6. Inheritance: Inheritance is the process by which objects of one class can link and share some common properties of objects from another class.

Inheritance provides the idea of code reusability, this means that we can add additional features to an existing class without modifying it.

Types of Inheritance

1. *Single Inheritance*
2. *Multilevel Inheritance*
3. *Hierarchical Inheritance*
4. *Multiple Inheritance* ***(Not in JAVA)***

The class that is inherited by other class is called a **base/super/parent class**. The class that inherits the properties of the other class (super class) is called **sub/derive/child class.**

```
class Person
{
data members;
member methods;
}
class Employee extends Person
{
data members;
member methods;
}
```


Edit with WPS Office

Person is base/super/parent class. Employee is sub/derive/child class.

7. Polymorphism: (Poly - Many, Morphism - Form)

Polymorphism is an ability of a function or message to be processed in more than one form. It allows two or more classes to respond to the same message in different ways, this means that we can use the same name for a method in two different (inherited class) or same class.

Real life example of polymorphism:

Suppose if you are in class room that time you behave like student, when you are in market

at that time you behave like a customer, when you at your home at that time you behave like

a son or daughter, Here one person present in different-different behaviors.

Following concepts demonstrate different types of polymorphism in java.

1) Method Overloading

2) Method Overriding

1) Method Overloading:

In Java, two or more methods of same name in same class with different argument list or parameters. This concept is known as Method Overloading.

```
class Overload
{
   void demo (int a)
{
      System.out.println ("a: " + a);
   }
   void demo (int a, int b)
{
      System.out.println ("a and b: " + a + "," + b);
}
   double demo(double a)
{
      System.out.println("double a: " + a);
```


Edit with WPS Office

```
    return a*a;
  }
}
```

2) Method Overriding

Child class has the same method as of Parent class. In such cases child class overrides the parent class method without even touching the source code of the parent class. This feature is known as method overriding.

```
public class BaseClass
{
  public void methodToOverride() //Base class method
  {
    System.out.println ("I'm the method of BaseClass");
  }
}
public class DerivedClass extends BaseClass
{
  public void methodToOverride() //Derived Class method
  {
    System.out.println ("I'm the method of DerivedClass");
  }
}
```

8. Exception handling: An exception is an event which occurs during the execution of a program that disrupts the normal flow of the program.

An unexpected situation or unexpected error during program execution is known as exception and the way of handling this error in program is called exception handling.

An exception can occur because of many ways:

1. A file that needs to be open cannot be opened.
2. A user has entered invalid data.
3. A network connection has been lost in middle of communication.
4. JVM has run out of memory.

9. Message Passing: Message passing simply means that (at a very abstract level) the fundamental mechanism of program execution is objects sending each other messages.

Edit with WPS Office

Naming Convention in Java: -

- **Case Sensitivity -** Java is case sensitive which means identifier **Hello** and **hello** would have different meaning in Java.
- **Class Names -** For all class names the first letter should be in Upper Case. If several words are used to form a name of the class each inner words first letter should be in Upper Case.
  Example: *class MyFirstJavaClass*
- **Method Names -** All method names should start with a Lower Case letter. If several words are used to form the name of the method, then each inner word's first letter should be in Uppercase.
  Example: *public void myMethodName()*
  Program File Name - Name of the program file should exactly match the class name. When saving the file you should save it using the class name (Remember java is case sensitive) and append '.java' to the end of the name. (if the file name and the class name do not match your program will not compile).

IDE: -A **Java IDE** (for Integrated Development Environment) is a software application which enables users to more easily write and debug **Java** programs. Many **IDEs** provide features like syntax highlighting and code completion, which help the user to code more easily.
An IDE typically contains a code editor, a compiler or interpreter, and a debugger, accessed through a single graphical user interface (GUI). BlueJ, Eclipse, JBOSS are IDEs for Java.

BlueJ: BlueJ is a free IDE from Monash University, Australia. It is a window based platform
for Java Development Kit. it requires to install JDK 1.3 version or more. Some features are-

- It is a window based platform where the user can perform the tasks more conveniently.
- When we declare class, it facilities with a sample program to give a brief idea of programming.

Edit with WPS Office

- Debugging can also be done easily as it indicates the errors at the bottom of the same window.
- It also supports the syntax of the program, which is written on JDK.

Types of Errors

In general, there are three types of errors those occur in a computer program.

1. Syntax Error
2. Logical Error
3. Run Time Error

1. Syntax Error: These errors result when the rules or grammar of the programming language are not followed. Such programming errors typically involve incorrect punctuation,
incorrect word sequence, undefined terms or misuse of terms.

Example : A statement **c=a+b** has a syntax error because every Java statement should be terminated with a semicolon (;), so it should be **c=a+b;**

2. Logical Error: A logical error, is an error in planning the program's logic. The computer actually does not know that an error has been made. It follows the programming instructions
and executes the results, but the output is not correct. The problem is that the logic being
followed does not produce the results those were desired.

Example : An instruction to print 10 odd numbers

```
for(x=1;x<=10;x++)
System.out.println(x*3);
```

It will not print only odd numbers , it should be System.out.println(x*2-1) ;

3. Run Time Error: There may be some mistakes other than syntax error and logical error
which come in the way of execution, known as Run Time Error or Exception

Difference between Testing and Debugging

Testing	Debugging

Edit with WPS Office

Testing is a process in which program is Validated.	Debugging is a process in which the errors in the program are removed.
It is a positive activity that seeks to demonstrate that the program is correct.	It is a negative activity in that it is centered around eliminating of known error or bugs.
Testing is complete when all desired verifications against specifications have been performed	Debugging is finished when there are no error and the program is ready for execution.

JAVA

Java initially named as oak, was development by the team headed by James Gosling at Sun
Micro System USA in 1990, later acquired by Oracle Corporation.
The First reason behind the development of Java was the need for a platform independent and
portable software to be embedded in the electronic device like Television and Remotes etc.
The Second reason is the internet and www as these media needed portable and platform
independent programs.
Java is one of the most popular and widely used programming language and platform.
A platform is an environment that helps to develop and run programs written in any programming language.
Java is fast, reliable and secure. From desktop to web applications, scientific supercomputers to gaming consoles, cell phones to the Internet, Java is used in every nook and corner.

About Java

- **Java is a simple language:** Java is easy to learn and its syntax is simple and easy to understand. It is based on C++ (so easier for programmers who know C++). Java has removed many confusing and rarely-used features e.g. explicit pointers,

Edit with WPS Office

operator overloading etc. Java also takes care about memory management and for that, it provides an automatic garbage collector. This collects the unused objects automatically.

- **Java is a platform-independent language:** The programs written in Java language, after compilation, are converted into an intermediate level language called the **byte code** which is irrespective of the machine on which the programs run. This makes java highly portable as its byte codes can be run on any machine by an interpreter called the Java Virtual Machine(JVM) and thus java provides 'reusability of code'.

- **Java is an object-oriented programming language:** OOP makes the complete program simpler by dividing it into a number of objects. The objects can be used as a bridge to have data flow from one function to another. We can easily modify data and function as per the requirement.

- **Java is a robust language:** Java programs must be reliable because they are used in both consumer and mission-critical applications, ranging from Blu-ray players to navigation systems.
- **Java is a multithreaded language:** Java can perform many tasks at once by defining multiple threads. For example, a program that manages a Graphical User Interface (GUI) while waiting for input from a network connection uses another thread to perform the wait instead of using the default GUI thread for both tasks. This keeps the GUI responsive.

- **Java programs can create applets:** Applets are the programs which run on web browsers.

- **Java does not require any preprocessor:** It does not require inclusion of header files for creating a Java application.

Therefore, Java is a very successful language and its need is certainly going to increase day by day.

Parts of Java

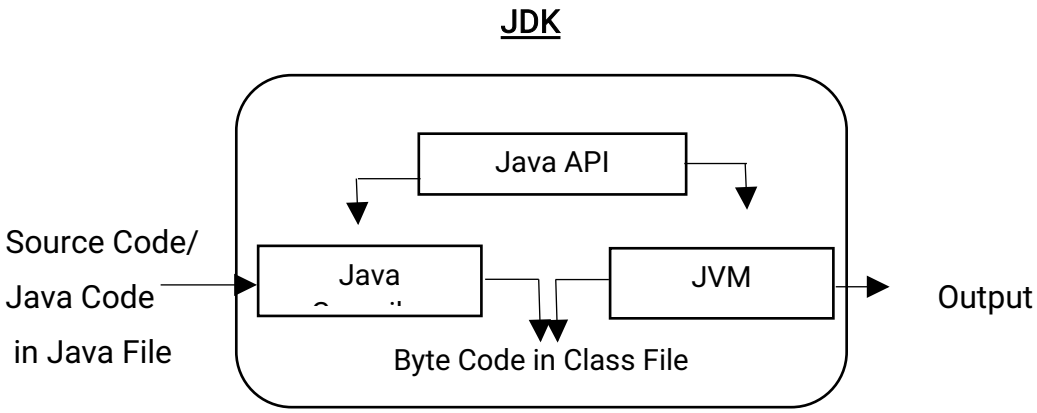
1. J2SE (Java 2 Standard Edition)

Edit with WPS Office

2. J2EE (Java 2 Enterprise Edition)
3. J2ME (Java 2 Micro Edition)

Execution of Java Program

1. **JDK (Java Development Kit):** JDK contains software development tools which are used to compile and run java programs.



2. **JRE (Java Runtime Environment):** JRE contains the run-time environment such as
   JVM and Java API (Java Application Programming Interface) that is java classes to run,
   java programs but doesn't contain any development tools Such as debugger and compiler.
3. **JVM (Java Virtual Machine):** Function of JVM is to load the appropriate class files for executing a java program and then execute it.
   JVM is an abstract machine designed to be implemented on the top of existing processor.
   Programs written in java are compiled into java byte code which is then interpreted by a
   special java interpreter for a specific platform actually this java interpreter is known as
   JVM.

4. **Java Compiler:** Java compiler doesn't convert source code to machine code rather it

Edit with WPS Office

converts source code to special intermediate code known as Byte code.



5. **Byte Code:** The byte code so generated is in the form of class file that can be interpreted. The byte code is machine independent that is it can run on any machine with the help of JVM.

6. **Java API:** Java Application Programming Interface that contains java packages and java
   classes. It is java library.
   Types of Java Programs: - There two types of Java Programs:
   1. **Internet Applets: -** They are small programs that are embedded in web page and run
      on web browser .
   2. **Stand Alone: -** Every standalone application of Java begins executing with a main
      method. To types of standalone applications:-
      - GUI Applications
      - Console Applications

Difference between Compiler and Interpreter:-

Compiler	Interpreter
It converts the whole source program into object program at once.	It converts the source program into object program one line at a time.
It displays the error for the whole program together after compilation.	It displays the error of one line at a time and after correction the control goes to the next line.

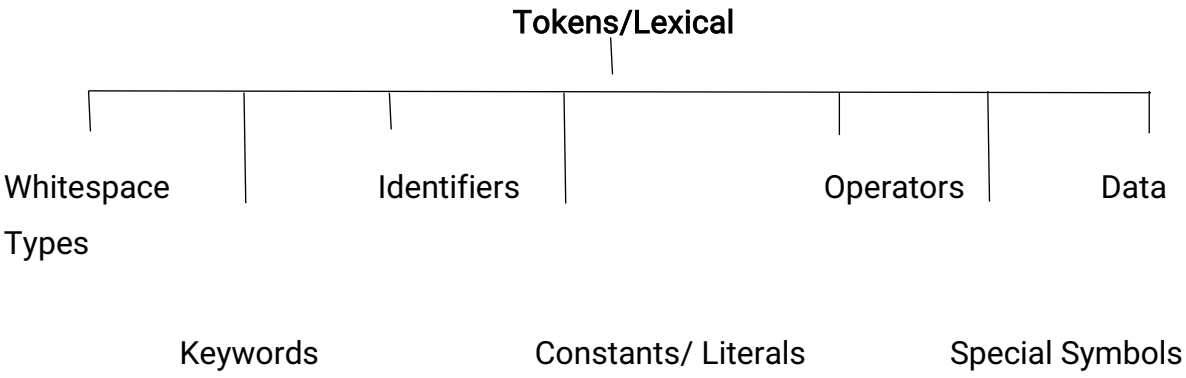
Steps to learn languages

Edit with WPS Office

English	**Programming**
Alphabets	Latters, Digits
|	|
Words	Tokens
|	|
Sentences	Instructions
|	|
Paragraph	Program
|	|
Lesson	Software

Tokens

In Java program a smallest individual units are known as Java tokens or lexical. Programs are written using these tokens and the syntax of language.



1. Whitespace: - White space is a tab space or a new line.

2. Keywords: - Keywords are reserved words that convey special meaning to the compiler. All keywords written in lowercase letters, they cannot be used as identifiers.
Some identifiers are class, void, if, else, switch, case, break, int, char, double, float, long, short, while, do, for, interface, extends, implements, import, try, catch, finally, throw, throws etc.

3. Identifiers: - Identifiers refers to the name of variables, functions and class. These are user-defined names. It is a sequence of letters, digits, underscore(_) and dollar ($) sign

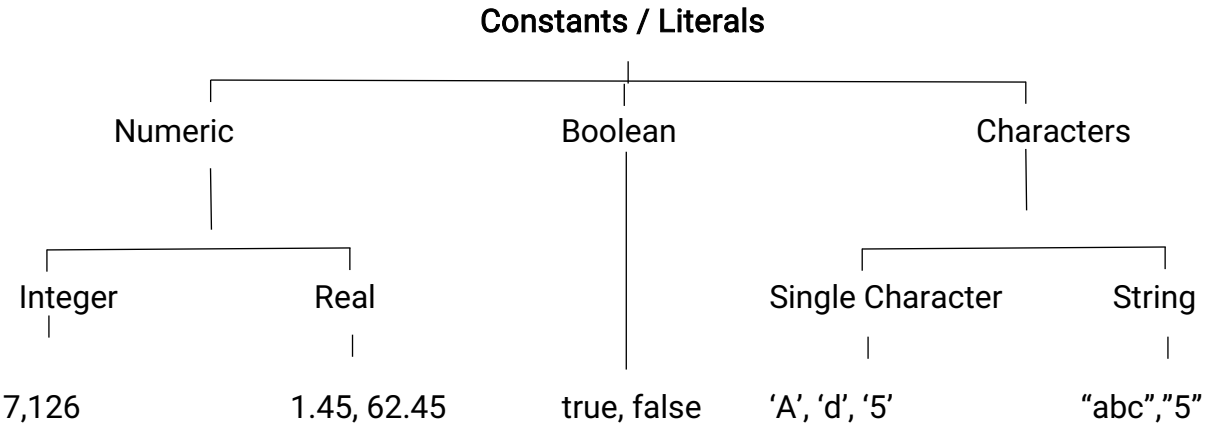
Edit with WPS Office

character.

Some examples of valid identifiers are :- a, total_cost, marks1, marks2, a3 etc.

Some examples of invalid identifiers are :- 2a, total cost, / , %b etc.

4. Constants / Literals: - A data item which doesn't changes its value during the execution of a
program is called constants or literals.



5. Operators: - It is a symbol or letters which makes the compiler perform specific operation on
the operands in a program.

Operands can be constants and variables. For example - a + 20

In above example a and 20 are operands and + is an operator.

Types of operators

1. *Arithmetic Operators (- ,+, *, /, %)*
2. *Relation Operators (<, >, ==, !=, >=, <=)*
3. *Assignment Operators (=)*
4. *Conditional / Ternary Operators (?, :)*
5. *Logical Operators (&&, ||, !)*
6. *Bitwise Operators (&, |, ^)*

6. Special Symbols:- In Java there are two categories of special symbols-

1. **Escape Sequence Characters: -** They are single characters used with backslash (\)

Edit with WPS Office

symbols.

Symbol	Purpose
\n	Insert a New line
\t	Insert a Tab space
\b	Insert a backspace
\'	Insert a single quote character
\"	Insert a double quote character
\\	Insert a backslash character

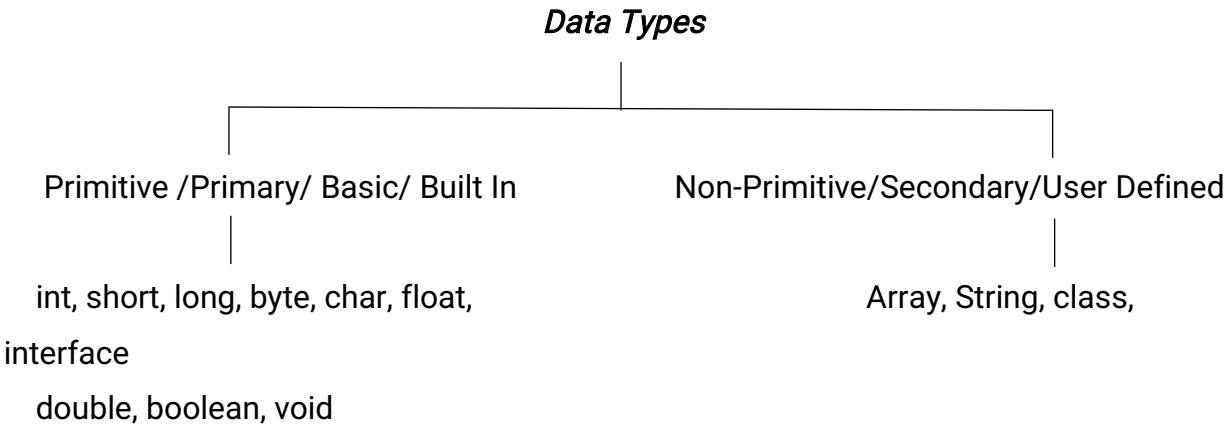
2. **Separators: -** Separators help define the structure of a program. In java there are few characters that are used as separators.

Symbol	Name	Purpose
( )	Parenthesis	Encloses arguments in method definitions and calling; adjusts precedence in arithmetic expressions; surrounds cast types and delimits test expressions in flow control statements
{ }	Braces	defines blocks of code and automatically initializes arrays
[ ]	Brackets / Subscripts	Declares array types and dereferences array values
;	Semicolon	Terminates statements
,	Comma	Separates successive identifiers in variable declarations; chains statements in the test, expression of a for loop

Edit with WPS Office

.	Period/ dot	Selects a field or method from an object; separates package names from sub-package and class names
:	Colon	Used after case labels in switch

7. Data Types: - Data types are defined as a set of possible values a variable can hold.



Differences between Primitive and Non-Primitive Data Types:-

Primitive	Non-Primitive
These are inbuilt data types. Java provides these data types.	These are created by programmers which make use of primitive type.
The size is fixed	The size depends on the number of member variables and their types.

Edit with WPS Office

It has only one memory location for a variable.	It has different memory location for variable.
To create a variable of primitive type 'new' keyword is not used.	To create a variable of non-primitive type 'new' keyword is used.

1. Integer: - Java defines four integer types: byte, short, int and long.

The size and ranges of integer types are:

Type	Size in Bytes	Size in Bits	Ranges
byte	1	8	-128 to 127
short	2	16	-32,768 to 32,767
int	4	32	-2^{31} to 2^{31} -1
long	8	64	-2^{63} to 2^{63} -1

2. Floating Point Types: - Floating point numbers are also known as real numbers. There are

two types of floating point types: float and double. The size and ranges of integer types are:

Type	Size in Bytes	Size in Bits	Ranges
float	4	32	$-3.4 * 10^{-38}$ to $3.4 * 10^{38}$
double	8	64	$-1.7 * 10^{-308}$ to $1.7 * 10^{308}$

3. Character: - In Java, the data type used to store characters is char. Java uses Unicode to

represent characters. Unicode defines a complete international character set that can represent

all the characters found in all human languages. In Java char is 16 bits(2 bytes) type and range

is from 0 to 65,536.

4. Boolean: - Java has a simple type, called boolean, for logical values. It can have only

Edit with WPS Office

two
possible values: true and false. The boolean type is declared using boolean keyword. The
size of boolean is 1 bit.

5. void: - void is used as return type of function. If a function does not return a value, its return
type must be void. void type variable cannot be declared in java.

6. String: - String is a sequence of characters, numbers or symbols. A String type is used to
declare a String variable. String is not a primitive type. Rather, String is defined as an object
of String class.
A quoted String constant can be assigned to String variable. Java supports String to allow it to
act similar to a primitive data type.

7. Array: - An array is a collection of variables of the variables of the same type that are
referenced by a common name.

8. Interface: - In Java, interface is a keyword that is used to create a special type of class that
cannot be instantiated.

9. Class: - Class is a collection of variables and methods. (Class= Data + Methods).

Variable

A variable is a named memory location, which holds a value. The value of variable can be change.

Declaration of Variable: - All variables must be declared before they can be used. The general
form of variable declaration is: data type <space> variable name;
The data type is one of Java's primitive data type or name of class or interface. Variable name is
an identifier name. Some examples of variable declaration is:

```
int   a;
float   b;
```


Edit with WPS Office

```
double    x;
char    m;
```

More than one variables can be declared of same type in one line, all the variables separated by
comma like:

```
int    p,q,r;
```

Initialization: - There are two ways to initialize a variable:

1. *Compile Time Initialization (Static Initialization)*
2. *Run Time Initialization (Dynamic Initialization)*

1. Compile Time Initialization (Static Initialization): - The variable can be initialized by assigning a value using assignment operator.

```
int    a;   // declaration of variable
x=10;  //compile time initialization
char   c1; // declaration of variable
c1= 'A';          //compile time initialization
```

Declaration and Initialization of variable in same time

```
int   y=10;       // declaration and initialization of variable
```

Declaration and Initialization of multiple variable in same time

```
int   a=10,b=30;
```

2. Run Time Initialization (Dynamic Initialization): - Java allows variables to be initialized dynamically, i.e. during execution of a program.

There three different methods of Run Time Initialization: -

1. Using Expression
2. Using Command Line Arguments
3. Using Library Classes (BufferedReader & Scanner)

1. ***Using Expression: -*** *The variable can be initialize using expression like:*

```
int   x=10; //compile time initialization
int   y=x+x; //run time initialization
```

2. ***Using Command Line Arguments: -*** *The variables can be initialize by passing as parameter of method in parenthesis.*

The scope and lifetime of variables: - Java allows variable to be defined inside any

Edit with WPS Office

block
enclosed in curly braces {}.
A block defines a scope. The scope determines what objects are visible to other parts of the
program and the lifetime of the objects. In Java the scopes are those defined by a class and
those defined by a method. The scope defined by a method begins with its braces. Variable
define inside a scope are not visible or accessible to code that is defined outside the scope.

For example:

```
class A
    {
        int   x;
        void main()
            {
            int y=30;
                x=39;
            }
      x=10;
      y=40; // Its not possible because y is local variable of function main
    }
```

Type Conversion or Casting

The process of converting one primitive type to another is called type conversion or casting.
The two types of conversion supported in Java are:

1. Implicit Conversion: - (Automatic Conversion or Widening Conversion or COERCION)
Implicit conversion is the conversion of one primitive type to the other compatible primitive
done by the Java compiler without the intervention of programmer.

Edit with WPS Office

Implicit Conversion can be done under two conditions:-

- Both types are compatible
- The destination types is larger than the source type.

For example: int x = 10;

double y = x;

Destination Source

2. Explicit Conversion: - (Narrowing Conversion)

Explicit conversion from one primitive type to another type using typecast operator '(data type)' in a java programs done using intervention of programmer.

Explicit Conversion can be done under two conditions:-

- Both types are incompatible
- The destination types is smaller than the source type.

For example:

i. int x = 70;

char y = (char) x;

Destination Type cast operator Source

ii. double a = 2.56;

int k = (int) a;

Destination Type cast operator Source

Operators

There are different types of operators in Java:

1. **Arithmetic Operators:-** They are used for various mathematical calculations, it can be

of following two types:

a. Binary Form

Operator	Use	Description

Edit with WPS Office

+	op1+op2	Addition of operands
-	op1-op2	Subtraction of operands
*	op1*op2	Multiplication of operands
/	op1/op2	Division of operands
%	op1%op2	Computes the remainder of dividing op1 by op2 (modulo division)

b. Unary Form

Operator		Use	Description
+		+op	Promotes op to int if it's a byte, short or char
-		-op	Arithmetically negates op
Prefix	++	++op	Increments op by 1,evaluates to the value of op after Change
	--	--op	Decrements op by 1,evaluates to the value of op after Change
Postfix	++	op++	Increments op by 1,evaluates to the value of op before Change
	--	op--	Decrements op by 1,evaluates to the value of op before Change

Prefix: - This operator works on the principle change before action, it means the value of

the variable changes before the action takes place.

Example: -

Edit with WPS Office

1. int n=10;
   int m= ++n * 2; // value of n is 11
   int x= n*2; // value of n is 11
2. int n=10;
   int m= --n * 2; // value of n is 9
   int x= n*2; // value of n is 9

Postfix: - This operator works on the principle change after action, it means the value of the variable changes after the action takes place.

Example: -

1. int n=10;
   int m= n++ * 2; // value of n is 10
   int x= n*2; // value of n is 11
2. int n=10;
   int m= n-- * 2; // value of n is 10
   int x= n*2; // value of n is 9

2. Relational Operators: - Relational operators in Java are used to compare two or more operands and return the result in a Boolean value.

Operator	Use	Description
<	op1<op2	op1 is less than op2
>	op1>op2	op1 is greater than op2
= =	op1= =op2	op1 is equal to op2
!=	op1!=op2	op1 is not equal to op2
<=	op1<=op2	op1 is less than or equal to op2
>=	op1>=op2	op1 is greater than equal to op2

3. Logical Operators: - Logical operators are used to connect existing relational expressions.

Operator	Use	Description

Edit with WPS Office

&&	exp1 && exp2	exp1 and exp2 are true then returns true
||	exp1 || exp2	exp1 or exp2 are true then returns true
!	!exp	If exp is true then returns false otherwise true

4. Assignment Operator: - Assignment operator (=) is used to assign the value to a variable.

Example: int a=10;

5. Ternary Operator: - Ternary operator (**?, :**) are used as conditional operator for make conditional statements.

var = expe **?** value1**:** value2**;**

6. Bitwise Operators: - Bitwise operators treat their operands as a sequence of 32 bits (zeroes and ones), rather than as decimal, hexadecimal, or octal numbers. For example, the decimal number nine has a binary representation of 1001. Bitwise operators perform their operations on such binary representations, but they return standard JavaScript numerical values.

Operator	Use	Description
&	op1&op2	Returns a one in each bit position for which the corresponding bits of both operands are ones.
|	op1 | op2	Returns a one in each bit position for which the corresponding bits of either or both operands are ones.
^	op1^op2	Returns a one in each bit position for which the corresponding bits of either but not both operands are ones.
~	~op	Inverts the bits of its operand.
<<	op1<<op2	Shifts op1 in binary representation op2 (< 32) bits to the left,

Edit with WPS Office

		shifting in zeroes from the right.
>>	op1>>op2	Shifts op1 in binary representation op2 (< 32) bits to the right, discarding bits shifted off.
>>>	op1>>>op2	Shifts op1 in binary representation op2 (< 32) bits to the right, discarding bits shifted off, and shifting in zeroes from the left.

7. Compound Assignment Operators (Shorthand Assignment Operators):- Java provides several shortcut assignment operators that allow to perform arithmetic and an assignment operation.

Operator	Use	Description
+=	op1+=op2	op1=op1+op2
-=	op1-=op2	op1=op1- op2
*=	op1*=op2	op1=op1*op2
/ =	op1/=op2	op1=op1/op2
%=	op1%=op2	op1=op1%op2

Operator Precedence: -

The following table is ordered from highest to lowest precedence.

Category	Operator	Associativity
Postfix	>() [] . (dot operator)	Left toright
Unary	>++ - - ! ~	Right to left

Edit with WPS Office

Multiplicative	>* /	Left to right
Additive	>+ -	Left to right
Shift	>>> >>> <<	Left to right
Relational	>> >= < <=	Left to right
Equality	>== !=	Left to right
Bitwise AND	>&	Left to right
Bitwise XOR	>^	Left to right
Bitwise OR	>|	Left to right
Logical AND	>&&	Left to right
Logical OR	>||	Left to right
Conditional	?:	Right to left
Assignment	>= += -= *= /= %= >>= <<= &= ^= |=	Right to left

Difference between Arithmetic Expression & Arithmetic Statement: -

Arithmetic Expression	Arithmetic Statement
An arithmetic expression may contain variables, constants and operators together to produce a meaningful result. Example: x + y , x + y * 10	If an arithmetic expression is assigned to a variable then it is known as arithmetical statement. Example: a = x + y , k = x + y * 10

Difference between Pure Expression & Mixed Expression: -

Pure Expression	Mixed Expression

In pure expression all the operands are same
type.
Example: int x = 10, y = 5, z;
z = x * y;
	In mixed expression the operands are different type or mixed.
	Example: int x = 10;
	double y = 5.8, z;
	z = x * y;

Edit with WPS Office

Function

A function is defined as a set of instructions given to the computer in order to perform a specific
task. Each function has a unique name or identity with which it is called within a program.
Function is a sub-program within a main program which process data and returns a value.
In Java functions are also referred as methods.

Parts of Functions:- A function has two parts:

1. **Function Header:-** It specifies the access specifier, return type, function name and the
   parameter list of the of the function. It's also known as function signature.

2. **Function Body:-** The function body is the remaining part of the function after the header
   enclosed by a pair of braces.
   Any set of declarations and statements between a pair of braces is known as a block.

   accessspecifier returntype functionname (parameter list/ argument list) / /
   Function Header

   Function Body
   {
   variables;
   statements;
   …….
   …….
   return statement;
   }

   Example:
   public void main(int x)

Edit with WPS Office

```
{
        int x,y,z;
        x=10;
        y=6;
        z=x+y;
}
```

1. Access Specifier: - It is public, private, protected or default. If programmers don't use any
access specifier (public, private, protected) then function work in default scope.

2. Return Type: - It specifies the type of data returned by the method. It can be any valid data
type. If a function doesn't returns a value, its return type must be void.

3. Function Name: - Function name shouldn't be same as the class name.

4. Parameter List: - Parameter lists are variables that receives the value of the arguments
passed to the function when it's called. It can be empty if there are no parameters.

Function Declaration: - Function declaration is function header with semicolon and parameters
are only with data type excluding variable name.

Function Calling: - Function calling is call the function to perform task. Function call by name
and parameters of functions. If function is static then it's called by class name otherwise called
by object.

Function Definition: - Function Definition is to define the function with function header and
body of the function.

Edit with WPS Office

General Form of class: -

```
class  ClassName
{   // start of class
        instance variables;
        static variables;
        member methods()
        {
          body of the method
        }
}// end of class
```

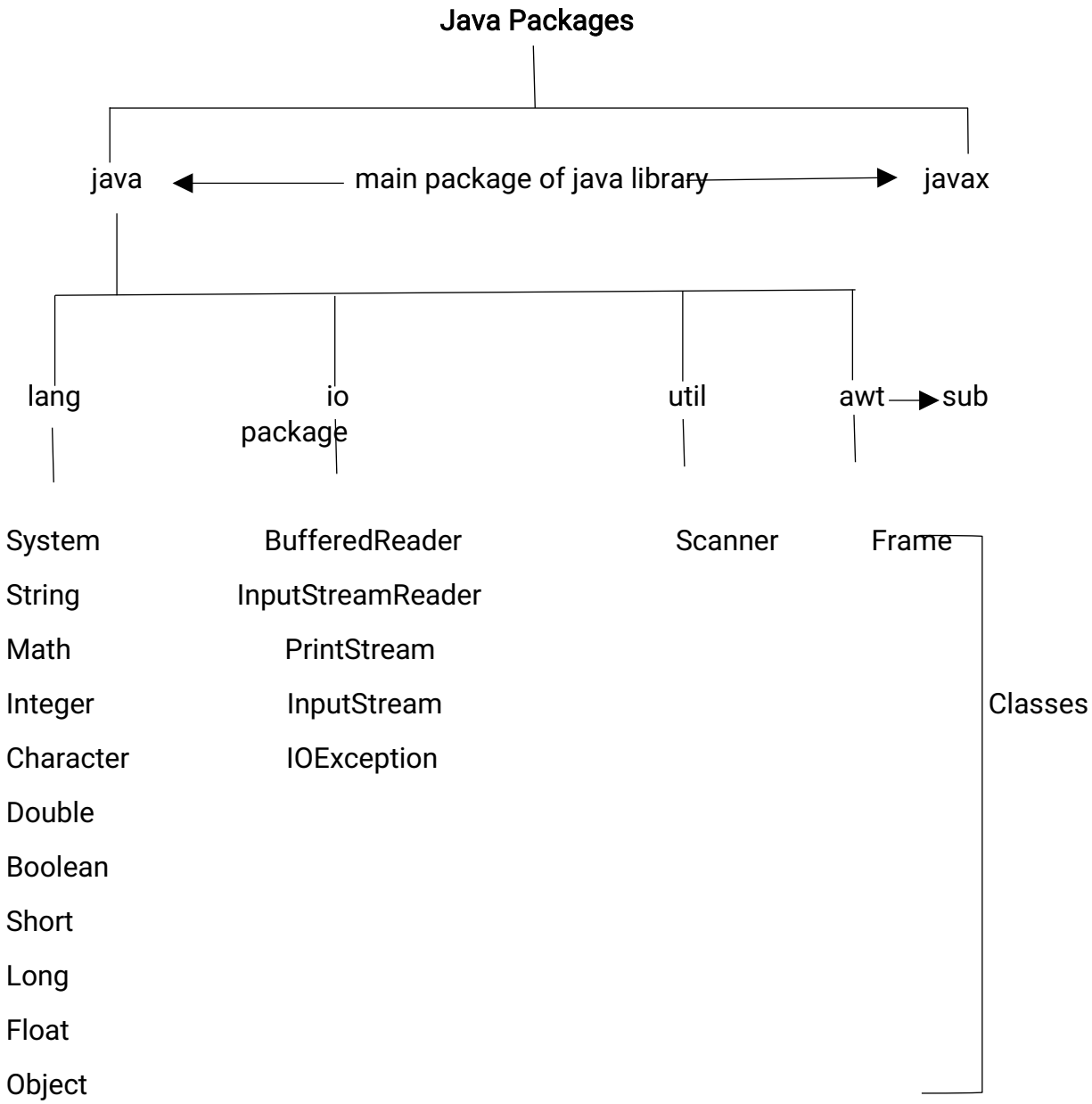
1. class is a keyword that identifies class declaration
2. ClassName is name of class. It is user defined name (identifier).
3. The method and variables defined within a class are called members of the class.

Package: - Package in Java is basically a collection of classes. Each package includes related
built in functions which may be used while developing programming logic or programs. These packages are needed to be called using the keyword 'import'.
Each Java programs automatically imports two entire packages the **java.lang** package and
the current package by default.
There are two ways to import the packages:

- To import all the classes contained in a particular package, use the import statement
  with the asterisk (*) wildcard character. For example: **import java.io.*;**

- To import the single class of particular package, use the import statement with the
  package and class name. For example: **import java.util.Scanner;**

Edit with WPS Office



System class: - All Java programs automatically import **java.lang** package. This defines a class

called System, which encapsulate several aspects of run time environment.

It also contains three predefined stream variables in, out and err. System.out and System.err are

object of PrintStream class. System.in is object of InputStream class.

These are byte streams. They are used to read and write characters from and to the console.

Output statement: -

Edit with WPS Office

1. System.out.print( ): - The cursor remains on the same line after displaying the result.

2. System.out.println( ): - The cursor skips the line and passes to the next line after displaying the required result.

These statements are used to get the output of the program or to display message on the screen.

1. System.out.print("Welcome in Java"); - This statement will print the comment in java on the screen.

2. int a=10;

   System.out.print(a); - This statement will print the value of variable a.

In case of a message along with a variable:

System.out.print("Value of a=" + a);

When a message is to be displayed along with a variable then they are to be separated with '+'.

Comments: - A comment is a remark from the programmer to explain some aspect of the code

to someone who is reading the source code. Comments lines are ignored by the compiler.

There are three types of comments defined in Java.

1. **Single Line Comments: -** Comments follow the two forward slash // symbol.

   Example: // start of class

2. **Multiline Comments: -** They are used for comments spread over many lines. They

   begin with forward slash and asterisk /* and end with asterisk with forward slash */.

   Example: /* multiline comments are

Edit with WPS Office

spread over lines */

3. **Documentation Comments: -** They are used to produce an HTML file that documents

   our program. They begin with forward slash two asterisk /** and end with asterisk with

   forward slash */.

Write a program to print your name.

Steps:

1. start
2. declare class Program using class keyword
3. opening brace { for start of class body
4. create main ( ) method with void return type
5. print your name using System.out.print() method
6. closing brace } for end of class body
7. end

Program:

```
class Program
{       // start of class
        void main( )   // main() method
        {              // start of main() method
                System.out.print("My Name is XYZ");
        }              // end of main() method
}       // end of class
```


Edit with WPS Office

Write a program to print sum of two numbers.

Method 1: ***using compile time initialization***

Steps:

1. start
2. declare class Program using class keyword
3. opening brace { for start of class body
4. create main ( ) method with void return type
5. declare 3 integer type variables
6. initialize values to 2 variables
7. write statement to store sum of 2 initialized variable into third variable
8. print value of third variable using System.out.print() method
9. closing brace } for end of class body
10. end

Program:

```
class Program
{       // start of class
        void main( )   // main() method
        {              // start of main() method
                int x,y,z;        // declaration of variables
                x=10;             // compile time initialization
                y=20;             // compile time initialization
                z= x + y;         // store sum of x & y into z
                System.out.print(" Sum= " + z);
        }              // end of main() method
}       // end of class
```


Edit with WPS Office

Method 2: *using run time initialization* (command line arguments)

Steps:

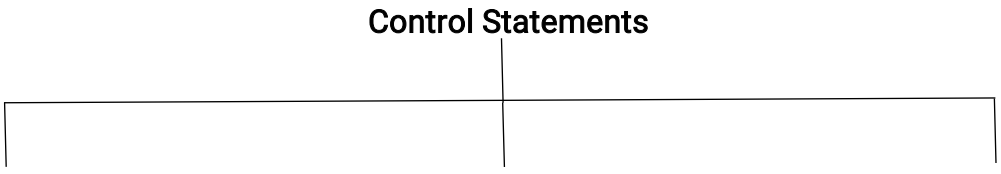
1. start
2. declare class Program using class keyword
3. opening brace { for start of class body
4. create main ( ) method with void return type and two integer type arguments
5. declare 1 integer type variable to store sum
6. write statement to store sum of 2 initialized variable into third variable
7. print value of third variable using System.out.print() method
8. closing brace } for end of class body
9. end

Program:

```
class Program                          Command Line Arguments for initialization
{       // start of class              only in BlueJ
        void main( int x, int y )      // main() method with x & y to initialize value
        {               // start of main() method
                int z;  // declaration of variable
                z= x + y;       // store sum of x & y into z
                System.out.print(" Sum= " + z);
        }               // end of main() method
}       // end of class
```

Control Statements

Control statements are used to control the flow of execution of program.



Edit with WPS Office

Conditional | Selection | Looping / Iteration

switch statement | while loop

Ternary | if statements | do - while loop

Operator | if – else statements | for loop

nested if – else statements

if – else if statements

Conditional Control Statements: - A conditional control statement allows a set of statements

to be executed only when a given condition is true .

Java supports the following types of conditional control statements:

1. if statement
2. if – else statements
3. nested if – else statements
4. if – else if statements
5. Ternary Operator

1. **if statements: -** The general form of the if statement is-

```
if ( condition)

statement 1;

statement 2;
```

if is a keyword, condition is any relational expression / function that returns a Boolean value.

Edit with WPS Office

If condition is true then statement 1 will be executed and after that statement 2, otherwise only

statement 2 will be executed.

```
if ( condition)
{
statement 1;
statement 2;
}
statement 3;
```

body of if

If condition is true then statement 1 & statement 2 will be executed and after that statement 3,

otherwise only statement 3 will be executed.

Multiple statements placed within the curly braces form a compound statement.

2. **if else statements: -** The general form of the if –else statement is-

```
if ( condition)

statement 1;

else
statement 2;

statement 3;
```

if is a keyword, condition is any relational expression / function that returns a Boolean value.

If condition is true then statement 1 will be executed and after that statement 3, otherwise

statement 2 and after that statement 3 will be executed.

Edit with WPS Office

```
if ( condition)
{
statement 1;      body of if
statement 2;
}

else
{
statement 3;      body of else
statement 4;
}

statement 5;
```

If condition is true then statement 1 & 2 will be executed and after that statement 5, otherwise

statement 3 & 4 and after that statement 5 will be executed.

3. **Nested if – else statements : -** The general form of the nested if –else statement is-

```
if ( condition 1)
{
if ( condition 2)
statement 1;          body of first if
else
statement 2;
}

else
{
```


Edit with WPS Office

```
statement 3;
}
statement 4;
```

If condition 1 is true then check condition 2 then statement 1 will be executed and after that

statement 4, otherwise statement 3 will be executed and after that statement 4,

4. **if – else if statements : -** The general form of the if –else if statement is-

```
if ( condition 1)
statement 1;
else if ( condition 2)
statement 2;
else if ( condition 3)
statement 3;
.....
.....
.....
else if ( condition n)
statement n;
else
statement 5;

statement 6;
```

If statements are executed from top down. As soon as one of the conditions controlling the if is

true, the statement associated with that if is executed, and rest of the if statements are bypassed.

if none of the conditions are true then the final else abatement will be executed. If final else

statement is not present, then no action is taken.

Edit with WPS Office

Dangling else problem: - The ambiguity that arises in a nested if statement due to presence of

more number of if than the number of else clause is called dangling else problem. The confusion can be removed using '{}'.

Example:

```
if(condition1)
{       // start of if
        if(condition 2)
        statement 1;
}       // end of if
else
        statement 2;
```

5. **Ternary Operator (Conditional Operator): -** Java includes a special ternary operator

   that can be used to replace certain types of if - else if statements. It is called ternary

   operator, as it requires three operands to work on. It is represented by **?** and **:** symbols.

   The general form of the ternary operator is:

   variable = condition **?** expression1**:** expression2**;**

   Here, condition is any relational expression, if condition is true, then expression 1 is

   evaluated otherwise expression 2 is evaluated.

Difference between Ternary Operator and if-else Statement:-

Ternary Operator	if-else statement

Edit with WPS Office

It can return only one value.	It can have more than one statement in a block.
It makes the program more concise and simple.	Construct much space.
In nested form it is more complex.	In nested form it is not complex.

<u>switch statement: -</u> The switch statement is Java's multi way branch selection statement.

The general form of a switch statement is:

```
switch ( expression / variable )
{                // start of switch
    case 1:
        Statement 1;
        break;

    case 2:
        Statement 2;
        break;
    ::::::::::::::::::::::::::::::::::        Body of switch
    ::::::::::::::::::::::::::::::::::
    ::::::::::::::::::::::::::::::::::
    case n:
```


Edit with WPS Office

```
                Statement n;
                break;
            default:
                Statement x;
        }       // end of switch
```

The value of the expression / variable is compared with each of the literal values in the

case statements. If match is found, the statement following the case is executed. If none

of the constant matches the value of the expression, then the default statement is

executed. The default statement is optional. If no match is found and default statement is

not present, no further action is taken.

The break statement is used to terminate the statement sequence.

If break statement is not present, execution will continue till the next case statement.

The next cases get executed until a break is encountered. Such a condition is called

fall through condition.

The variable which is passed in switch to compared with each of literal values in the

case statements is called as **Control Variable**.

Difference between if statement and switch statement: -

if statement	switch statement
Comparison can be done between a variable and range of values.	Comparison is done between variable and a value.
Logical expression can be used.	Logical condition cannot be used.
It can be used to compare float values and String values also.	It can compare integers, boolean and Character only.

Edit with WPS Office

One can compare value of one variable with other variable.	In this variable value can be compared with constant value.

Iteration / Looping: - Java provides such iteration statements like **for, while,** and **do-while**.

A loop repeatedly executes the same set of instructions until a termination condition is met.

There are three basic types of loops: Definite, Indefinite & Infinite loop.

i. **Definite Loop:** It is a loop which performs up to given limit.

Example: 5 times, 10 times, n times.

ii. **Indefinite Loop:** It is a loop which performs as long as the user wants.

Example: how many people will come to see the fair.

iii. **Infinite Loop:** It is a loop which never terminates.

A loop usually includes 4 steps:

- Initialization of conditional variable
- Executable statements
- Condition or test expression
- Updation of the conditional variable

Loops are classified as:

1. **Entry Control Loops:** In these loop condition or test expression is evaluated before

   each iteration.

2. **Exit Control Loops:** In these loops the condition or test expression is evaluated after

   each iteration.

Edit with WPS Office

while loop : It repeats a statement or block of statements, while the condition is true. The

general form of while loop is:

```
initialization of conditional variable;
while (condition or text expression)
{       // start of loop
        executable statement;
        updation of conditional variable;
}       // end of loop
```

Body of loop

do-while loop : It repeats a statement or block of statements, while the condition is true. The

general form of do-while loop is:

```
initialization of conditional variable;
do
{       // start of loop
        executable statement;
        updation of conditional variable;
} while (condition or text expression);    // end of loop
```

Body of loop

Difference between while and do-while loop : -

while loop	do-while loop
It checks for the condition first and then executes the block of statements.	It executes the block of statements once and then checks for the condition.
It is entry controlled loop.	It is exit controlled loop.

Edit with WPS Office

It does not execute if condition is not true.	It executes at least once even if the condition is false.

for loop : It repeats a statement or block of statements, while the condition is true. The general form of for loop is:

```
for(initialization ;   condition or text expression ;   updation )
{       // start of loop
        executable statement;          Body of loop
}   // end of loop
```

Variations in for loop :-

1.
```
for(initialization ;   condition or text expression ;  )
{       // start of loop
        executable statement;
        updation;
}   // end of loop
```

2.
```
initialization ;
for(  condition or text expression ;  )
{       // start of loop
        executable statement;
        updation;
}   // end of loop
```

3. initialization ;

Edit with WPS Office

```
for(  condition or text expression ;          updation )
{       // start of loop
        executable statement;
}   // end of loop
```

4. We can create infinite loop , i.e. never terminates, if we leave all three parts of the for loop empty.

   For example:

```
for( ; ; )
{       // start of loop
        executable statement;
}   // end of loop
```

Nested Loops: - Java allows loops to be nested. That is one loop can be inside another loop.

For each outer loop, the inner loop executes n number of times until the condition of inner

loop is true.

Example:-

```
for( i=1; i<=3; i++)
{
for( j=i; j<=3; j++)
{
        System.out.print("*");
}
System.out.println(" ");
}
```


Edit with WPS Office

Output: ***

**

*

Jump Statements: - Java supports jump statements: **break, continue** and **return.**

Jump statements are used to transfer control to another part of program.

1. **break** :- By using break; we can force immediate termination of a loop and terminate

   the case of switch.

   Example :-

```
for ( i=1; i<=10; i++)
{
    if( i < 5 )
    break;
    System.out.print( i );
}
```

   Output :- 1 2 3 4

2. **continue** :- By using continue; we can force early iteration of a loop.

   Example :-

```
for ( i=1; i<=10; i++)
```


Edit with WPS Office

```
{
        if( i % 2 = = 0 )
        continue;
        System.out.print( i );
}
```

Output :- 1 3 5 7 9

3. **return :-** By using return, we can return the value of function.

Example :-

```
int main( )
{

        return 0;
}
```

Math Class Functions

The **Math** class contains all the floating-point functions that are used for geometry and trigonometry, as
well as several general-purpose methods. **Math** defines two **double** constants: **E** (approximately 2.72)
and **PI** (approximately 3.14).

Transcendental Functions:-

The following three methods accept a **double** parameter for an angle in radians and return the result of
their respective transcendental function:

Edit with WPS Office

Method	Description
static double sin(double *arg*)	Returns the sine of the angle specified by *arg* in radians.
static double cos(double *arg*)	Returns the cosine of the angle specified by arg in radians.
static double tan(double *arg*)	Returns the tangent of the angle specified by arg in radians.

The next methods take as a parameter the result of a transcendental function and return, in radians, the
angle that would produce that result. They are the inverse of their non-arc companions.

Method	Description
static double asin(double *arg*)	Returns the angle whose sine is specified by arg.
static double acos(double *arg*)	Returns the angle whose cosine is specified by arg.
static double atan(double *arg*)	Returns the angle whose tangent is specified by arg.
static double atan2(double *x*, double *y*)	Returns the angle whose tangent is x/y.

Exponential Functions:-

Math defines the following exponential methods:

Method	Description
static double exp(double *arg*)	Returns e to the arg.
static double log(double *arg*)	Returns the natural logarithm of arg.
static double pow(double y, double x)	Returns y raised to the x; for example, pow(2,3) returns 8.0.
static double sqrt(double *arg*)	Returns the square root of arg; for example, sqrt(25) returns 5.0.
static double cbrt(double *arg*)	Returns the cube root of arg; for example, cbrt(27) returns 3.0.

Rounding Functions:-

The **Math** class defines several methods that provide various types of rounding operations. They are

Edit with WPS Office

shown in Table.

Table The Rounding Methods Defined by Math.

Method	Description
static int abs(int *arg*)	Returns the absolute value of *arg*.
static long abs(long *arg*)	Returns the absolute value of *arg*.
static float abs(float *arg*)	Returns the absolute value of *arg*.
static double abs(double *arg*)	Returns the absolute value of *arg*.
Example : double x = Math.abs(-5); Absolute value of x is : 5.0	
static double ceil(double *arg*)	Returns the smallest whole number greater than or equal to *arg*.

Example :
double x = Math.ceil(5.3);
Absolute value of x is : 6.0;
double y = Math.ceil(5.9);
Absolute value of y is : 6.0;	

static double floor(double *arg*)	Returns the largest whole number less than or equal to *arg*.

Example :
double x = Math.floor(5.3);
Absolute value of x is : 5.0;
double y = Math.floor(5.9);
Absolute value of y is : 5.0;	

static int max(int *x*, int *y*)	Returns the maximum of *x* and *y*.
static long max(long *x*, long *y*)	Returns the maximum of *x* and *y*.
static float max(float *x*, float *y*)	Returns the maximum of *x* and *y*.

Edit with WPS Office

static double max(double *x*,double *y*)	Returns the maximum of *x* and *y*.
Example : double x = Math.max(5,3); Absolute value of x is : 5.0;	
static int min(int *x*, int *y*)	Returns the minimum of *x* and *y*
static long min(long *x*, long *y*)	Returns the minimum of *x* and *y*.
static float min(float *x*, float *y*)	Returns the minimum of *x* and *y*.
static double min(double *x*,double *y*)	Returns the minimum of *x* and *y*.
Example : double x = Math.min(5,3); Absolute value of x is : 3.0;	
static double rint(double *arg*)	Returns the integer nearest in value to *arg*.

Example :
double x = Math.rint(5.3);
Absolute value of x is : 5.0;
double y = Math.rint(5.9);
Absolute value of y is : 6.0;	

static int round(float *arg*)	Returns *arg* rounded up to the nearest int.
static long round(double *arg*)	Returns *arg* rounded up to the nearest long.

Example :
double x = Math.round(5.3);
Absolute value of x is : 5.0;
double y = Math.round(5.9);
Absolute value of y is : 6.0;	

static double random()	**Returns any value between 0 to 1 randomly.**

Edit with WPS Office

Wrapper Class

Normally, when we work with Numbers or Character, we use primitive data types such as byte, int, long, double,char etc.

However in development we come across situations were we need to use objects instead of primitive data types. In-order to achieve this Java provides wrapper classes for each primitive data type.

These classes are present in java.lang package.

Data Type	Wrapper Class
int	Integer
long	Long
short	Short
byte	Byte
float	Float
double	Double
char	Character

parse…() function: parse() functions are used to convert string numeric values in primitive type. They receives the string and return it as related wrapper class data type (only numeric). Example:

1. int a=Integer.parseInt("5");

2.double x=Double.parseDouble("1.5");

3. float y=Float.parseFloat("2.3");

4. long k=Long.parseLong("10");

valueOf() function: vlaueOf() functions are used to convert string numeric values in object type. They receives the string and return it as related wrapper class object (only numeric). Example:

1. Integer a=Integer.valueOf("5");

   int a=Integer.valueOf("5");

2.Double x=Double. valueOf ("1.5");

  double x=Double. valueOf ("1.5");

3. Float y=Float. valueOf ("2.3");

Edit with WPS Office

float y=Float. valueOf (“2.3”);

4. Long k=Long. valueOf (“10”);

long k=Long. valueOf (“10”);

toString() function: toString() functions are used to convert numeric values in string type

Example:

1. Strting a=Integer.toString(5);

2.String x=Double. toString (1.5);

3. String y=Float. toString (2.3);

4. String k=Long. toString(10);

toBinaryString() function: toBinaryString() function is used to convert integer values in binary type

toOctalString() function: toOctalString() function is used to convert integer values in octal type

toHexString() function: toHexString() function is used to convert integer values in hexadecimal type

Example:

1. Strting a=Integer.toBinaryString(5);

2. Strting a=Integer.toOctalString(15);

3. Strting a=Integer.toHexString(51);

Character Methods:

Here is the list of the important instance methods that all the subclasses of the Character

class implement:

1) **boolean isLetter(char)** Determines whether the specified char value is a letter.

   Example: boolean b=Character.isLetter(‘5’);

   value of b is false

   boolean b=Character.isLetter(‘D’);

   value of b is true

2) **boolean isDigit(char)** Determines whether the specified char value is a digit.

Edit with WPS Office

Example: boolean b=Character.isDigit('5');

value of b is true

boolean b=Character.isDigit('D');

value of b is false

3) **boolean isWhitespace(char)** Determines whether the specified char value is white space.

Example: boolean b=Character.isWhitespace (' ');

value of b is true

boolean b=Character.isWhitespace ('D');

value of b is false

4) **boolean isUpperCase(char)** Determines whether the specified char value is uppercase.

Example: boolean b=Character.isUpperCase ('D');

value of b is true

boolean b=Character.isUpperCase ('d');

value of b is false

5) **boolean isLowerCase(char)** Determines whether the specified char value is lowercase.

Example: boolean b=Character.isLowerCase ('d');

value of b is true

boolean b=Character.isLowerCase ('D');

value of b is false

6) **char toUpperCase(char)** Returns the uppercase form of the specified char value.

Example: char b=Character.toUpperCase ('D');

value of b is 'D'

char b=Character.toUpperCase ('d');

value of b is 'D'

Edit with WPS Office

7) char toLowerCase(char) Returns the lowercase form of the specified char value.

Example: char b=Character.toLowerCase ('D');

value of b is 'd'

char b=Character.toLowerCase ('d');

value of b is 'd'

8) String toString(char)Returns a String object representing the specified character value that is, a one-character string.

Example: String b=Character.toString ('D');

value of b is "D"

BufferedReader Class:

Java provides read() and readLine() functions to receives the input from keyboard on terminal (output screen) for program.

These functions are available in BufferedReader class as non-static (instance) member of class.

So called by object of BufferedReader class.

There are two methods to create object of BufferedReader class:

1. InputStreamReader isr = new InputStreamReader(System.in);

BufferedReader br = new BufferedReader(isr);

2. BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

read() and readLine() functions raised an exception for that we use the **throws IOException**

with main method in program.

BufferdReader, InputStreamReader and IOException are available in java.io package, So we need to import this package in our program like:

import java.io.*;

readLine() function receive the values in string form , So we use wrapper class to convert it in related data type value.

Edit with WPS Office

Example:

```
import   java.io.*;
class  Example
{
static void main() throws IOException
{
BufferedReader  br = new BufferedReader(new  InputStreamReader(System.in));
System.out.print(“Enter number= ”);
int a=Integer.parseInt(br.readLine());
System.out.print(“value of a=”+a);
}
}
```

read() function receives the value in byte form it is used input byte or char type values.

Example:

```
import   java.io.*;
class  Example
{
static void main() throws IOException
{
BufferedReader  br = new BufferedReader(new  InputStreamReader(System.in));
System.out.print(“Enter Character= ”);
    char x=(char)br.read();
System.out.print(“value of x=”+x);
}
}
```

The read() function can be also called by System class using object of InputStream class

System.in like:

```
 char x=(char)System.in.read();
```


Edit with WPS Office

Scanner Class

Java Scanner class comes under the java.util package. Java has various ways to read input from the keyboard, the java.util.Scanner class is one of them.

It is used for obtaining the input of the primitive types like int, double etc. and strings. It is the easiest way to read input in a Java program, though not very efficient if you want an input method for scenarios where time is a constraint like in competitive programming.

- To create an object of Scanner class, we usually pass the predefined object System.in, which represents the standard input stream. We may pass an object of class File if we want to read input from a file.

Example:

```
Scanner sn=new Scanner(System.in);  // to read input from keyboard
Scanner sn=new Scanner("exam.txt");  // to read input from file
```

- To read numerical values of a certain data type XYZ, the function to use is nextXYZ(). For example, to read a value of type short, we can use nextShort()
- To read strings, we use nextLine().
- To read a single character, we use next().charAt(0). next() function returns the next token/word in the input as a string and charAt(0) funtion returns the first

Edit with WPS Office

character in that string.

The Java Scanner class breaks the input into tokens using a delimiter that is whitespace by default. It provides many methods to read and parse various primitive values.

Java Scanner class is widely used to parse text for string and primitive types using a regular expression.

Java Scanner class extends Object class and implements Iterator and Closeable interfaces.

SN	Return Type	Method	Description
1)	boolean	hasNext()	It returns true if this scanner has another token in its input.
2)	boolean	hasNextBigDecimal()	It is used to check if the next token in this scanner's input can be interpreted as a BigDecimal using the nextBigDecimal() method or not.
3)	boolean	hasNextBigInteger()	It is used to check if the next token in this scanner's input can be interpreted as a BigDecimal using the nextBigDecimal() method or not.
4)	boolean	hasNextBoolean() hasNextByte() hasNextDouble() hasNextFloat() hasNextInt() hasNextLine() hasNextLong() hasNextShort()	It is used to check if the next token in this scanner's input can be interpreted as a Boolean using the nextBoolean(), Byte using the hasNextByte(),Double using the hasNextDouble(),Float using the nextFloat(), Long using the nextLong(),int using the nextInt(),Short using the nextShort() method or not.
5)	IOException	ioException()	It is used to get the IOException last thrown

Edit with WPS Office

			by this Scanner's readable.
6)	Locale	locale()	It is used to get a Locale of the Scanner class.
7)	String	next()	It is used to get the next complete token from the scanner which is in use.
8)	BigDecimal	nextBigDecimal()	It scans the next token of the input as a BigDecimal.
9)	BigInteger	nextBigInteger()	It scans the next token of the input as a BigInteger.
10)	boolean	nextBoolean()	It scans the next token of the input into a boolean value and returns that value.
11)	byte	nextByte()	It scans the next token of the input as a byte.
12)	double	nextDouble()	It scans the next token of the input as a double.
13)	float	nextFloat()	It scans the next token of the input as a float.
14)	int	nextInt()	It scans the next token of the input as an Int.
15)	String	nextLine()	It is used to get the input string that was skipped of the Scanner object.
16)	long	nextLong()	It scans the next token of the input as a long.
17)	short	nextShort()	It scans the next token of the input as a short.
18)	int	radix()	It is used to get the default radix of the Scanner use.
19)	void	remove()	It is used when remove operation is not supported

Edit with WPS Office

			by this implementation of Iterator.
20)	Scanner	reset()	It is used to reset the Scanner which is in use.
21)	Scanner	skip()	It skips input that matches the specified pattern, ignoring delimiters
22)	String	toString()	It is used to get the string representation of Scanner using.

Edit with WPS Office