



UNIT – I

Overview of Java Language

Introduction :- Java एक General purpose Object Oriented Programming (Opps) language हैं । इसे James Gosling, Patrick Mounghoth, Chris Warth, Ed Frack, Mark Sheridan के द्वारा Sun Micro System, USA में 1991 में Develop किया गया था । यह Java का पहला Version था जिसे एक B-Version के रूप में use किया गया था । Java के इस Version को developing में 18th month का समय लगा । Java language को starting में Cak language के रूप में जाना गया । लेकिन बाद में इसे "Java" के Name से जाना गया । Java का Complete Version सबसे पहले January 1996 में Office announce किया गया था ।

Java को वास्तव में Internet Programming के लिए नहीं बनाया गया था । बल्कि Java को consumer electronic devices (Home Application) के लिए प्रयुक्त होने वाले SW development के लिए design किया गया ।

Java development की team में यह पाया गया कि पहले से स्थित language जैसे बाएँ C++ में reliability portability के संबंध में कई कमियाँ होती हैं । इसलिए यद्यपि Java development team में Java language का model C, C++ के base पर बनाया लेकिन इनमें से C++ के उन feature को हटा दिया जो problem कर सकते थे । और इन्हीं सब कारणों से व एक simple reliable portable powerful language बन गई । Java का सबसे महत्वपूर्ण feature यह होता है कि यह किसी particular H/W या OS पर depend नहीं होती हैं । Java में develop किया गया program किसी भी System पर execute कराया जा सकता है । Java को एक revolutionary technology भी कहा जाता है ।

D;ksafd bls igys dksbZ Hkh ,slh language develop ugha dh xbZ Fkh fd fdllh Hkh H/W Operating System ij py lds A

Some Facts about Java

- 1- Java को सबसे पहले Oak name से develop किया गया था । इसका name बदलकर 'Java' कर दिया गया ।
- 2- Java का first version 18 month में develop किया गया था । जो कि Java का Beta Version था, 1991 में complete version 1996 में officially rectare किया गया था ।
- 3- 1992 में Sun micro System की एक development team green project team सबसे पहले यह demo दिया गया था कि touch sensitive screen का use करते हुए different name application का use किया जा सकता है ।
- 4- Java में develop किए गए applets को execute करने के लिए 1994 में Sun micro System द्वारा एक web browser को develop किया गया जिसे "Hot Java" नाम से जाना गया ।



Characteristics at Java:-

1. Simple:- Java की command का architecture C/C++ से बनाया गया है । इसमें C/C++ के कुछ feature को हटा दिये गये हैं । जो internet पर security के लिए अनुकूल नहीं थे । उदा. pointer इस प्रकार Java C/C++ के structure का use करते हुए बनाई गई है । जिसके कारण यह programmer के लिए एक familiar language है । इसलिए Java को a++ का simplified version भी कहा जाता है ।

2. Secure:- यद्यपि Java C/C++ के command structure का use करता है । परंतु इसमें से C++ के अनेक feature हटा दिए गये हैं जो कि security के लिए अनुकूल नहीं थे जैसे :- use of pointer इसके साथ ही Java में बेहतर Encryption की method का use किया जा सकता है । जिसके कारण Java एक secure language है । Internet में Java का use करते हुए विभिन्न security provide की जा सकती है ।

3. Portable:- Java एक portable language है । इसलिए इसमें बनाया गया program किसी भी H/W या Operating System पर चलाया जा सकता है । Java दो कारणों से portable है ।

1. Java compiler source file को compile कर byte code instruction Generate करता है जिसके बाद में किसी भी H/W या O.S. पर चलाया जा सकता है ।
2. Java में Binary data type का size किसी H/W machine पर depend नहीं करता ।

4. Object oriented:- Java एक true object oriented language है । Java में बनाए गए किसी भी program में पूरा program code एक class में और एक object के रूप में होता है । Java के साथ library file के रूप में अनेक classes प्राप्त होती है । जिन्हें inheritance के द्वारा अपने program में use कर सकते हैं । Java में object को module बनाना easy होता है ।

5. Robust:- Java एक Robust language है । यह Reliable coding के लिए several sayeguaral provide करती है । Java में data type के लिए compile time व Runtime पर Stick checking की जाती है । Java में memory management से संबंधित problem को दूर करने के लिए automatic garbage collection होता है । Java में मुख्य feature expiation handling का हाता है । जिसके कारण programmers exequiction के दौरान होने वाले संभावित error को टाला जा सकता है । इस प्रकार System को crash होने से बचाया जा सकता है ।

6. Multithreaded:- Java multithreaded programming को support करता है । यहाँ multithreaded means multiple task को simultaneously handle करना है । Java में इसे program बनाए जा सकते हैं । जो एक साथ कई process कर सकते हैं । अर्थात् किसी other program के Run करने के लिए पहले program के end होने का wait नहीं करना पड़ेगा । But दोनों program simultaneously Run होगी । यह feature कई गुना बढ़ा देता है ।



7. Architecture Natural :- Java एक ऐसी language है जिसको किसी भी H/W या O.S. पर Run किये जा सकते हैं। Java में बनाए गये program को easily बनाया जा सकता है। इस पर O.S. के change होने upgrade होने से processor & system resource का Java के program पर कोई प्रभाव नहीं पड़ता है।

8. Interpreter :- Java में program का compilation दो step में होता है 1st step में Java के source को compile किया जाता है। जो इसे byte code में change कर देता है। बाद में virtual machines के द्वारा विभिन्न machine के हिसाब से Interpreter कर दिया जाता है। इस प्रकार Java को एक compile Interpreted language कहा जाता है।

9. High Performance :- Java language में बने program की performance other interpreted language program की अपेक्षा अधिक होती है। इन program की performance अधिक होने के कारण byte code का use किया जाता है। Java के Architecture इस प्रकार design किया गया है कि यह Runtime पर System की Overhead को काफी कम कर देता है। इसके साथ ही Java के program में multithreading का use इसकी execution की speed को काफी बढ़ा देती है।

10. Distributed: - Java एक distributed language के रूप में develop किया गया है। जिसका use Network से related application को develop करने में किया जा सकता है। Java में यह ability होती है कि यह Network में data या programmes को share कर सकता है। Java application internet या स्थित remote object को local machine पर स्थित object की तरह access कर सकते हैं।

11. Dynamic :- Java एक dynamic language है। Java में यह capability होती है। कि यह new class libraries, method & object को dynamically link कर सकता है।

Java & Internet

Java Internet से strongly associate होता है क्योंकि Java में develop किया गया पहला program not Java या जो कि एक web browser या जिसका मुख्य use internet पर applets को execute करने के लिए किया जाता था। Internet के use applets program बनाने के लिए Java का use कर सकते थे और उन्हें local machine पर किसी Java enabled browsers की help से execute करा सकते थे। इस Java enabled का use Internet के द्वारा किन्हीं भी स्थिति किसी भी comp से applet को down compile पर Run किया जा सकता है।

www एक open ended information retrieval है जिसे internet के distributed environment के design किया गया है। इस System में web pages होते हैं जो information and control दोनों provide करते हैं। web System के open ended होने के कारण internet पर information



की किसी भी direction से search किया जा सकता है यह सब HTML के कारण संभव हुआ है । web pages में HTML text होते हैं । जिसके द्वारा internet पर विभिन्न प्रकार की information को search किया जा सकता है । Java मुख्यतः distributed environment से संबंधित programming करने के लिए design किया गया था । **जैसे :-** internet क्योंकि web and Java same को concept को share करते हैं । इसलिए Java को easily web System से incorporate किया जा सकता है । Java के development के पहले www state image और text के display के साथ Java के incorporation के साथ web pages के द्वारा विभिन्न animation special effects का support किया जाने लगा । web pages के साथ Java का use करने से web pages काफी interactive एवं dynamic बन गये । दूसरी तरह internet के use के साथ Java के programs को internet के use के साथ Java के programs का internet पर किसी other person के computer पर चलाना संभव हो गया ।

H/W & S/W Requirement

Java को Current में लगभग सभी operating System को support करता है जैसे Linux, microsoft windows, salaries और विभिन्न H/W को support करता है जैसे :- Internet based machine, IBM compitable, macintosh.

Java के win95 के लिए बनाये गये version के लिए निम्नलिखित minimum requirement होती है ।

1. IBM Comperete 486 System.
2. 8 MB RAM.
3. Windows 95 System.
4. 500 MB Hard Disk.
5. CD- ROM drive.
6. Sound card.
7. Mouse.

Java Environment

Java Environment के अंतर्गत विभिन्न development tools और सैकड़ों classes एवं method include रहती हैं । Java के इन develop tools को सम्मिलित रूप से Java development kit (JDK) या S/W development kit (SDK) कहा जाता है । और इसके साथ आने वाली विभिन्न classes और method को सम्मिलित रूप से Java slandered library या API (Application Programming & Interface) भी कहते हैं । JDK के अंतर्गत विभिन्न tools Programmes आते हैं । जिनका प्रयोग Java Program को development करने execute कराने debugging और documentation के लिए किया जाता है ।



Java Environment:- यह दो भाग होते हैं।

1. JDK (Java Development Kit)
2. JAVA API (Application Program Interface)

1. JDK:- इसमें java file को compile करने run करने के लिए class और method होती है।

- **Javac:-** यह java का compiler होता है जो java के source code को byte code में convert करता है। जिसे java interpreter समझ सकता है।
- **Java:-** यह java का interpreter होता है, जो byte code को read करके machine code में convert करता है तथा interprete हुए application को run करता है।
- **AppletViewer:-** इसके द्वारा java की applet run किये जाते हैं।
- **Javadoc:-** इससे java का documentation HTML format में create हो जाता है।
- **Jdb (Java Debugger):-** यह java का debugger होता है। इसके द्वारा हमारे programs में errors find कर सकते हैं।
- **Javah:-** इसमें native methods को उपयोग करने के लिए header file बनाई जाती है।
- **Javap:-** Java disassembler, जो bytes code files को program discription में convert कर देता है।

JAVA (API) = (Application Program Interface):- यह पहले से define कि गई classes और interface का समूह होता है, कुछ predefined packages निम्न है:-

- Language Support Package : **java.lang** : इसमें language को support करने की विभिन्न classes होती है। जिसे compiler automatically import कर लेता है। इसमें primitive types, string, math function, threads and exception की class रहती है।
- Utility Package : **java.util** :- इसमें utility class जैसे vector, hash tables, random number date etc. रहती है।
- Networking Package: **java.net** : इसमें networking और socket programming से related class रहती है।
- Applet Package: **java.applet** : इसमें applet create करने व implement करने की विभिन्न class रहती है।
-
- **Java.io:-** इसमें input output को support करने वाली class रहती है।

General Structure of a Java Program



- Documentation Section
- Package Statement
- Import Statement
- Interface Statement & Definition
- Class Definition
- Main Method Class Definition

JVM (Java Virtual Machine):

Java में बनाये गये किसी भी program को किसी भी platform, H/W या O.S पर चलाया जा सकता है । इस प्रकार Java एक platform independent language है Java ऐसी पहली programming language है जो किसी विशेष H/W या O.S तक सीमित नहीं होता है । Java का यही feature इसे अन्य language से अलग करता है ।

Java के अलावा सभी language के compiler source code को किसी विशेष comp. Machine के लिए machine code में बदलते हैं । Java compiler भी यही work करता है । परंतु दो steps में first step में Java compiler JavaC, program के source code को compile करता है और एक intermediate code produce करता है । जिसे byte code कहा जाता है । यह byte code भी machine code की तरह होते हैं परंतु यह एक ऐसी machine के लिए बनाए गये होते हैं जो वास्तविकता में exist नहीं है । इस तरह की machine को Java virtual machine कहा जाता है । और यह computer की memory में होती है । physically नहीं होती है ।

Second step में Java का interpreter Java इन byte codes को वास्तव में स्थित intermitted के लिए machine code में बदलकर execute करा देता है ।

Virtual machine के लिए बनाया गया code (byte code) machine code की तरह किसी विशेष machine के लिए नहीं होते हैं । यहाँ पर Java Interpreter virtual machine के लिए उत्पन्न किए गए byte code को real machine के लिए machine code में बदलता है और execute कराता है । और यह Java Interpreter अलग-अलग machine के लिए अलग-अलग होता है ।

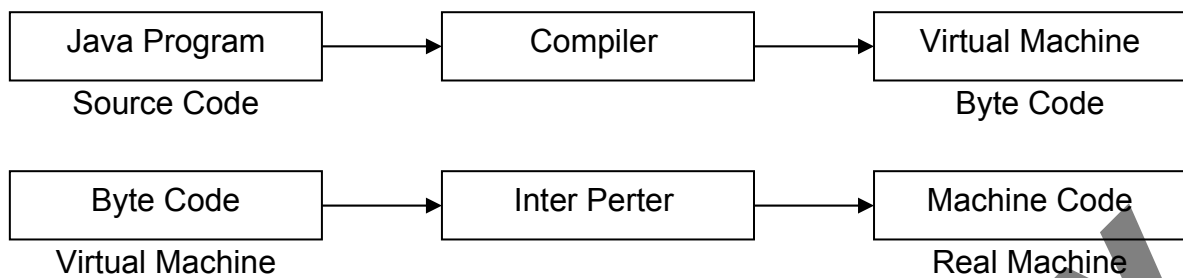
इस प्रकार virtual machine के लिए बनाये गये codes को किसी भी machine पर execute कराया जा सकता है ।

Java का compiler एक intermediate code बनाना है, जिसे byte code कहते हैं, यह byte code ऐसी machine के लिए होते हैं जो physically नहीं होती है। इस machine को JVM कहते हैं, यह computer की memory में रहती है।

Byte code optimized instruction के समूह होते हैं जो java runtime system के द्वारा execute होते हैं।



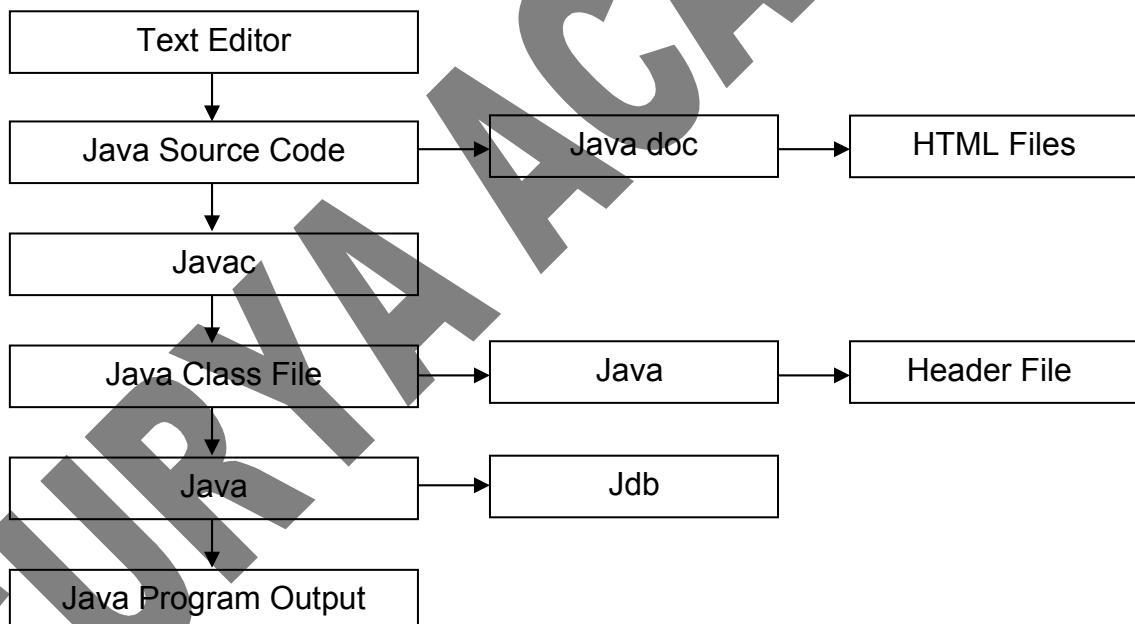
- Java का compiler java में लिखा गया है।
- Interpreter “C” language में लिखा गया है।



Note:-

- 1. Plateform:-** जहाँ program run होता है वहाँ के hardware और software background को plateform कहते हैं।
- 2. Plateform Independent Application:-** ऐसी application जो किसी भी प्रकार के operation system platform पर उपयोग हो सकती है।

API (Application Programming Interface):





Java Support System

Java के विभिन्न application programs एवं applets programs बनाने के लिए कुछ support system की आवश्यकता होती है। जो निम्न है :-

1. **Internet connection:-** Local computer को internet से connect करने के लिए internet connection जरूरी है।
2. **Web Server:-** यह एक program होता है जो internet पर किसी information के लिए भेजी request को accept करता है और उसके अनुसार आवश्यक document भेजता है।
3. **Web Browser:-** यह एक program होता है जिसके द्वारा www को access किया जा सकता है और applet को Run किया जा सकता है।
4. **HTML:-** यह एक language जिसका प्रयोग web page बनाने में किया जाता है इसके द्वारा Hyperlink का प्रयोग कर विभिन्न web pages को link किया जा सकता है।
5. **Applet Tag:-** Java में बनाए गये applet program को HTML में embedded करने के लिए किया जा सकता है।
6. **Java Code:-** (Java source code) इसका प्रयोग Java language में Java application और Java program बनाने में किया जाता है।
7. **Byte Code:-** यह compile Java source code होता है। जिन्हें किसी भी Java virtual machine द्वारा प्रयोग किया जा सकता है। इनका प्रयोग Internet पर applet program को run करने में किया जाता है।

Java Program Structure

एक Java का program विभिन्न section से मिलकर बना होता है जो निम्न है :-

1. **Documentation.**
2. **Package Section.**
3. **Import Section.**
4. **Interface Statement.**
5. **Class definition.**
6. **Main class.**
7. **Main Function.**

1. **Documentation :-** Documentation Section में program से संबंधित details comments के रूप में होती है। program का नाम author name late lime और अन्य details, documentation section में document के लिए mater को लिखने के लिए special type के



comment का use किया जा सकता है । [1** *1] इस comment को documentation comment कहा जाता है ।

2. **Package Section :-** Java Program में सबसे पहला statement package statement use किया जाता है । यह statement computer को यह बताता है कि package का क्या नाम है और इस package के अंतर्गत आती है ।

Ex. :- Package <package name>;

3. **Import statement :-** Java के program में किसी अन्य class की methods या datatype member को access करने के लिए इस class को include करना आवश्यक होता है इसके लिए Import keyword का प्रयोग किया जाता है । यह command C++ के # include command के समान ही होती है ।
4. **Interface Section :-** Interface section में interface को declare किया जा सकता है यह एक operational section होता है ।
5. **Class Defection :-** इस section में उस class को define किया जाता है जिन्हें main class, main function में use किया जाना होता है ।
6. **Main class :-** प्रत्येक Java program में केवल एक ही main method हो सकती है । जिससे program execution के object declare किए जाने हैं और इन object के द्वारा इन classes की method को execute कराया जाता है ।

Tokens

Java में किसी भी program को develop करते समय इस program में विभिन्न प्रकार के declaration, statement, special word एवं expression का use किया जाता है । Java के program में use होने वाली इन individual units को tokens कहा जाता है । Java के compiler द्वारा इन tokens को expression व statement बनाने के लिए use किया जाता है ।

Java program में निम्न 5 प्रकार के tokens होते हैं :-

1. **Reserved keyword :-** abstract, Null, else, float, if, int, for,
2. **Identifiers :-** Uppercase, Lowercase, Underscore.
3. **Literals:-** Integer, character, Float, String, Boder.
4. **Operator :-** सभी Operator.
5. **Separators :-** () Parenthesis, Braces { }, Brackets [], Semi Colon ; , Comma (,) , Period (.)



Statements

Java के Statement किसी भी language इसका use comp. को किसी कार्य को करने के लिए interaction देने में किया जाता है। Java में निम्न लिखित statements होते हैं :-

- | | |
|----------------------------------|--|
| 1. Empty Statements:- | Comments etc. |
| 2. Labeled Statements :- | Labels |
| 3. Expression Statement:- | Declaration statement, post or pre increment mathematical expre. |
| 4. Selection statement:- | if, else switch. |
| 5. Iteration:- | loops |
| 6. Jump:- | Break, Continue, Throw, |
| 7. Synchronisation:- | Syn, keyword (Multithreaded pro. eas) |
| 8. Guarding:- | Try, Catch, finally |

Constants & Variable

Constants से तात्पर्य fixed values से होता है। जो program के execution के समय change नहीं होती है।

Value से तात्पर्य उन सभी value से होता है। जो program के execution के समय change हो सकती है। एक variable प्रोग्राम के execution के समय अलग-अलग समय अलग-अलग value store करता है।

Data Type

Computer में विभिन्न प्रकार के values को store कराया जा सकता है। इन value को अलग-अलग तरह के variables या constant value के रूप में store कराया जा सकता है। जिन्हें data type कहा जाता है।

Java में निम्न लिखित data type होते हैं।

Int :-

- | | | |
|----------|---|-----------------------|
| 1. byte | - | -128 +127 |
| 2. short | - | -32768 + 32767 |
| 3. int | - | +2 x 10 ⁹ |
| 4. long | - | +9 x 10 ¹⁸ |



float :-

- 1. float - -3.4×10^{-38} to $+3.4 \times 10^{-38}$
- 2. Double - -1.7×10^{-308} to $+1.7 \times 10^{-308}$

Char :- 1 Char

String :- Group of Char

Boolean :- True or False

Declaration of Variables:

Java में बने program में computer में किसी भी value को store करने के लिए variable का use किया जाता है । किसी भी program में किसी भी variable को use करने के लिए इसे declare करना आवश्यक है । Java में variable विभिन्न storage location के नाम है । जिसमें value को store किया जायेगा Java के program में variable computer को निम्नलिखित बातें declare करते हैं ।

1. Name – variable
2. Variable किस तरह के data type का है ?
3. Program में variable के declaration का स्थान इस variable का scope decide करता है ।

Declaration:-

Syntax :- Data type variable1, variable2, -----

Example :- int x,y,z;

Initialization: -

Syntax :- Data type variable1=value, variable2=value;

Example :- int x=10, y=20;

Symbolic Constant: - Symbolic Constant एक constant value होती है । जो program के execution के दौरान change नहीं होती है । इस value को किसी भी Symbol या word (PI, max) के द्वारा प्रदर्शित किया जाता है । Symbolic constant declare करने के लिए final keyword का प्रयोग किया जाता है । Java का यह feature C++ के समान ही है । और final keyword C/C++ के # define और const keyword की तरह प्रयोग किया जाता है ।

Declaration of Symbolic Constant

Syntax :- final datatype symbol=value;

Example :- final int max=10;

Java convention (परंपरा) के अनुसार symbolic constant capital letter में लिए जाते हैं । परंतु ऐसा करना जरूरी नहीं है ।



Type Casting

Java में programmer की commands के लिए strong type checking की जाती है। किसी एक data type की value को बड़े data type में store कराया जा सकता है। लेकिन बड़े type की value को छोटे data type में store नहीं किया जा सकता जब कभी किसी बड़े data type की value को छोटे data type में store करना होता है explicitly type conversion करना है। नहीं तो इस expression में java एक error generate करता है। Java का यह feature C, C++ की तुलना में अलग है जिसमें explicit type conversion जरूरी नहीं है। और इस स्थिति में error produce नहीं होती है। type casting करने के लिए निम्नलिखित Syntax का प्रयोग किया जाता है। Variable=(datatype of Variable1, variable2;

Ex :-

```
int x;  
float y=10.50;  
x=(int)y;
```

Rule of type conversion:-

1. जब कभी छोटे data type की value को बड़े data type में store कराया जाता है तो implicit या automatic type conversion होता है। और type casting की आवश्यकता नहीं होती है।
2. जब बड़े data type की value को छोटे data type में store कराया जाता है तो यहाँ explicit type conversion करना होता है। नहीं तो Java एक error produce करता है।

class typecast

```
{  
public static void main(String args[])  
{  
    byte b;  
    int i =257;  
    double d=323.142;  
    b=(byte) i;  
    System.out.println("i and b=" +i + " " +b);  
    i=(int)d;  
    System.out.println("d and b=" +i + " " +d);  
    b=(byte) d;  
    System.out.println("d and b=" +d + " " +b);  
    byte c=45;  
    c=(byte)(c*2);  
    System.out.println("c=" +c);  
}  
}
```



Operator:-

Java language के structure C, C++ के आधार पर ही बने है । एक operator वह symbol है, जो प्रोग्राम में विशिष्ट मेथेमेटिकल या लॉजिकल मेनिप्यूलेशन करता है । C में एक विशिष्ट कार्य के लिये कुछ विशेष Operators हैं ।

Operators

Arithmetic

Relational

Logical

Conditional

Assignment

Shorthand Assignment

Unary

Special

Bitwise

Type

+, -, *, /, %

>, <, >=, <=, ==, !=

&&, ||, !

?:

=

+=, -=, *=, /=, %=

++, --

, comma

~, &, ^, <<, >>, ^

1. Arithmetic Operators:- Arithmetic Operators का प्रयोग आंकिक गणनाओं के लिए किया जाता है ।

Arithmetic Operators मुख्यतः पांच प्रकार के होते हैं :-

OPERATORS MEANING

+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Modulus Operator

Modulus Operator का उपयोग division प्रक्रिया का रिमेण्डर (remender) दर्शाने के लिये होता है । %

Operator का प्रयोग real numbers के साथ नहीं करना चाहिए ।

यदि हम % operator की गणना में ऋणात्मक (-) व धनात्मक (+) चिन्ह का प्रयोग करते हैं तो फल हमेशा पहली संख्या में प्रयोग चिन्ह जैसा आता है ।

eg :-

$$-10\%3 = -1$$

$$10\%3 = 1$$

2. Relational Operator:- जब दो संख्याओं में असमानता एवं समानता प्रकट करते हुए लिखना होता है तब हम Relational operator का प्रयोग करते हैं । 'C' में छः ऑपरेटर्स हैं जो compare करने में काम आते हैं । दो भिन्न expression की वेल्यू से बने expression की वेल्यू को relational expression कहा जाता है ।

OPERATOR MEANING

<	is less than
<=	is less than or equal to
>	is greater than
>=	is greater or equal to
=	is equal to
!=	is not equal to

इन operators का order of precedence समान होता है और Associativity बाएं से दाएं होती है। इन छः operators का उपयोग logical expression में होता है जो यह बताते हैं कि logical expression में condition सही है या गलत। अगर condition सही है तो output 1 होगा यदि गलत है तो 0 होगा। यदि हम एक statement में Arithmetic operator व relational operator दोनों का ही प्रयोग करते हैं तो पहले Arithmetic operator अपना कार्य करता है उसके बाद relational operator अपना कार्य करता है।

eg:- a=6, b=4

if (a==b)

“true”

else

“false”

Output 0

3. Logical Operators:- एक से ज्यादा अनुपातिक operators का प्रयोग करने के लिए हमें logical operator होते हैं :-

&&	logical AND
	logical OR
!	logical NOT

&& → ये दो expression को combine करने के काम आते हैं।

Syntax :- exp1 && exp2

यदि पहली condition गलत है तो लूप का execution terminate हो जाएगा और यदि पहली condition सही और दूसरी condition गलत है तो भी execution terminate हो जाएगा। अर्थात् जब तक दोनों condition सही नहीं होती हैं तब तक execution नहीं होगा।

|| → यह भी दो operators को combine करता है परन्तु इसमें अगर एक condition ही सही हो तब भी लूप का execution होगा।

Syntax :- exp1 || exp2

! → जब हम इस operator का प्रयोग करते हैं तो पहली अवस्था सत्य तथा द्वितीय अवस्था गलत होने पर ही इससे सम्बन्धित statement run होगा।



Syntax :- exp1 ! exp2

4. Conditional Operator :- Conditional Operator से तात्पर्य ? : से है यह if statement का संक्षिप्त रूप है । जो statement if else statement द्वारा कई लाइनों का होता है इस operator के प्रयोग से वह statement एक लाइन में लिखा जा सकता है ।

Syntax :- (condition)?true statement:false statement;

Example:-

```
if (x!=y)
    x=a;
else
    x=b;
```

इस statement को conditional Operation की मदद से इस प्रकार लिखेंगे :-

```
x= (x!=y)?a:b;
```

Example :-

```
int a=15, b=9, c;
c=(a>b)?a:b;
Printf("Greater no. %d",c);
```

Output 15

5. Assignment Operators:- Assignment Operator से तात्पर्य '=' से है । यह Operator किसी variable में कुछ assign करने के काम आता है । Compiler जब value evaluate करता है तो वह right से left जाता है । हमेशा हमें किसी भी value का calculation करना है तो उसमें कोई value न होने पर उसमें 0 लिखना जरूरी है नहीं तो वह garbage value ले लेगा । इस कारण Assignment Operators का प्रयोग किसी भी प्रोग्राम में बहुत जरूरी है ।

eg:- int a=0,b=0,c=0,d=0;

6. Shorthand Operator:- Shorthand Operator से आशय यह है कि इसमें हम expression को shortly या compound करते हैं । shorthand operator में हम दो expressions को एक ही statement में एक साथ प्रयोग में लेते हैं ।

सभी shorthand assignment operator के समान precedence होते हैं । जो साधारण assignment operators (=) के समान होते हैं परन्तु Arithmetic Operators से कम होते हैं ।

eg :-



Simple Assignment

a=5,b=3

a=a+10

a=a-b

a=a*10

a=a/b

a=a%b

Shorthand Assignment

a=5, b=3

a+=10

a-=b

a*=10

a/=b

a%=b

7. Unary Operator:- जो operator single operant से चल जाता है वह नंदतल वचमतंजवत कहलाता है । जब हमें किसी variable में से एक घटाना या बढ़ाना हो तो हम Unary Operator का उपयोग करते हैं । 'Java' में ये operator दो प्रकार के होते हैं :-

1. – Decrement

2. ++ Increment

ये दो प्रकार से प्रयोग किए जाते हैं Prefix तथा postfix . Prefix में operators, operant के पहले आते हैं तथा postfix में operators के बाद में ।

++ variable

variable ++

-- variable

variable --

इस operator में यह ध्यान रखना चाहिये कि यह Operator variable के दायें ओर प्रयोग करना है अथवा बायें ओर क्योंकि दिशा बदलने से इनका स्वभाव बदल जाता है ।

यदि हम variable के बायें ओर इस operator का प्रयोग करते हैं। तो यह पहले variable में एक जोड़ता या घटाता है । यदि हम यह operator variable के दायें ओर प्रयोग करते हैं तो यह operator बाद में घटाता अथवा जोड़ता है ।

eg:- 1. `int c=0, a=5;`
`c=++a;`
`System.out.println(a,c);`
Output → 6,6

2. `int c=0, a=5;`
`c=a++;`
`System.out.println(a,c);`
Output → 6,5

पहले example में हमने a variable में ++ का increment पहले कराया उसके बाद उसे C में store कराया जिससे दोनों की value में 1 का increment हो गया । दूसरे example में a में increment बाद में हुआ है जिससे c में तो पहले a store हो गया उसके बाद a में 1 का increment हुआ ।

8. Special Operators:- ये operators किसी special condition को fulfill करने के लिये ही प्रयोग में लाए जाते हैं । special Operators निम्न हैं ।

1. Comma Operator :- इस operator का प्रयोग एक जैसे statement को जोड़ने के लिए किया जाता है। इसके प्रयोग से हम कई statements को एक लाइन में लिखकर प्रोग्राम को अत्यधिक संक्षिप्त बना

सकते हैं । जब भी हमें प्रोग्राम को अधिक compact बनाने के लिए दो सम्बन्धित expressions को जोड़ना होता है तब comma operator का प्रयोग होता है । इसकी वजह से expression को बायें से दायें association मिलती है ।

eg:- int res,a,b;
 res=(a=5,b=10,a+b);
 System.out.println(res);

9. Bitwise Operator:- इस operator का प्रयोग Bit को जांचने और बिट को दायें और बायें हटाने के लिए प्रयोग किया जाता है । इस operator का प्रयोग Integer& character type के variable पर ही किया जाता है । Bitwise operators निम्न हैं .

~	One's complement
&	Bitwise AND
	Bitwise or
^	Bitwise exclusive or
<<	shift left
>>	shift right

OPERATOR PRECEDENCE: -

 **Arithmetic Operators:-** अर्थमेटिक ऑपरेटर्स *, /, %, +, -, = का क्रम या precedence निम्नलिखित है -

Order of Precedence	Operators
1 st	*, /, %
2 nd	+, -
3 rd	=

इन ऑपरेटर्स की एसोसिएटिविटी बाएँ से दाएँ होती है ।

- 1- $A/B * C$ समान है $(A/B) * C$ के अर्थात् हालाँकि / * की प्रायोरिटी समान है, परन्तु कौन सा पहले आया वही पहले evaluate होता है ।
- 2- अगर एक्सप्रेशन में पेरेंथेसिस भी प्लेस किया गया है तो पेरेंथेसिस के अन्दर का एक्सप्रेशन भी उसी प्रायोरिटी आर्डर में रहेगा जो ऊपर समझाया गया है और सबसे भीतर के पेरेंथेसिस वाले एक्सप्रेशन की गणना पहले की जाएगी यानि $2 * (x * (y + 3))$ में $(y + 3)$ की गणना पहले होगी इसके बाद $(x * (y + 3))$ की ओर अन्त में $2 * (x * (y + 3))$ की गणना होगी ।
3. हमेशा पेरेंथेसिस को जोड़ों में प्रयोग करें, अगर आन क्लोजिंग पेरेंथेसिस लगाना भूल गए तो एरर हो जाती है ।



इसमें पहले सबसे अन्दर वाला ब्रेकेट फिर * ऑपरेशन और फिर एडिशन होगा । तब यह ऑपरेशन होता है जिसकी ज्यादा priority होती है । अगर एक से अधिक ऑपरेटर की समान प्रायोरिटी है तो पहले आने वाले ऑपरेटर की गणना पहले होगी ।



Relational Operator:- ऑपरेटर <,>,<=,>=, की प्रायोरिटी समान होगी और ऑपरेटर !=,== की प्रायोरिटी इन से कम होगी ।

Type casting

From	To
Byte	Char, short, int, long, float, double.
Short	int, long, float, double.
int	long, float, double .
char	int, long, float, double.
long	float, double.
float	double.
double	float.

CONTROL STATEMENTS/CONTROL STRUCTURE:

Statements computer को किसी प्रकार का कार्य करने के लिए दिये गये instructions होता है यह कार्य data को एक स्थान से दूसरे स्थान तक ले जाना हो सकता है या किसी प्रकार के निर्णय को लेना हो सकता है या किसी action को दोहराना हो सकता है इस प्रकार से C Program में statement का स्वरूप सबसे छोटी executable unit होता है ।

Decision Making/Branching Statement:- Decision Making Statement का उपयोग condition check करने के लिए किया जाता है ।

- If Statement:-** यह दो तरीके से निर्णय लेने वाला statement है । इसमें execution के दौरान सबसे पहले condition को check किया जाता है । यदि condition true है तब statement true execute होता है । ब्रेकेट { } का use statement true होने पर एक से ज्यादा statement लिखवाने के लिए किया जाता है । condition false होने पर false statement true execute होता है ।

Syntax:: if (text expression)

```
{
    statement blocks;
}
```

statement x;

Note:- अगर condition के true होने पर एक से अधिक statements executed कर बाने है हो तो उन को { } में देना आवश्यक है ।



- ii. If else:- if else statement में दी गई condition के true होने पर if के नीचे लिखे गये statement के block execute होते हैं अन्यथा else के नीचे दिये गये statement के block execute होते हैं। if else में condition के true होने पर अगर हम एक से अधिक statement executed करवाने हैं तो उनको { } में देना आवश्यक है नहीं syntax error आयेगी।

Syntax: if (test expression)

```
{
    true block statement (s);
}
else
{
    false block statement (s);
}
statement x;
```

- iii. **Switch Statement:-** इसका उपयोग एक से अधिक विकल्पों में से किसी एक को select करने के लिए किया जाता है।

Rules:-

- Switch में दिया गया expression integer या character होना चाहिए।
- Cause value हमेशा एक constant होना चाहिए।
- Case statement के बाद break का उपयोग करना चाहिए।

Note:- default statement का execution किसी भी cause के match ना होने पर होता है।

Syntax: switch (expression)

```
{
    case value 1:
        block 1
        break;
    case value 2:
        block 1
        break;
    case value n:
        block 3
        break;
```



```
default:
    false statement;
}
statement x;
```

Looping statement:- Program में किसी condition के आधार पर किसी block को बार-बार excute करने के लिए looping statement का उपयोग करते हैं।

Looping statement तीन प्रकार के होते हैं।

- i. While Loop
 - ii. Do While Loop
 - iii. For Loop
- i. **While Loop:-** इस loop में पहले condition check की जाती है, यदि condition true है तो loop की body को excute किया जाता है तथा excution के बाद condition को दुबारा check किया जाता है। अगर फिर भी condition true होती है तो loop चलता रहता है, जैसे ही condition false हो जाती है तब यह process रुक जाती है।

अगर हम { } नहीं देते है तो loop की body में while के नीचे वाला statement उसकी body कहलाता है

Initialization;

Syntax: while (test condition)

```
{
    body of the loop
}
```

- ii. **do while Loop:-** इस loop में पहले statement excute होते है फिर condition check होती है। अगर condition true है तो loop फिर से excute होगा नहीं तो loop रुक जाएगा। इस प्रकार से condition के false होने पर भी यह loop एक बार execute होता है।

Syntax:do

```
{
    body of loop
}
```

while (test condition);

- iii. **for Loop:-** For loop का उपयोग किसी loop को fixed time तक चलाने के लिए किया जाता है। इस loop में एक line में 3 statement दिये जाते हैं।

- a. initialize a value
- b. to check the candistion
- c. counter को बढ़ाने के sating की जाती है।

Note:- एक से अधिक statement देने के लिए { } में statement देते हैं।

for (initialization; test condition; increment)



```
{  
    body of loop  
}
```

Nested Loop:- जब किसी एक loop के अंदर दूसरे loop का उपयोग किया जाता है तो वह nested loop कहलाता है।

Note:- Outer loop का counter जब तक नहीं बदलता जब तक inner loop पूरा ना हो जाये।

JUMPING STATEMENT:

- a. Break
- b. Continue
- c. Return
- a. Break:- Break का यूज किसी लूप से बाहर आने या switch statement से बाहर आने के लिए किया जाता है।
- b. Continue:- इसका use किसी loop के कुछ भाग को छोड़कर उसी loop को आगे से execute करने के लिए किया जाता है।
- c. Return:- Return key word किसी function से value return करवाने के लिये करते हैं।

Labelled Loop:- अगर हमें nested loop से बाहर आने या nested loop को continue करना है, तब हम labelled break और labelled continue का उपयोग करते हैं यहाँ पर labelled एक valid identifier है।

Ex: Labelled continue

outer:

```
for (int m=1; m<11; m++)  
{  
for (int n=1; n<=11; n++)  
{  
System.out.print(" " + m);  
if (n==m)  
continue outer;  
system.out.print(" Hello ");  
} }  
}
```

Ex: Labelled break:

outer:

```
for (int m=1; m<11; m++)  
{
```



```
for (int n=1; n<=11; n++)  
{  
    System.out.print(" " + max);  
    if (n==m)  
        break outer;  
    System.out.print(" Hello ");  
}  
}
```

UNIT – II

CLASS, OBJECT AND METHOD:-

Java एक object oriented programming language है जिसमें विभिन्न data को store करने के लिए basis data types के साथ user स्वयं का नया data type भी बना सकता है इसके लिए एक विशेष mechanism class का प्रयोग किया जाता है । इस प्रकार class के user (program), input किए जाने वाले data के nature के अनुसार कोई भी नया data type बना सकता है । एक बार आवश्यकतानुसार कई समान प्रकार के data type बनाए जा सकते हैं । इस प्रकार बस एक object के लिए template की तरह होती है । और एक object एक class का instance होता है ।



जब भी class declare की जाती है । तो वह निश्चित format व nature के अनुसार declare की जाती है । जब class को declare करने के लिए उन class के अंदर कौन-कौन से data type आएंगे और इस data पर apply किया जाने वाला code, method specify किया जाता है किसी भी class declaration में केवल data या केवल code अथवा दोनों भी हो सकते हैं । class को declare करने के लिए class keyword का use किया जा सकता है ।

General form of class:-

```
class classname
{
    data type instancevariable1;
    data type instancevariable 2;
    -----
    Return datatype methodname (parameter list)
    {
        //some comand
    }
}
```

Characterestics / Some Facts of Class:-

1. कोई भी class हमेशा बस keyword के द्वारा declare की जाती है ।
2. class defination हमेशा brackets { } द्वारा close रहती है । और class defination के अंत में semi colon का use नहीं किया जाता है । [C,C++ से अलग feature है]
3. class defination में प्रयोग किए गए data या variable इस class के instance variable कहलाते हैं।
4. इन instance variable पर apply होने वाले programming code को बस defination के अंदर ही सामान्य function की तरह ही प्रयोग किया जाता है । और इन्हें class की method कहा जाता है ।
5. class में define किए गए विभिन्न instance variable और method को सम्मिलित रूप में class member कहा जाता है ।
6. अधिकांश classes में instance variable को method में प्रयोग किया जाता है । या method के द्वारा use किया जाता है इसलिए method यह निर्धारित करती है कि किसी class के instance variable किस प्रकार use किए जायेंगे ।
7. class के द्वारा declare किए गए नए data type के प्रत्येक variable को object कहा जाता है ।
8. प्रत्येक object में instance variable की अपनी स्वयं की copy होती है । अर्थात् प्रत्येक object में समान प्रकार के instance variable use किए जाते हैं । परंतु प्रत्येक object में इन instance variable की value अलग-अलग अथवा समान हो सकती है ।



Adding variable & method

किसी भी class में प्रयोग किए जाने वाले instance और variable को class scope में declare किया जाता है । class के अंदर विभिन्न instance और variable को सामान्य function & variable की तरह ही विभिन्न modifiers का प्रयोग कर declare किया जाता है ।

Defination of variable :-

```
class abc
{
type variable1;
type variable2;
-----
-----
type variabeln;
}
```

Defination of method:-

```
class abc
{
type method name (parameter list)
{}
-----
-----
}
```

Declaration of object:- जब किसी class को declare किया जाता है । तो वह class data type को प्रदर्शित करती है । इस नए datatype का प्रयोग इसी type के नए object data declare करने में किया जाता है । इस प्रकार object user द्वारा बनाए गए नये data type के variable ही होते हैं । object का declaration और प्रयोग दो steps में होता है । पहली step में object को declare किया जाता है । दूसरी step में इस object के लिए memory allocate की जाती है । और इसे initialize किया जाता है ।

Statement

I. rectangle rect;

Effects

rect

Nul

II. rect=new rectangle(10,15);

rect

width

hight



Accessing class member:- class में define किए गए विभिन्न instance variable और method इस class के member कहलाते हैं। इन विभिन्न member को इस class के object के साथ dot (.) operator का प्रयोग कर access किया जा सकता है।

Syntax:- object name.Instance-variable;
object name.methodname (parameter);

Constructors

“किसी class में declare किए गए instance को initialize कराने के लिए एक विशेष method का प्रयोग किया जाता है। जिसे constructor कहते हैं। किसी object के बनने पर constructor इसे default values या constructor में pass की गई value द्वारा initialize करता है। जब किसी object को बनाया जाता है तो constructor call किया जाता है। और इस constructor के द्वारा object का type, memory allocation एवं initialization निर्धारित किया जाता है। constructor एक सामान्य method से अलग होता है। क्योंकि constructor का declaration करते समय कोई भी return type specify नहीं किया जाता [void भी नहीं लिखा जाता है] इसके साथ ही constructor का नाम class name के समान होता है। किसी भी class में एक अथवा अधिक constructor declare किए जा सकते हैं। इन सभी constructor के parameters की संख्या type अलग-अलग होना चाहिए। इन सभी constructors को सम्मिलित रूप से multiple constructor कहा जाता है। और इस प्रकार किसी class में एक से अधिक constructor declare करने की प्रक्रिया को constructor overloading कहा जाता है।

Types of constructor:-

किसी class में विभिन्न constructor उनके parameters की संख्या और type के आधार पर अलग-अलग तरह से declare किए जा सकते हैं। इस प्रकार किसी class में निम्नलिखित constructor हो सकते हैं।

1. **Default constructor:-** Default constructor में class के instance variable के लिए default value assign की जाती है। इस प्रकार constructor द्वारा जो भी object बनाए जाते हैं। उन सभी में यही default constructor होती है।
2. **Parameterized constructor:-** इस constructor का प्रयोग program execution के समय कुछ value को pass कराकर object को initiating किया जा सकता है। इस प्रकार के



constructor को declare करते समय cons. Definition में parameter को भी declare किया जाता है ।

3. **Copy constructor:-** इस constructor का प्रयोग इसी class के पहले से बने किसी object के instance variable की value के द्वारा नए बनाए जा रहे object को initialize करवाने के लिए किया जाता है । इस प्रकार इस constructor में जो नया object की exact (वैसी की वैसी) copy होती है ।

Method Overloading

किसी class में समान नाम की दो method हो सकती है । जब किसी कार्य को विभिन्न data type पर जाना होता है । तो प्रत्येक data type के लिए अलग- अलग नाम से method declare नहीं की जाती है । लेकिन इस method में pass किए जाने वाले Argument का type अलग- अलग रखा जाता है । इस प्रकार compiler द्वारा इन method को समान नाम होने पर भी अलग- अलग identity कर लिया जाता है । program execution के दौरान जिस datatype के Argument इन method को pass किए जाते हैं । उसके आधार पर compileres उपयुक्त method को call कर लेता है ।

इस प्रकार एक ही कार्य को अलग- अलग data type पर करने के लिए समान नाम की कई method overloading की जाती है तो इसे method overloading कहा जाता है ।

Method overloading करने के लिए निम्नलिखित बातें याद रखना जरूरी है :-

1. Method के नाम समान होना चाहिए ।
2. इन विभिन्न methods में pass किए जाने वाले arguments के datatype अलग-अलग होना चाहिए ।
3. Method overloading में किस method को call किया जाए compiler method के return type को ध्यान नहीं रखता है ।

//Example of Method Overloading

```
class methodover
```

```
{
```

```
int a,b;
```

```
methodover()
```

```
{
```

```
a=4;
```

```
}
```

```
methodover(int x,int y)
```



```
{  
    a=x;  
    b=y;  
}  
methodover(int t)  
{  
    a=b=t;  
}  
void show()  
{  
    System.out.println(a+ " " + b );  
}  
}  
class methodoverload  
{  
    public static void main(String args[])  
    {  
        methodover obj=new methodover(5);  
        methodover obj1=new methodover(5,4);  
        obj.show();  
        obj1.show();  
    }  
}
```

Static Members:

सामान्यतः class में जो variable declare किए जाते हैं वे इस class के प्रत्येक object के साथ memory में बनते हैं। इस प्रकार इस class के जितने भी object बनाए जायेंगे। instance variable के भी उतने ही group होंगे। लेकिन class में इस प्रकार variable भी declare किए जा सकते हैं जो memory में केवल एक बार बनते हैं। और इन्हें class के विभिन्न object द्वारा share किया जाता है। इस प्रकार के variable को class का static variable कहा जाता है। और इसे सामान्य variable की तरह ही declare किया जाता है। लेकिन इसके साथ static variable को सामान्यतः static method द्वारा access किया जाता है। static variable को call करने के लिए किसी object की आवश्यकता नहीं होती है।



Static Method:

Class में declare किए गए static variable को access करने के लिए methods अन्य सामान्य methods की तरह ही declare की जाती है । लेकिन declaration के समय इसमें static keyword का प्रयोग किया जाता है । static method को call करने के लिए किसी object की आवश्यकता नहीं होती है बल्कि इसे class name के साथ dot (.) operator का use करते हुए प्रयोग किया जाता है ।

class में declare किए गए static variable और static methods को सम्मिलित रूप से class के static member कहा जाता है ।

declaration of static member:-

```
class abc
{
    -----
    static datatype variable name
    -----
    static void show()
    {
        -----
        -----
    }
}
```

calling of static methods:-

classname.methodname (Argument)

//Example

class staticblock

{

int a=4;

static int x=4;

static int y;

static void statfun(int r)

{

System.out.println("x=" +x);

System.out.println("y=" +y);

```
System.out.println("r=" +r);  
}  
  
static      //Static block  
{  
    System.out.println("In Static Block");  
    y=x+4;  
}  
}  
class examplestatic  
{  
    public static void main(String args[])  
    {  
        staticblock.statfun(100);  
    }  
}
```

Nesting of Methods:- किसी methods के method को call किया जा सकता है । इसे nesting of method कहते है । methods की nesting कई level तक हो सकती है । मतलब एक method के अंदर दूसरी व दूसरी method के अंदर तीसरी method को इसी प्रकार के अंदर किसी दूसरी methods को call किया जा सकता है ।

INHERITENCE

Java तें बनाई गई किसी class और intence variable और इसके source code के द्वारा किसी अन्य class में भी use किया जा सकता है । इसके लिए Java का एक विशेष feature प्रयुक्त होता है ।

जब किसी दूसरे program में पहले से बनाए गए program के source code की आवश्यकता होती है । तो इसके द्वारा बनाकर inherit कर लिया जाता है । Inherit करने पर पहले से स्थित किसी class की विभिन्न method एवं variables उनके साथ प्रयोग किए गए access specifire (modifire) के आधार पर नई class में acces में acessabile हो जाते है । इस प्रकार किसी पहले से स्थित बर्से के source codes को नई class में access करने की प्रक्रिया को inharitence कहा जाता है ।

Java में पहले से स्थित बर्से को नई class मे access करने के लिए extends कहा जाता है ।

जिस class को inherit किया जाता है उसे supper class कहा जाता है । जिस class में super class के member को access किया जाता है उसे sub class कहा जाता है ।

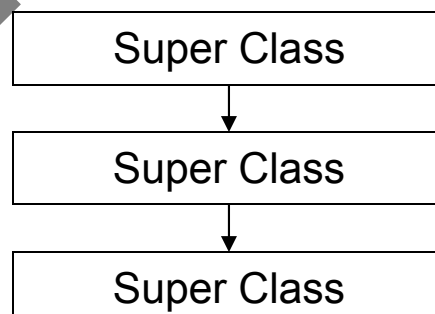


```
class abc
{
-----
-----
-----
}
class two extends one
{
-----
-----
-----
}
```

Types Of Inheritance

1. Single .
2. Multilevel.
3. Multiple.
4. Hairarical.
5. Hybrid.

1. **Single Inheritance**:- जब एक supper class को एक sub class के द्वारा Inherit किया जाता है तो इस प्रकार का inheritance single inheritance होता है ।
2. **Multilevel Inheritance** :- जब एक supper class को एक sub class के द्वारा Inherit किया जाता है । और बाद में इस sub class को क्रमशः विभिन्न sub class के द्वारा inherit किया जाता है । तो इस प्रकार का inheritance multilevel inheritance होता है ।



3. **Multiple Inheritance** :- Java C,C++ की तरह दो supper class को एक sub class में inherit नहीं कर सकता है इसलिए Java में multiple inheritance पूरी तरह C++ के समान use न



कर कुछ अलग तरीके से किया जाता है । Java में जब किसी sub class में किसी base class को inherit किया जाता है और बाद में किसी interface के द्वारा किसी अन्य class की methods के declaration को प्रयोग किया जाता है । तो इसे multiple inheritance कहते हैं ।

Class A	Interface XYZ
<pre> { void display() { ----- } } class B implements XYZ { void show () { ----- ----- } } </pre>	<pre> { void show() { </pre>

4. **Hierarchical Inheritance** :- जब किसी बसे में किसी supper class को inherit किया जाता है और विभिन्न declaration को interface द्वारा implement कराया जाता है । इस प्रकार के inheritance को Hierarchical inheritance कहा जाता है ।

5. **Hybrid Inheritance** :- जब विभिन्न levels पर विभिन्न classes multilevel inheritance और multiple inheritance दोनों ही प्रकार से की जाती है । तो इस प्रकार का inheritance hybrid inheritance कहा जाता है ।

supper class	interface XYZ
sub class	{

sub-sub implements XYZ	-----
	}

Overriding Methods

जब किसी super class को किसी sub class में inherit किया जाता है और sub class व super class में कोई समान method होती है तब sub class में super और sub class की common method दोनों ही accessible रहती है । sub class की method को already sub class में define रहती है इसके साथ ही super classes की common method को sub class में inherit कर लिया जाता है । इस प्रकार sub class में दोनों methods accessible रहती है । जब sub class के object के द्वारा इस method को call किया जाता है तो हमेशा sub class की method call होती है super class की नहीं । इस प्रक्रिया को **method overriding** कहा जाता है ।



super class की method को call करने के लिए sub class में ही एक method बना जी जाती है और इनमें indirectly super keyword के साथ dot operator का use करते हुए super class method को call कर लिया जाता है ।

Syntax :- super.methodname ();

Super Key Word:- Super key word का उपयोग inheritance में किसी sub class में किया जाता है। इसके द्वारा हम super class के constructor को call कर सकते हैं।

अगर super class और sub class में instance variable और instance methods के नाम एक समान हैं, तब sub class में Super class के member को access करने के लिए super key word का उपयोग किया जाता है।

अगर super class में constructor है तो sub class में constructor होना आवश्यक है तब उस constructor में super class के constructor को call करने के लिए पहला statement super होना चाहिए।

Ex:

//Example of Method Overriding and super keyword

class base

```
{
    int a,b;
    base (int x,int y)
    {
        a=x;
        b=y;
    }
    void show()
    {
        System.out.println("A=" +a +"b="+b);
    }
}
```

class der extends base

```
{
    int s;
    der(int a,int b,int c)
    {
        super(a,b);
        s=c;
    }
}
```



```
void show ()
{
    System.out.println("S="+s);
}
}

class inherid
{
    public static void main(String args[])
    {
        der object=new der (1,2,4);
        object.show();
    }
}
```

Output

S=4

Final variable, methods, classes

Final Variable :- सामान्य variables की value prg. execution के समान change हो सकती है । परंतु कुछ condition में कुछ values program के execution के समय fix रखना आवश्यक होता है तब इन variable declaration के लिए सामान्य variable declaration के साथ-साथ final keyword का प्रयोग किया जाता है । एक बार किसी variable को final declare करने के बाद इसकी value को prog. execution के समय change नहीं किया जा सकता । यदि किया command में इस final variable के value को change किया जा रहा होता है तो compiler एक error generate करता है । final variable को symbol constant या constraints के नाम से भी जाना जाता है ।

Final Method:- जब किसी super class method को sub class में override नहीं कराया जाना होता है तो इस method को final keyword के साथ declare किया जाता है इस प्रकार की method को final method कहा जाता है । final method को इसकी sub class में override नहीं किया जा सकता मतलब किसी sub class में इसकी super class की final method की तरह नई method declare नहीं की जा सकती ।

Final Class:- जब किसी class को एक separate class के रूप में ही use किया जाना होता है ओर इससे किसी नई class में inharit नहीं किया जाना होता है तो इसे final keyword के साथ declare किया जाता है तो इसे class को final class कहा जाता है । एक बार किसी class को final declares करने के बाद इसमें कोई नई class inharit नहीं की जा सकती लेकिन इसके object बनाकर इसकी method call की जा सकती है । इस प्रकार कोई भी final class super class नहीं बन सकती ।



Finalize Method:- Java में memory management के लिए एक विशेष method garbage collection का प्रयोग किया जाता है इसके अंतर्गत जब भी memory में कोई ऐसा object होता है जिसकी program execution areas आगे कोई आवश्यकता नहीं है तो उसे memory से हटा दिया जाता है और इस object के program के प्रयोग के लिए free कर दि जाती है ।

कुछ विशेष परिस्थितियों में कुछ commands garbage collection के समय चलाना आवश्यक होती है । तब इन commands को classes में एक विशेष method declare करके इस methods में लिखा जाता है इस प्रकार की method को जो हमेशा किसी object के लिए garbage collection के समय automatically execute होती है । finalize method () कहा जाता है । इस method का नाम हमेशा finalizer () होता है और return type void होता है ।

Syntax: protected void finalize()

```
{  
    // Object and resources release code  
}
```

Abstract class

जब किसी class को इस प्रकार से use किया जाना होता है कि इसके object न बनाए जा सके लेकिन inherit किया जा सके तो इस प्रकार की class को abstract keyword के साथ declare किया जाता है इस प्रकार की class को abstract class कहा जाता है । यदि किसी class को abstract class declare कर दिया जाता है तो इस class के object नहीं बनाए जा सकते । इस प्रकार इस class की method instance variable का directly access नहीं किया जा सकता है । abstract class की method व instance variable को access करने के लिए इस class को किसी sub class में inherit कर लिया जाता है और फिर इस sub class के object के द्वारा indirectly abstract को access किया जा सकता है ।

declaration :- abstract class ABC

```
{  
    _____  
    _____  
}
```

Abstract method:- Abstract class में declare की गई method abstract method कहलाता है । इन method को abstract class का object बनाकर call नहीं किया जा सकता ।

Ex: abstract class mlcr

```
{
```



```
protected int code;
abstract void getcode(int c);
void show()
{
    System.out.println(code);
}

}
class bca extends mlcr
{
    void getcode(int x)
    {
        code=x;
    }
    void show()
    {
        System.out.println(" bca code is "+code);
    }
}
class msc extends mlcr
{
    void getcode(int x)
    {
        code=x;
    }
    /* void show()
    {
        System.out.println(" msc code is "+code);
    } */
}
```



```
class myclass
{
    public static void main(String args[])
    {
        mlcr ref; //It creates reference
//    mlcr on=new mlcr(); //Compilation error
        bca ob1=new bca();
        msc ob2=new msc();
        ref=ob1;
        ref.getcode(101);
        ref.show();
        ref=ob2;
        ref.getcode(102);
        ref.show();
    }
}
```

UNIT – III

Array, String and Vectors

Array:- एक समान datatype के variables का group array कहलाता है । एक array में विभिन्न variables continues memory location में store रहते हैं । और इन्हें सम्मिलित रूप से किसी एक नाम से पहचाना जाता है । इसे array name कहते हैं array में स्थित विभिन्न variable को individually विभिन्न number द्वारा access किया जा सकता है जिसे इस array की subscription कहा जाता है । किसी भी array में यह subscript (zero) से start होती है और array में जितने elements हैं उससे एक कम होती है ।

Types of Array:- array दो प्रकार की होती है।



1. **Single Dimensional Array:-** जब कोई एक sub script रखता है। अर्थात इसमें data को एक row और एक से अधिक columns में व्यवस्थित करके रखा जाता है। तब इस प्रकार के array को single dimensional array कहा जाता है।

Initialization of an array:- किसी array को initialize करते समय उसका size देना आवश्यक नहीं है।

declaration :- Syntax: datatype identifier [];

or datatype [] identifier ;

Ex: int n[];

or

int []n;

Memory allocation :- identifier []=new datatype[size];

Ex: int n[]=new int size];

Initialization :- Syntax: datatype identifier []= {value1, value2,.....};

Or datatype []identifier = {value1, value2,.....};

Ex: int n[]={1,2,3,4};

2. **Two Dimentional array:-** Double dimensional array में row और columns में data को व्यवस्थित करके रखा जाता है इसमें array की एक से अधिक subscript होती है।

Declaration :- Syntax: datatype identifier [][];

or datatype [][] identifier ;

Ex; int n[][]= new int [5] [2];

Memory allocation :- identifier [][]=new datatype[Rows][Columns];

Ex: n= new int [5] [2];

Declaration :- int n[][]= new int [5] [2];

Initialization:- int n[][]= {{1,2,3}; { 4,5,6}, { 7,8,9} };

int n[0][0]=4;

int n[1][1]=5;

String:- विभिन्न characters का group string कहलाता है। strings form में information को store कराने के लिए Java में एक एक builtin class है जिसे Java.lang में define है। जब किसी string को comp. में store किया जाना होता है। तो string type के object का प्रयोग किया जाता है। इसके साथ ही विभिन्न string manipulation का काम string class में में define विभिन्न method द्वारा किया जाता है। string clas को प्रयोग करने के लिए इसे expilicity import करने की आवश्यकता नहीं होती है। क्योंकि यह Java.lang में define होता है। और Java.lang प्रत्येक Java program मे implicitly import होता है।

Declaration :- String str;

memory allocation:- str=new String();



Initialization:- str=new String ("Hello");

all in one :- String str=new String("Hello");

String method

methods

Task procedure

S2=S1.toLowerCase → convert. the string S1 to lower.

S2=S1.toUpperCase → convert. the string S1 to upper.

S2=S1.replace ('x','y')→ replace all apprence of x with y.

S2=S1.trim→ Remove while space at the bugging \$ and of the string.

S1.length () → Gives the length of S1

S1.charAt (n) → Gives the char of S1

S1.compair to (S2) → Return (-) if S1<S2, (+) if S1>S2 (0) if S1=S2

S1. substring (n) → दी गई position के बाद के पूरे char.

S2. substring (n,m) → Ashish में "ish" देगा 2-6=4 2 position से 4 character देगा ।

String Value of (p)

P. to string →

S1. index ('x')→

S1.Index of ('x',n)→

String.value of (Variable)

String Buffer Class:- String Buffer Class के द्वारा string को run time पर add व edit किया जा सकता है। इसमें character व substring को append व nidil में insert किया जा सकता है।

- i. setChar At (n,x):- इसमें n position के character को x character से modify कर सकते हैं।
- ii. append (S2):- इसमें string S2, S1 end में append हो जाएगा।
- iii. Insert (n, S2):- इसमें string S2 string S1 में N position पर insert हो जाएगा।

Vector:

Java में बनाए गए किसी program के द्वारा comp. memory में विभिन्न datatype के data को एक group के रूप मे store कराने के लिए Java के एक builtin data type vector का प्रयोग किया जाता है । Vecter data type वास्तव में एक data type का object है जिसे Vector class द्वारा बनाया जाता है । Vector class Java package के क sub package Util में declare एक class है जिसका मुख्य उद्देश्य run time पर user के requirement के आधार पर विभिन्न datatype को एक group के रूप में store कराना होता है । एक Vector व array में यह difference होता है कि Vecter अलग-अलग datatype के object को एक



group के रूप में store कराया जा सकता है । जिसमें से प्रत्येक object को individual उसके index No. द्वारा access किया जा सकता है । जबकि array में समान data type की values को ही store कराया जा सकता है । उदा. यदि int data type का array बनाया जाता है । तो इसमें केवल int data type की value को ही store कराया जा सकता है अन्य किसी datatype की नहीं । जबकि vector में विभिन्न object के रूप विभिन्न data type की value को store कराया जा सकता है ।

Declaration of vector variable:- Vector के विभिन्न object को vector keyword के द्वारा declare किया जाता है । किसी vector के object को केवल declare करके use नहीं किया जा सकता उसका memory allocation आवश्यक है ।

Import Java Util. *;

Vector V;

Vector obj-name;

Memory Allocation:- पहले declare किए गए object के लिए memory allocation के बाद ही vector object को प्रयोग किया जा सकता है । किसी vector object के लिए memory allocation कराने के लिए new keyword के साथ vector class के किसी भी constructor का प्रयोग किया जा सकता है ।

Vector V=new Vector (1,1);

Syntax :-

object name=new constructor ();

DEclaration & memory allocation:-

Vector V=new vector(1,1);

Ex: import java.util.*;

class vecfunction

{

public static void main(String args[])

{

Vector v=new Vector();

for(int i=0;i<args.length;i++)

{

v.addElement(args[i]);

}

v.removeElement("bca");



```
System.out.println("After Deleting bca");
for(int i=0;i<v.size();i++)
{
    System.out.println(v.elementAt(i));
}

// v.insertElementAt("BCA",6);
v.addElement("Surya");
System.out.println(" In Display All Elemnt ");
for(int i=0;i<v.size();i++)
{
    System.out.println(v.elementAt(i));
}
v.removeElementAt(2);
System.out.println(" In Show after Deletion ");
for (int j=0;j<v.size();j++)
{
    System.out.println(v.elementAt(j));
}
v.removeAllElements();
if(v.isEmpty());
{
    System.out.println("Vector is Empty");
}
}
}
```

Wrapper Class:- Java में primitive data type किसी method में call by value के द्वारा pass होता है। जबकि object call by refrance के द्वारा pass होते हैं। Primitive data types को call by refrance के द्वारा pass करने के लिए पहले object में बदलना होता है। जिसके लिए java में wrapper class में होती है।

कुछ wrapper class निम्न है:-

Builtin Datatype	Equivalent wrapper class
int	Integer
char	Character



float Float

long Long

Methods

1. Primary datatype को object में बदलना [use of constructor]

float x=2.3f;

1. Integer obj=new Integer(20);
2. Float f=new float(x);

2. object to primary data type

1. int i=obj.intValue();
2. float t= f.floatvalue();

3. Number to string:-

String str = Integer.toString(30);

String s = Float.toString(x);

4. String to Numeric object:-

Integer in =Integer.valueOf(str);

Float obj.=Float.valueOf(s);

5. Numeric String to Primary data type:-

int x=Integer.parseInt (str);

long l= long. parseLong (args[0]);

Interface

Java में S/W development के लिए विभिन्न classes में एक रूपता लाने के लिए class की तरह ही एक entity interface का प्रयोग किया जाता है । किसी भी method में और symbolic constraints को declare कर दिया जाता है । जो विभिन्न classes में प्रयोग किए जाने होते है । किसी भी interface में हमेशा final variable या symbolic constraints ही declare किए जाने है variable नहीं । और interface में declare की गई प्रत्येक method केवल prototype के रूप में use की जाती है । Interface की किसी method में commands का प्रयोग नहीं किया जाता है बल्कि इन्हें केवल declare किया जाता है । define नहीं किया जाता । एक ही interface को प्रयोग करने वाली विभिन्न classes में इस interface में declare की गई method common होती है ।

Defining a interface :- एक interface को define करने के लिए interfaace को प्रयोग किया जाता है एक interface की defination class जैसी ही होती है । लेकिन अंतर केवल class व interface keyword का होता है ।



General Form:

```

access identifier    interface name

{

    return type    method name (parameter list);
    type final var=value;
    -----
    -----}

```

Implimenting Interface

किसी Interface को प्रयोग करने के लिए विभिन्न classes के द्वारा इसे implement किया जाना आवश्यक होता है । किसी interface का implementation करने पर इस class के द्वारा इस implimentaion के variable को access किया जा सकता है । और इसकी method का declaration प्रयोग किया जा सकता है । किसी interface का implementatoion करने के लिए implements keyword का प्रयोग किया जाता है।

```

class    ClassName    implements    InterfaceName
{

    Statements;
    Statements;
    Statements;
    -----
    -----
}

```

Extending an interface:

Syntax: class className extends superclass implements interface1, interface2

```

{
    -----
    -----
}

```

Example:

```

// multiple inheritace
interface myinter1
{ static final int code=101;
  void fun1();
}

```

```

interface myinter2 extends myinter1

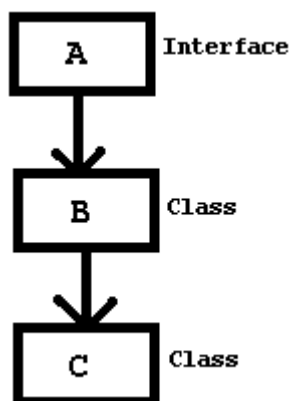
```



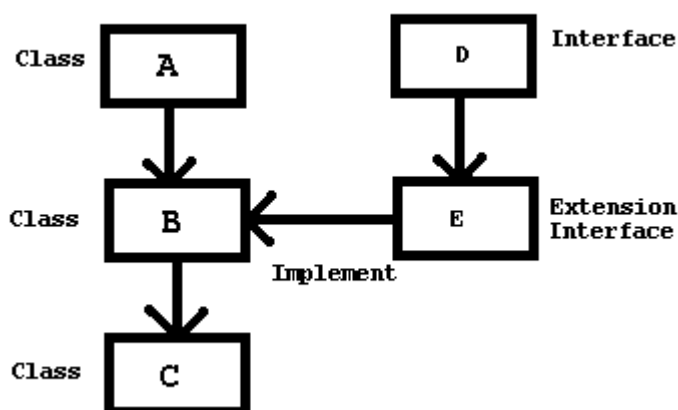
```
{  
    void fun2();  
    void fun3();  
}  
class msc implements myinter2  
{  
    public void fun2()  
    {  
        System.out.println(" Hello");  
    }  
    public void fun1()  
    {  
        System.out.println(" M.Sc");  
    }  
    public void fun3()  
    {  
        System.out.println(" Students");  
    }  
  
    public static void main(String args[])  
    {  
        msc ob=new msc();  
        ob.fun1();  
        ob.fun2();  
        ob.fun3();  
        System.out.println(" a is "+ob.code);  
    }  
}
```

Various Forms of interfaces →

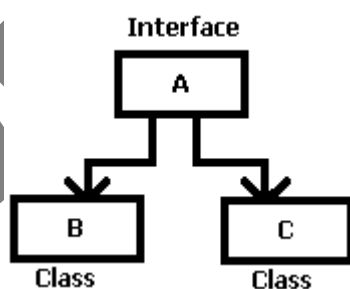
A



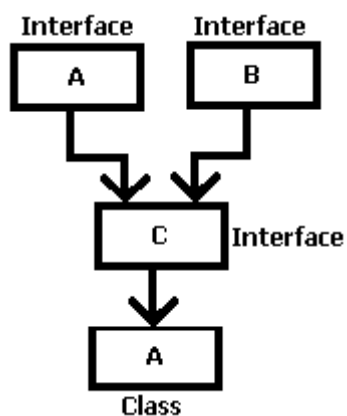
B



C



D



Packages:- Packages में विभिन्न class और interfaces को define किया जाता है। जिससे same name के classes का result disorder नहीं होता है।

TYPES OF PACKAGES:

1. Java API Packages:- इसमें विभिन्न functionality के अनुसार classes group में रखी गई है।

- b. Java.lang:-** इसमें language को support करने की विभिन्न classes होती है। जिसे compiler automaticly import कर लेता है। इसमें primitive types, string, math function, threads and exception की class रहती है।
- c. Java.util:-** इसमें utility class जैसे vector, hash tables, random number date etc.
- d. Java.io:-** इसमें input output को support करने वाली class रहती है।
- e. Java.net:-** इसमें networking और socket programming से related class रहती है।
- f. Java.applet:-** इसमें applet create करने व implement करने की विभिन्न class रहती है।

2. User Define Package:- User अपनी आवश्यकतानुसार स्वयं का package भी बना सकता है।

Step1 : जिस नाम का package है उस नाम की directory create करें।

Step 2: program में पहला statement package <packgename> लिखेंगे।

Step 3: Classes को define करेंगे।

Step 4: Directory में file को save करेंगे।

Step 5 : Java के द्वारा file को compile करेंगे।

Importing Package:- एक packages की classes को दूसरे package में उपयोग करने के लिए import statement का उपयोग करते हैं।

Syntax: import package.[subpackage.*]classname;

Classpath Variable:- यह java का environmental variable होता है, जिस उपयोग class files को path को define करने के लिए किया जाता है। Java का compiler और interpreter class path enviremental variable में define किये गये path में class files को search करता है। By default java का compiler और interpreter class file को current directory में search करता है। Class path में package की parent directory तक का path दिया जाता है।

package pack1;



public class pro

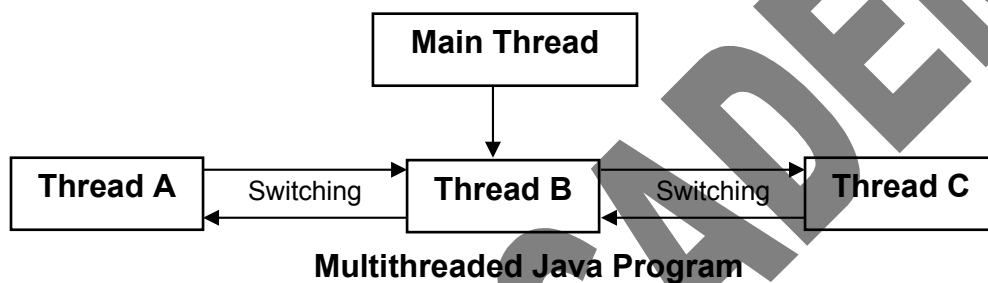
```
{  
    int x=5;  
    private int x_pri = 10;  
    protected int x_pro = 15;  
    public int x_pub = 20;  
    public pro()  
    {  
        System.out.println(" Base class ");  
        System.out.println(" x= "+ x + " x_pri " + x_pri + " x_pro " + x_pro + " x_pub " +  
x_pub);  
    }  
}
```

UNIT-IV

MULTITHREADING:- Multithreaded program में दो या दो से अधिक part होते हैं, जो concurrently (एक साथ) run होते हैं। Program का प्रत्येक part को thread कहलाता है।



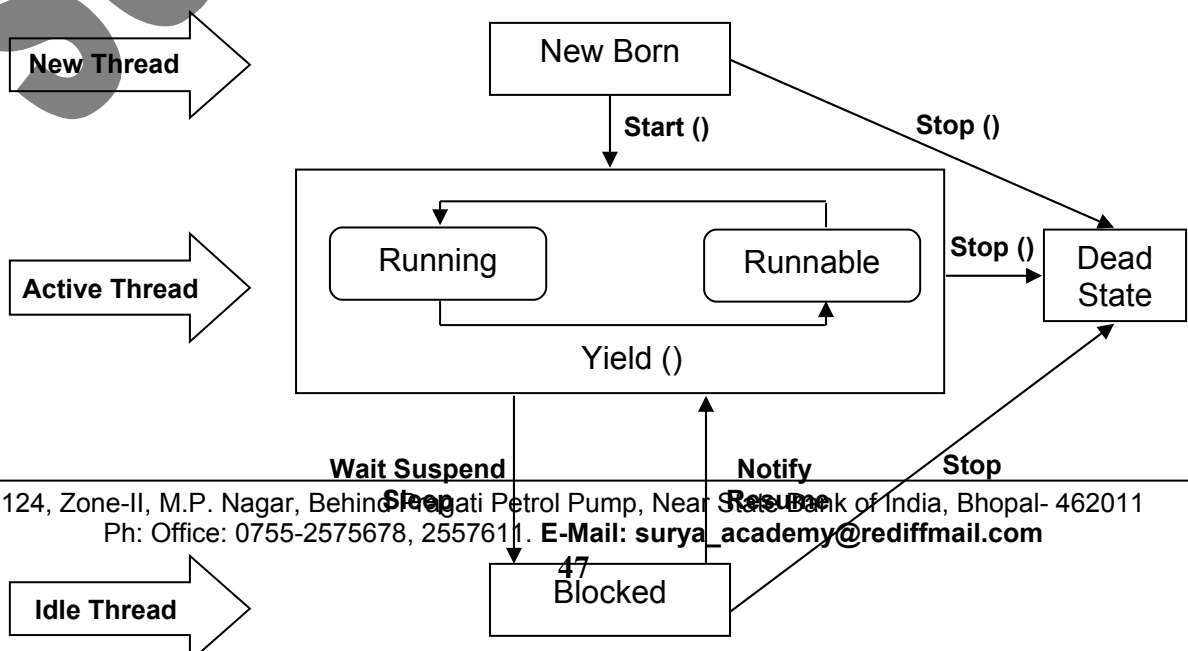
- प्रत्येक thread का execution के जिल अलग से path होता है, ये thread एक address space को share करते हैं, इस लिए ये light wight के होते हैं।
- Multitasking के special के रूप को multithreading कहते हैं।
- Multithreading के द्वारा efficient program लिख सकते हैं, जिसमें CPU का अधिकतम उपयोग हो सकता है और उसका idle time minimum हो जाता है। जो network environment के लिए बहुत आवश्यक है।
- सभी thread एक साथ Run होते हैं और resowise को एक साथ share करते हैं। Java का interpreter threads के बीच के switching को control करता है। जिससे सभी thread एक साथ run होते दिखाई देते है।
- किसी program में Multithreading दो प्रकार से कि जा सकती है।



- To Extend the Thread Class
- To Implement the Runnable Infer face.

LIFE CYCLE OF A THREAD or Thread Model :- एक thread की निम्न state होती है।

- New Born State
- Runnable State
- Running State
- Blocked State
- Dead State





THREAD CLASS:- Thread class java.lang को handle करने की विभिन्न methods होती है।

THREAD CLASS METHODS:

- **start ():-** इसके द्वारा thread new born state में आता है।
- **run ():-** यह thread का entry point होता है।
- **sleep ():-** इसके द्वारा thread दिये गये time तक block (Suspend) हो जाता है।
- **getpriority ():-** किसी thread की priority जानने के लिए इस methods का उपयोग करते हैं।
- **get name ():-** यह current thread का नाम return करता है।
- **stop ():-** इसका उपयोग thread को terminate करने के लिए किया जाता है।
- **is alive ():-** अगर कोई thread running state में है, तब यह method true return करती है।
- **join ():-** इस methods के द्वारा current thread के terminate होने तक thread wait करता है।

RUNNABLE INTERFACE:- Runnable interface से thread create करने कि निम्न step है।

Step1:- Runnable interface को implement करके एक class बनायेगे।

Step2:- run () method को implement करेगे।

Step3:- Thread class का एक object बनाये जिसमें उस class का object pass करेगे जो runnable interface को implement करके बनाया गया है।

Step4:- Thread को run करने के लिए thread class की start () methods को call करेगे।

Example:- class my extends Thread

```
{  
    public void run()  
    {  
        for (int i=1;i<=5;i++)  
        {  
            System.out.println(" \n From thred my : "+i);  
        }  
    }  
}
```



```
}  
System.out.println(" Exit from My");  
}  
}  
class my1 extends Thread  
{  
    public void run()  
    {  
        for (int i=1;i<=5;i++)  
        {  
            System.out.println(" \n From thred my1 : "+i);  
        }  
        System.out.println(" Exit from my1 ");  
    }  
}  
class my2 implements Runnable  
{  
    public void run()  
    {  
        for (int i=1;i<=5;i++)  
        {  
            System.out.println(" \n From thred my2 : "+i);  
        }  
        System.out.println(" Exit from my2");  
    }  
}  
class thr  
{  
    public static void main(String args[])  
    {  
        my obj1=new my();  
        my1 obj2=new my1();  
        my2 obj3=new my2();  
        Thread t=new Thread(my2);
```



```
obj2.start();  
obj1.start();  
  
t.start();  
}  
}
```

THREADS PRIORITY :- Threads priority का उपयोग thread scheduler के द्वारा होता है जो ये decide करता है कि कौन सा thread कब run होगा।

- प्रत्येक thread के लिए CPU के द्वारा दिया गया time विभिन्न factors पर depened करता है।
- High priority का thread lower priority के thread को preempt कर सकता है।
- किसी thread की priority set करने के लिए **setPriority ()** method का उपयोग होता है, जो thread class की member है।

Syntax:- final void setPriority (int level);

Level:- MIN_PRIORITY or MAX_PRIORITY के मध्य होनी चाहिए।

- Currently यह values 1 to 10 है।
- Threads की default priority NORM_PRIORITY में specify रहती हैं। जो currently 5 है।
- ये सभी priority threads class में find variables के रूप में store है।
- किसी threads कि current priority मालूम करने के लिए **getPriority ()** method को call किया जाता है।

Syntax:- final int getPriority ();

Example in Thread Priority:- class pr extends Thread

```
{  
    public void run()  
    {  
        for (int i=1;i<=5;i++)  
        {  
            System.out.println(" \n From thred my : "+i);  
        }  
        System.out.println(" Exit from My");  
    }  
}  
  
class pr1 extends Thread  
{  
    public void run()
```



```
{
for (int i=1;i<=5;i++)
{
    System.out.println(" \n From thred my1 : "+i);
}
System.out.println(" Exit from my1 ");
}
}
class pr2 extends Thread
{
    public void run()
    {
        for (int i=1;i<=5;i++)
        {
            System.out.println(" \n From thred my2 : "+i);
        }
        System.out.println(" Exit from my2");
    }
}

class prthr
{
    public static void main(String args[])
    {
        pr obj1=new pr();
        pr1 obj2=new pr1();
        pr2 obj3=new pr2();
        obj2.setPriority(Thread.MAX_PRIORITY);
        obj3.setPriority(obj1.getPriority()+1);
        obj1.setPriority(Thread.MIN_PRIORITY);
        obj1.start();
        obj2.start();
        obj3.start();
        System.out.println(" Exit from Main " );
    }
}
```



}

}

THREAD SYNCHRONIZATION:

जब दो या दो से अधिक thread एक sheared resources को access करते हैं। तब हमारी resquirment यह है कि एक बार में एक thread ही उस sheared resources को एक समय में उपयोग कर पाये त बवह process जिसके द्वारा यह प्राप्त किया जा सकता है, उसे synchronization कहते हैं।

synchronization की key monitor होती है। Monitor एक object होता है, जिसका उपयोग एक manuldy tack या mute के रूप में किया जाता है। जिससे दिये गये समय में एक thread ही shared resources को use कर पाता है।

Object को monitor में enter होने के लिए synchronized methods को call करते हैं, जब तक thread synchronized methods में रहता है तब दूसरे thread उस instance के लिए wait करते हैं।

Synchronization के लिए दो methods का उपयोग किया जाता है।

1. किसी methods को synchronized बना दिया जाए।

Example:-

//Example of Synchronized Statement

```
class mycl
{
    void call(String s) {
        System.out.print(" * " + s);
        try {
            Thread.sleep(1000);
        }
        catch(InterruptedException e) {
            System.out.println(" Interrupted ");
        }
        System.out.println(" * ");
    }
}
```

class caller implements Runnable {

```
    String s1;
    mycl obj1;
    Thread t;
    public caller(mycl obj2,String s)
    {
```



```
obj1=obj2;s1=s;
t=new Thread(this);
t.start();
}
public void run()
{
    synchronized(obj1) //To synchronized block
    { obj1.call(s1);
    }
}
}
class synchthr {
    public static void main(String args[])
    {
        mycl obj1=new mycl();
        caller obj2=new caller(obj1,"Hello");
        caller obj3=new caller(obj1,"BCA");
        caller obj4=new caller(obj1,"Students");
    }
}
```

Exception in Java

- Exception एक object होता है, जो program के code में आने वाली error को describe करता है।
- Program के वो statement जिन्हें हमको exception के लिए monitor करना है। try block में लिखे जाते हैं।
- जब कोई exception आता है तब वह throw हो जाता है तब catch block कर लेता है और अपनी coding के अनुसार handle करता है।
- Java का runtime system, system के द्वारा generated exception को automatically handle कर लेता है।

Syntax:

```
try {
    // code
}
catch(exceptionclass obj1)
{
```



```
}  
catch(exceptionclass obj2)  
{  
}
```

finally Block:- Finally block try/catch के execute होने के बाद execute हो जाता है। try block में run time error हो या ना हो finally block हमेशा execute होता है। finally block को try block के बाद में या last catch block के बाद add किया जाता है।

Uses of finally block:

1. Closing File
2. Releasing System Resources

Syntax:

```
(1) try {  
    // code  
}  
catch(exceptionclass obj1)  
{  
}  
catch(exceptionclass obj2)  
{  
}  
finally  
{ }
```

```
(2) try {  
    // code  
}  
finally  
{ }
```

Ex.: //Example and demo. of finally block

```
class myexc  
{  
    public static void main(String args[])  
    {  
        int a=10,b=5,c=5,x;  
        try{  
            x=a/0;
```



```
}  
catch(ArithmeticException p)  
{  
System.out.println(" Division by zero" + p );  
x=b/0;  
}  
finally  
{  
int y=a/(b+c);  
System.out.println(" y= " +y);  
}  
System.out.println(" End of Program ");  
}  
}
```

COMMON JAVA EXCEPTION

- ❖ Arithmetic Exception:- इसका उपयोग arithmetic से सम्बंधित exception के लिए किया जाता है। जैसे 2/0
- ❖ Array Store Exception:- जब किसी array में wrong type का data store किया जाता है।
- ❖ Array Index Out of Bounds Exception:- जब किसी array का wrong index दिया जाता है।
- ❖ File Not Found Exception:- जब किसी ऐसी file को access किया जाता है जो पहले से बनी नहीं है।
- ❖ Null Pointer Exception:- जब string और number के बीच का conversion fail हो जाता है।

Thread Exception:- Interrupted Exception:- जब कोई एक thread दूसरे thread के द्वारा interrupted होता है।

- **Illegal Thread State Exception:-** जब request किया गया operation thread की current state के साथ compatible नहीं होता।
- **Thread Death:-** जब thread को killed किया जाता है।
- **Illegal Argument Exception:-** जब किसी methods को call करते समय wrong argument pass किये जाते हैं।

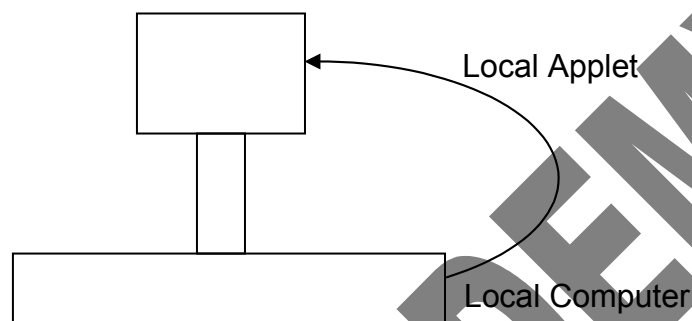
UNIT - V

Applet:- Applet internet पर transmit होने के लिए बनाई गई application होती है। जो किसी java compatible web browser पर execute होती है। ये किसी web document के द्वारा execute होती है और internet server पर access व transport होती है।

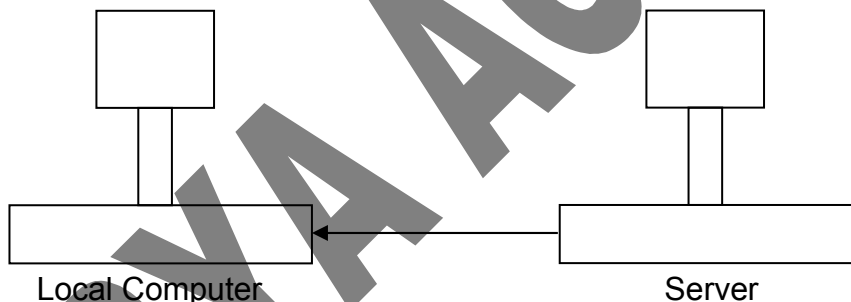
- Applet के द्वारा asthmatic operation, graphics display करना, sound को play करना user के द्वारा input लेना, animation create करना और game play करना किया जा सकता है।
- Applet दो प्रकार की होती है।

1. Local Applet
2. Remote Applet

1. Local Applet:- ऐसी applet जो local system पर store रहती है, local applet कहलाती है। इसलिए internet की आवश्यकता नहीं होती है।



2. Remote Applet:- Remote applet किसी और के द्वारा developed कि जाती है और internet के through किसी remote computer पर store रहती है, इसके लिए internet connection की आवश्यकता होती है और HTML page पर applet का address देने के लिए codebase attribute की आवश्यकता होती है।



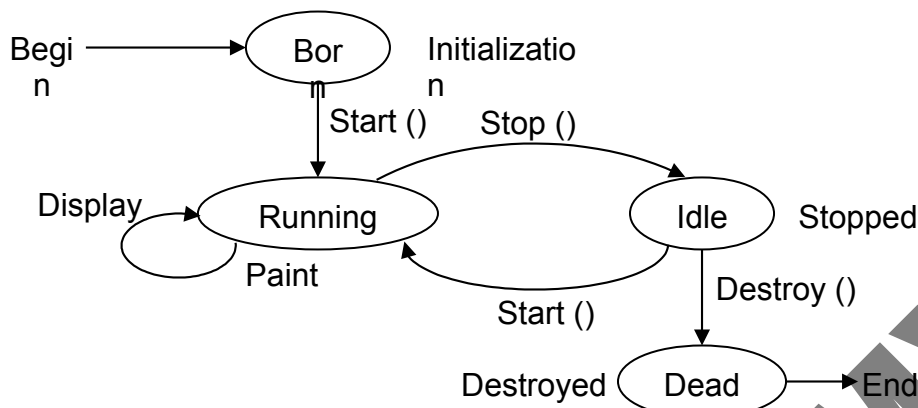
Applet Vs Application:

- Applet में main method नहीं होती है।
- Applets independnet run नहीं हो सकते ये किसी web page के द्वारा run होते हैं।
- Applets किसी local computer से file को read और write नहीं कर सकते हैं।
- Applets network पर किसी दूसरे sever से communicate नहीं कर सकती है।
- Applet के द्वारा local computer का कोई program run नहीं हो सकता।
- Applets दूसरी languages जैसे C या C++ की librares को उपयोग नहीं कर सकते हैं।

Applet Life Cycle:- एक applect कि निम्न state होती है।

- Born or Initialization
- Running
- Idle

➤ Dead or Destroyed



- 1. Initialization State:-** इस state का उपयोग applet के लिए object create करने के लिए value को initialize करने के लिए, font और images को load करने के लिए तथा colour को set करने के लिए किया जाता है।
- 2. Running State:-** जब system applet class कि start () method को call करता है तब applet running state में enter होती है, इस method को बार-बार भी call किया जा सकता है। जबकि init () को एक बार ही call किया जा सकता है।
- 3. Display State:-** जब Applet में कुछ output operation screen पर perform किये जाते हैं। तब यह state running state के बाद call हो जाती है, इसके लिए paint () method का उपयोग होता है।
- 4. Idle State:-** जब किसी applet को running state में कुछ समय के लिए छोड़ दिया जाता है तब वह उसकी idle state होती है।
- 5. Dead State:-** जब किसी applet को memory से हटा दिया जाता है, तब वह dead state में चली जाती है। जब हम किसी webbrowser से quit करते हैं तब destroy () method automatically call हो जाती है।

Creating an Executing Applet:

Step 1:- Create a class which extends applet class

```

Ex : import java.awt.*;
import java.applet.*;

public class myapp1 extends Applet {
    public void paint(Graphics g)
    {
        g.drawString("Hello BCA.",120,30);
    }
}

```



Step 2:- Applet को compile करने के बाद हमें myapp1.class executable applet प्राप्त होगी।

Step 3:- Create a web page

```
<html>
```

```
<body>
```

```
<applet code="myapp1" width=500 height=200>
```

```
</applet>
```

```
</body>
```

```
</html>
```

Appletviewer या किसी java के compatible web browser से .html file को run करेंगे।

HTML TAG :-

1. **<Head>:-** इस tag का use heading देने के लिये किया जाता है, इसके अंदर title tag का use किया जाता है।

```
<head> heading </head>
```

2. **Heading Tags:-** Heading के format में text देने के लिये H1 से H6 text का use करते हैं।

Syntax: <H1> Text </H1>

```
<H2> Text </H2>
```

EX: <H1 Align= "Center"> Hello </H1>

3. **
:-** इसका use किसी web page में एक blank line insert करने के लिये किया जाता है।

Syntax:

Ex: <Body>

Surya
Academy

```
</Body>
```

Surya
Academy

4. **Body Tag:-** Body tag का use विभिन्न tags and text देने के लिये किया जाता है।

Attributes of <Body>

alink: specify the color of active link.

rgb (X,X ,X) / XXXXXXXX / color name.

Background: Give the file name of an image for background.

Link: Specify the color of all links in the document.

Text: Specify the color of text in the document.

vlink: Specify the color of visited link in the document.

5. **Text Style Tags:-** HTML के text को विभिन्न formate में देने के लिये विभिन्न प्रकार के tag use होते हैं, जो निम्न है:-

- i. ** --- Tag:-** इसके द्वारा text को bold कर सकते हैं।
 - ii. **<I> Italic Tag:-** इसके द्वारा text को italic कर सकते हैं।
 - iii. **<U> Underline Tag:-** इसके द्वारा text को underline कर सकते हैं।
- 6. :-** इसका use text का size colour and font name देने के लिये किया जाता है।
- Syntax:** Text
- Ex:** Hello M.sc
- A size of font can be up to 1 to 7 or – 7 to 1
 - Default size is 3
- 7. <Center Tag>:-** इसका use text का center alignment set करने के लिये किया जाता है।
- Syntax:** <center> Text </center>
- 8. <Link Tag>:-** दो या उससे अधिक web pages को link करने के लिये <a> anchor tag का use करते हैं।
- इस tag को “Hyper link tag” भी कहते हैं।
- EX:** Click here
- └───────────> Full path
- For hyperlink with image:
- MSc course

Getting input from the User

Event Delegation Model:

- ❖ Applet event driven program होते हैं।
- ❖ Command event mouse, keyboard और push button और text field के द्वारा generate होते हैं।
- ❖ Java में events java.awt.event package के द्वारा support किये जाते हैं।
- ❖ Event एक object होता है, जो sources की state में हुए बदलाव को describe करता है। प्रत्येक event की अपनी registered methods होती है।
- ❖ Source एक object होता है जो event generate करता है। एक source एक से अधिक type के event generate कर सकता है।
- ❖ Sources event generate करके listeners पर भेजता है।

Syntax:- `Public void addtypelistener (Typelistener)`
यहाँ पर `type` event का नाम है तथा `e` `eventlistener` का reference है।

Event Listener:



- ❖ Listener एक object होता है जो event के होने पर notification बताता है, इसकी दो requirement होती है।

1. **First:-** यह specific type के event के लिए registered होना चाहिए।

2. **Second:-** इसमें event को relieve करने की process implement होनी चाहिए।

कुछ event listener interface निम्न है:-

Interfaces	Description
❖ ActionListener	इसके action event के लिए एक method define होती है।
❖ ComponentListener	इसमें चार method होती है, जो component को hide, move, show और resized कर सकते हैं।
❖ ItemListener	इसमें एक methods होते हैं जो item में changes करने को recognized करती है।
❖ MouseListener	इसमें five methods होती है जो mouse के click, pressed, released, move, component में enter होने को recognized करती हैं।

Event Class:

- ❖ सभी events के लिए event object एक super class होता है।
- ❖ सभी AWT events के लिए awtevent super class होती है। जो delegation model को handle करती है।
- ❖ Java.awt.event package में विभिन्न event defined है जो user interfaces के द्वारा henerate होते हैं।

Java.awt.evnet की विभिन्न event class निम्न है:-

Event Class	Description
② Actionevent	जब किसी button को press किया जाता है या list item को double click या किसी menu item को select किया जाता है, तब यह event जनरेट होता है।
② AdjustmentEvent	जब किसी scrollbar को manipulate किया जाता है। तब यह event जनरेट होता है।
② ComponentEvent	जब किसी component को hide, move, resize या visible किया जाता है, तब यह event जनरेट होता है।
② ContainerEvent	जब किसी component को container से add या remove किया जाता है। तब यह event जनरेट होती है।
② FocusEvent	जब किसी component पर focus आता है या जाता है। तब यह event जनरेट होता है।



- ② **ItemEvent** जब किसी check box या list के item को click किया जाता है तब यह event junrate होता है।
- ② **KeyEvent** Key bord के द्वारा जब input recive होता है तब यह event junrate होता है।
- ② **MouseEvent** जब mouse को orage, move, click, press या release किया जाता है तब यह event junrate होता है।
- ② **TextEvent** जब किसी text field या text area में कोई changes या बदलाव किये जाते हैं। तब यह event junrate होती है।
- ② **WindowEvent** जब किसी window को activated, close, open, deactivated या Quit किया जाता है, तब यह event junrate होता है।

User Interface:

- ② **Textfield:-** Textfield class के द्वारा single line text की entry के object create किये जाते हैं। इसमें arrow key, cut/past key के द्वारा editing कि जा सकती है।

Ex.: TextField Txt1 = new TextField ();
 TextField Txt2 = new TextField (25);

Textfield Method:

1. **gettext ():-** यह textfield के text (string) को return करता है।
Syntax:- String text.gettext ();
2. **Set Text ():-** To set the text.
Syntax:- void tst.set text (string);
3. **Setechochar ():-** यहाँ वो character दिया जाता है जो textfield में character देते समय display हो।
Syntax:- SetEchochar (char ch);
 SetEchochar ("*");

Text Area ():- Textarea control का उपयोग multiline values को देने के लिए किया जाता है।

Label:- Label एक object होता है, जिसका उपयोग string को display करने के लिए किया जाता है।

Font Class:- Font class का उपयोग font का नाम देने उसका style और size देने के लिए किया जाता है।

Ex : //Getting input from the user



```
import java.awt.*;
import java.applet.*;
import java.awt.event.*;

public class appinput extends Applet implements ActionListener {
    TextField num1,num2;
    Button ok,ex;
    String msg="";
    public void init()
    {
        Label no1=new Label("Number I : ");
        Label no2=new Label("Number II : ");
        num1=new TextField(4);
        num2=new TextField(2);
        ok=new Button("Calc");
        ex=new Button("Exit");
        add(no1);
        add(num1);
        add(no2);
        add(num2);
        add(ok);
        add(ex);
        ok.addActionListener(this);
        ex.addActionListener(this);
    }
    public void actionPerformed(ActionEvent e)
    {
        String s=e.getActionCommand();
        String s1,s2,s3;
        int a=0,b=0,c=0;
        if (s.equals("Calc"))
        {
            s1=num1.getText();
            s2=num2.getText();
            a=Integer.parseInt(s1);
```



```
b=Integer.parseInt(s2);
c=a+b;
s3=String.valueOf(c);
msg=" sum is " + s3;

}
else
    msg=" Sorry";

repaint();
}

public void paint(Graphics g)
{
    g.drawString(msg,30,80);
}

}
```

Passing parameters in an applet

किसी HTML page में param tag द्वारा applet के लिए argument दिये जा सकते हैं। Applet इस tag के attributes को get parameters method के द्वारा access कर लेता है। यह method parameter की value को string object के रूप में return करता है। इन value को numeric के रूप में उपयोग करने के लिए हमें convert करना होता है।

Syntax: <Param Name = <Attribute name> value = Attributevalue>

Ex: import java.awt.*;

import java.applet.*;

public class appparam extends Applet {

String fname;

float lead;

int fsize;

boolean avail;

public void start()

{

String x;

fname=getParameter("fname");

if (fname==null)

fname="Courier";



```
x=getParameter("fsize");
try{
    if (x!=null)
        fsize=Integer.parseInt(x);
    else
        fsize=0;
}
catch(NumberFormatException e)
{
    fsize=-1;
}
x=getParameter("lead");
try{
    if (x!=null)
        lead=Float.valueOf(x).floatValue();
    else
        lead=0;
}
catch(NumberFormatException e)
{
    fsize=-1;
}
x=getParameter("available");
if(x!=null)
    avail=Boolean.valueOf(x).booleanValue();
}
public void paint(Graphics g)
{
    g.drawString(" Font Name: " +fname,0,10);
    g.drawString(" Font size :"+fsize,0,40);
    g.drawString(" Leading :"+lead,0,60);
    g.drawString(" Font :"+avail,0,85);
}
}
//Html file
<applet code=appparam width=400 height=300 >
    <param name=fname value=Helvetica>
    <param name=fsize value=16>
    <param name=available value=true>

</applet>
```