2 2 marks

# JAVA EVOLUTION

History :- java programing lang was developed by James gosling along with patick naughton, ed frank, chris warth, mike sheridan.
while they were working at ~~Sun~~ Sun micro System. in the year 1991 initialy it was named oak programing their was an oak tree that stood outside gosling office. later project went by Green & finaly renamed as java
Gosling like the coffee very much in a village called java which is in jakarta, indonesia. gosling that of naming the Project java. The project Started to develop an application for an digital devices such as dth television. & coal called oak but project was renamed as java in the year 1995.
following table shows evolution of java.

Scanned with CamScanner

| Periods | events took place. |
|---|---|
| 1. June 1991 | project for java lang was initiated with the name oak by james gosling & teams @ Sun micro System. |
| 2. 1992 | This lang was officially known as oak. |
| 3 1995 | Sun micro System released the first public implementation as java |
| 4. 1998-99 | java 2.0 was released in this version their was a support for various platform J2EE, J2ME |
| 5. 2006. | for marketing purposes, Sun renamed to J2 versions as JavaEE, JavaME, JavaSE |
| 6. 2007 | Sun released java as Open source software. |
| 7. 2010 | Oracle occured Sun micro System. |

java features :- following are features of java which make it as revolutionary

Scanned with CamScanner

programing language.

1. java can complied & interpreted - normally programing languages can be complied or interpreted. but java is a language which can be complied & inter -preted. i.e; java compiler first translates java source program. into a special code called byte code. then java interpreter interprets this byte code to obtain the equalvalent machine code. this machine code then directly executed to obtain output.

2. java is a platform independent & portable programing language - platform independence is the most exciting feature of java program that means progeams in java can be executed on varities of platform. this feature is based on goal :WORA (write once run anywhere) java support portability in 2 ways i.e java complier generates byte code which can be further used to obtain the corresponding

machine code. Secondly the premitive datatypes are used in java are machine independent

3. java is known OOP language — java is a true object oriented language as every thing in java is an object. in java all code & data lies within the classes related to objects.

4. java is robost & secure — java ensure the reliable code. datatypes are strictly check -ed at the complie time as well as run time. in java the memory managment technique is selected by garbage-collection techniques. this technique eliminates various memory mangement problems. The exception handling feature of java helps the programer to handle serious errors delicate without crashing the over all System.

5. java is duigned for distubted System — this feature is very much useful in network environment. in java 2 different object on can communicate this can be achived by RMI [remote method invocation]. This feature is very much

-useful in client server communication.

6. java is simple & small programming lang

java is very simple programing lang even though you have no programming background we can.

can learn this language very efficiently & quickly.

7. java is a multi treaded & intiactive language - java support multi treaded programing which allows programer to write a such program that can perform main task simultaneously this allows the programer to develop intiactive system

8. java is dynamic & extensiable programming leng - This language is capable of dynamicaly linking new class libraries, methods, objects. java also support the function written in c & c++ these fuction are called native methods.

How do java differ & from c & c++ :-

| C | C++ | Java. |
|---|---|---|
| 1. C language needs to be complied | 1. C++ language needs to complied | 1. Java is a language that get interpreted & Compile. |
| 2. C is platform dependent. | 2. C++ is platform dependent | 2. Java is a platform indep-ent. |
| 3. There is no concept of threading in C | 3. C++ does not support multi threading program-ming. | 3. Java support multi threading |
| 4. using C, the GUI application cannot be Creat-ed. | 4. C++ does not have facility to creath & impleme-nt the GUI | 4. Java one can design very intes-act & user frien-dly GUI |
| 5. C does not support database oriented applica-tion. | 5. database hand-ling using C++ is very complex & does not allow database Connecti-ve. | 5. Java servleti Can be created which can intu act with the database like MS access, oracle MySQL, so on. |
| 6. C cannot be embedded in scripting language | 6. C++ code cannot be embedded in any scripting lang-uage | 6. Java code can be embe-ded within scripting language by means of applet Program |

Scanned with CamScanner

| | | |
|---|---|---|
| 7. Their is no concept of inheritence | 7. C++ Support 3 types of inheritance | 7. java Supports 2 types of inheritance directly where as java indirectly Support multiple inheritance. |
| 8. C uses pointers | 8. C++ can use pointers | 8. java their is no concept of pointers |
| 9. C does not support templates | 9. C++ Supports templates. | 9. java does not support templates |
| 10. Their is no concept of class in C programing | 10. C++ Supports the concept class one can write progms without using class | 10. In java their must be atleast one class present |
| 11. C is most Simple | 11. C++ is Simple to use & Implement | 11. java is Safe & reliable. |
| 12 C can be complied on varities of compliers but on same platform | 12. C++ can be complied on varities of compliers but on same platform | 12. java can be complied using an unique compiler & interpreter |

Scanned with CamScanner

Java & world wide web :- ① world wide web is information retrieval system. design for inter networking environment. the information in this system is accessed through the web pages. user can navigate from one web page to another in search of the information on the web.

② There are various scripting languages used for designing web pages, the most commonly used scripting language html

③ Java can be easily used on the on web system. with the use of java pro -grams. the web is becoming more inter -active & dynamic similarly with the use of web we can run particular java pogram on any

④ Java communicate with ~~java~~ web using java applets

The communication b/w java applets & web occurs as follows.

a* The user sends the request for the duired webpage using web browsers Example - google, internet explor, mozilla firefox to the web servor.

the web server accepts this request & process its & sends the required document requested by the user.

b* The web page in the form of html tag is return to the user in his browser this html page contains <applt> which represents that corresponding web page is an applt

c* while processing the applt. the server converts the applet source code in the form of byte code.

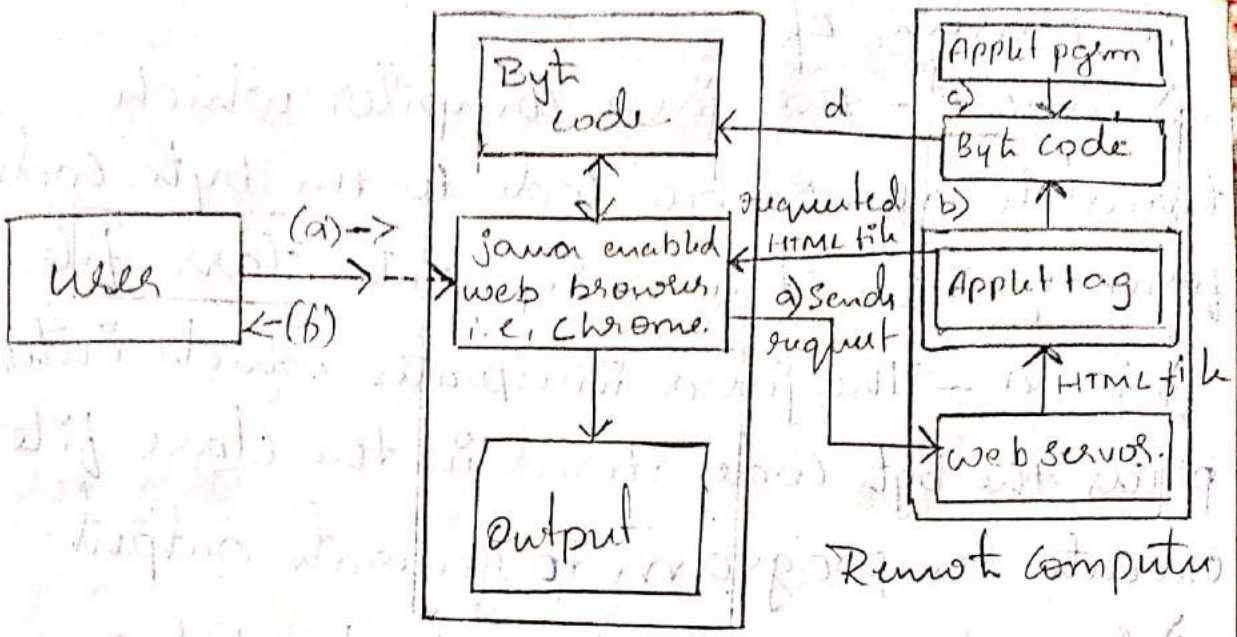
d* server also transfers the applt byte code to the user computer

e* java enabled web browserer interpret the byte code & provides the output

f* Now user can have interaction with applets.

Below figure illustrate interaction with the applets.

Scanned with CamScanner




java environment :-

1. java environment is made up of 3 things. they are development tools, classes & methods and java run time environment.
2. The java development tools consists java development tool kit [JDK]
3. The classes & methods are called application programing interface stored in a container called package. which forms a library called as java Standard library [JSL].
4. java runtime environment is supported by java virtual machine [JVM].

java development tool kit :- java development kit is nothing but, a collection of tools that are used for development & run time programs.

Scanned with CamScanner

it consists of

a) java c - the java compiler which translate the source code to the byte code form & stores it in a separate class file.

b) java - The java interpretes which intrs. prints the byte code stored in the class file & execute the program to generate output.

c) java doc - for creating the html document for documentation from source code file

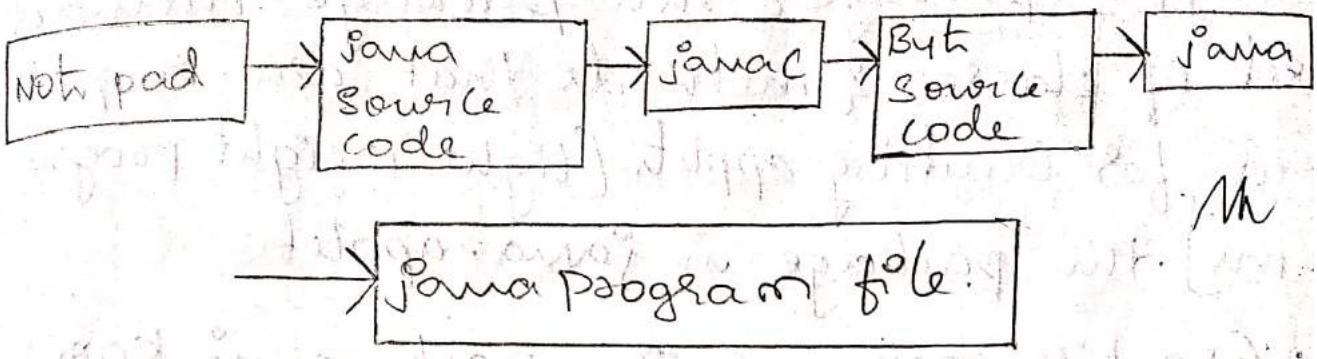
d) java h - it produces header files for the user of native methods.

e) jdb - The java debugger which helps to find the error in the programs.

f) java p - java disassembler which helps converting byte code into program discription

g) applet viewer - for executing the java applet.

## Execution process of java application programs using JDK :-




## API [Application programming interface] :-

The java API consists of large number of classes & methods. these classes & methods grouped into a package.

Example - in java their are 6 packages namely

* I/O package :- This package is know as java.io. the collections of classes in this package helps in input & output operations.

* Language support package :- This package is know as java.lang. the collection of classes & methods required for implement basic features of java.

* Networking package :- This package is know as java.net. the classes & methods of this package are required for communicating with other computer through internet.

Scanned with CamScanner

* Applet package :- This package includes set of classes & methods that are required for creating applets [light weight programs]. this package is java.applet

* Graphics package :- This package is known as java.awt. the classes & methods of this package are required to develop interactive graphical user interface [GUI]

* utility package :- This package is known as java.util. classes & methods of this package are required when we are using utility tools like System date, time & to generate random numbers automatically.

Simple java programs :-

```
/* This is my first program */
class first_pgm
{
    public static void main(String args[])
    {
        System.out.println("first pgm");
    }
}
```

Scanned with CamScanner

Not the entire logic of java program should be enclosed within a class.
the name of main class & name of program should be same. java program should have extension Dot java [.java]

An application programs :-

Example - To add 2 numbers

in this program 2 classes are created as shown below.

```
class Sum
{
    int a,b,c;
    void get_data(int x, int y)
    {
        a=x;
        b=y;
    }
    void show_data()
    { int c = a+b;
      System.out.println("Sum of 2 no = "+c);
    }
}
class Sumdemo
{
    public static void main(sting a[])
    {
```

Scanned with CamScanner

```
Sum obj = new Sum();
obj.get_data(14,13);
obj.show_data();
}
}.
```

The above java program contains 2 classes Sum & Sumdemo. The class Sum contains the method get_data for reading 2 values. Another class in Sumdemo which contains main method & main functions. i.e why class Sumdemo is called main class.

The above java program written in notepad & must be saved as Sumdemo.java. because name of the java file should be same of main class name from the class Sumdemo. we can access the methods of Sum class. they are get_data & show_data, to feed the data we should access get_data method. & show_data method displays the result

To access the 2 methods we need to create an object of the class Sum using

Scanned with CamScanner

new operator. construct memory for the class sum. the name of the object is also created which is used to access methods of class sum. The object name followed by dot operator followed by method name.
This Syntax access the contents of class.

Java Program Structure :- The program structure for the java is as shown below.

```
Documentation section/comments Section
package statements section.
import statements section.
class difination.
main method class.
{
  public static void main (String a[])
  {
    i/o statements/statements;
  }
}
```

Documentation Section - This Section provides information about source program like name of the java program. this section contains data which is not complied by java complier because data is enclosed

Scanned with CamScanner

under the Symbol /*.....*/ every thing written in this section are termed as comments.

package section - it consists of name of package by using the keyword package when we use classes from this package in our program then it is necessary to write the package statement at the begining of program.

import statement section - all the required Api can be imported by the import statement their some core package present in jdk java this packages includes classes & methods required for java programming. there packages can be imported in the program in order to use class & methods of program

class difination section - the class defination contains difination of class [generic class] this class mosmed contains the data & methods manupalating the data.

main method class - this is another class also called main class because it contains main method. this class

Scanned with CamScanner

can access the methods defined in other class.

Java tokens :- The smallest & individual & logical unit of java statements are called tokens. In java their are 5 types of tokens are used & there are

1. reserved keywords.
2. identifiers.
3. literals.
4. Operators.
5. Separators.

Reserved Keywords - The words which are used in programming to develop an application & they are case sensetive. Keywords used in java are listed below abstract, assert, boolean, byte, break, case, catch, char, class, const, continue, do, default, double, else, enum, extends, final, finally, finalized, for, float, goto if, implements, import, instanceof, int, interface, long, native, new, package, private, protected, public, ~~void~~ return, short, static, super, switch, synchronized, this, throw, throws, transien-

try, void, ~~volia~~, while, true, false, null, volatile.

Identifiers – are kind of tokens difined by the programer, they are used for naming the classes, methods, objects, Variables, interfaces & packages in the program.

following are the rules to be followed for the identifers.

1. The identifers can be written using alphabets, digit, underScore & dollar Sign.
2. No identifer Should start from a numb -er.
3. Allways start with alphabets
4. identifers Should not contain any other Special Charaits exept dollar Sign
5. Their Should not be Space b/w them.
6. The identifiers are Case Sensetive for example - int a; int A; here a & A treated as two different identifiers.
7. identifers can be of any lugth.

Scanned with CamScanner

Literals – are kind of variables that store sequence of characters for representing the constant value.

5 types of literals are

* integer literal
* floating point literal.
* boolean literal
* character literal.
* string literal.

as the name suggest the corresponding datatype, constants can be stored within their literals.

Operators – are the symbols used in the expression for evaluating them. Some of operators are.

* Arthimatic operator ('+', '-', '*', '#')
* Reg Relational operators ('==', '!=', '>=', '<=', >, <).
* ternary operator.
* Conditional operator ('?', ':')
* bit wise / logical operators ('&&', '||')
* Special operator (",")

Scanned with CamScanner

Separators - for dividing the group of Code & arrange them Systematically Certain Symbols are used which are know as Separator.

| Name of Separator | Description |
|---|---|
| 1. ( ) parenthesis | parenthesis are used to enclose the argument to passed to it. to define precedence in the expression. for Surrounding cast type |
| 2. { } Curly braces | are used for initializing array, for defining the block |
| 3. [ ] Square brackets | are used for declaring array type |
| 4 ; Semi colon. | is used to terminate the Statement |
| 5 , comma | are used to Separate the contents |
| 6 . period operator | period is used to Separate the package name |

Scanned with CamScanner

from sub packages.

Java Statements :- Statements are the executable sentences in java that are terminated by semi colon. Various types of statement are used in java are

1. empty statement - These statement do nothing but. used simplied as a place holder.

2. labled statement - are kind of statement which begins with the labels. the label must not be reserved keywords. or identifiers. that are already difine in the program, these statements are reachable using jump statements.

3. expression statement - majority statement in java are of expression type such as argument statement, pre & post increment, pre & post decrement, memory allocation statement are few examps of expression statement.

4 jump statement - are used to transfer the control to any desired location in the program.

Scanned with CamScanner

5. Selection Statement – The Selection Statements are the control Structure Statements which make use of the progra -ming constructs such as if, if else, Switch.

6. Iteration Statement – for Specifing the looping there types of Statements are used, the programming constructs Su as while, do while, for, are the example of iteration Statement.

7. Synchronization Statement – for handl -ing multi treading in java there Statement are used.

8. Guarding Statement – are used in order to handle errors or to avoid program from crashing. the try-catch() block Statement in java are called guarding Statements.

Implementing a java program :- Any java can be written using Simple text editor like notepad or word pad the file name should be same as

Scanned with CamScanner

the name of the main class used in the corresponding java program.
Note that the name of the program in first prog & class name in first prog, the extension to the file name should be .java. therefore we have to save our program by the name firstprog.java

Step 01 - Open notepad & edit the program as shown below.

```
class firstprog
{
    public static void main(String a[])
    {
        System.out.println("welcome to OO world");
    }
}
```

Note the main class is named as firstprog. So, the above program should be saved as firstprog.java.

Step 02 - Now using JDK tool installed in your machine. Set the path

Step 03 - using javac, compile the above program in the command prompt. as shown below.

```
javac firstprog.java↵
```

Scanned with CamScanner

Step 04- Now interpret the program using command as shown below

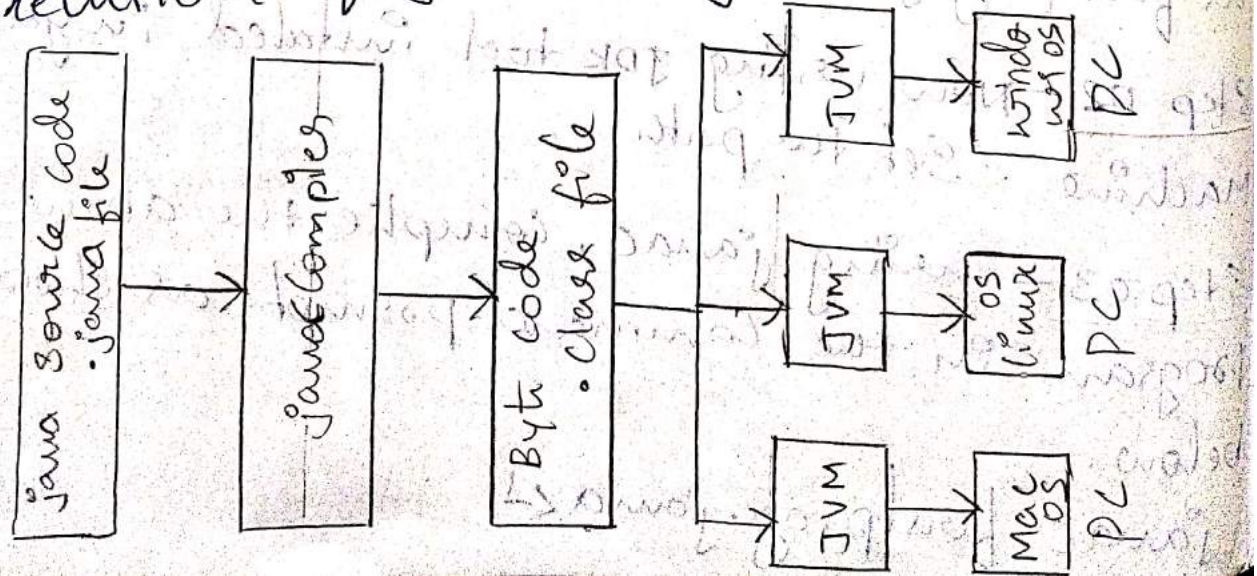
java firstprog ↵ [extension not required in interpreting]

Now output obtained on the output screen as shown below

Welcome to OO world.

Java virtual machine [JVM] :- it is set of software & program components as the name suggest. it is virtual computer that resides in a real computer. Java can achive its platform independent feature due to this JVM.

Before the understanding the concept of JVM. let us understand the creation & execution of java program.



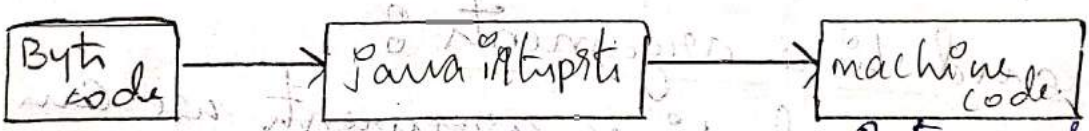

Scanned with CamScanner

1) when we want to writ any java pgm we writ the source code & store it in a file with extension .java

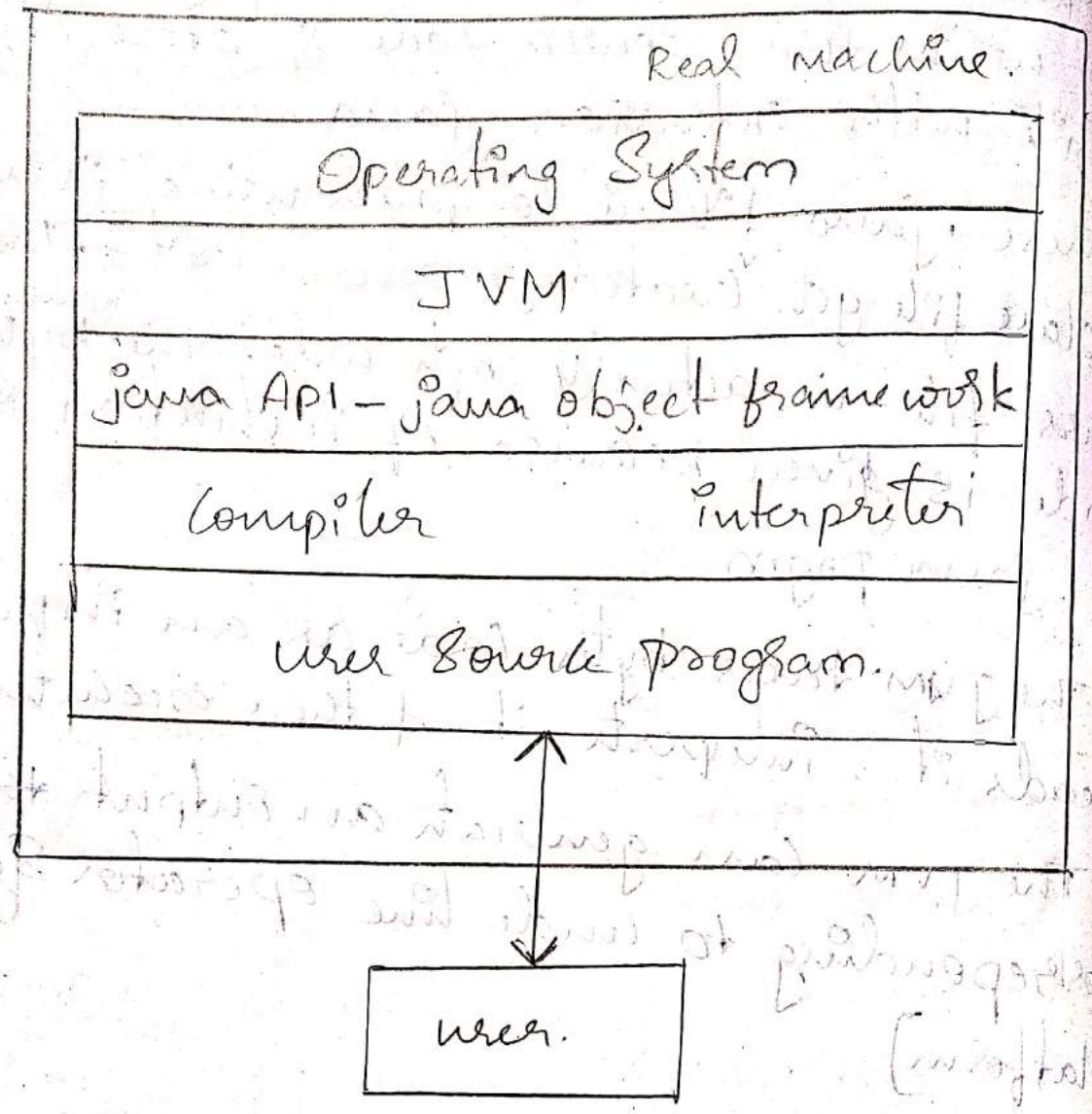
2) There .java file is compiled using javac & class file gets created. [viva Question]. This class file is actualy byte code. the byte code is given because of instruction set of java pgm

3) The JVM takes bytes code as an input reads it, interperts it & then executes it

4) The JVM can generate an output corresponding to underline operator System [platform]




The java Api works as an intermediate b/w the user pgm & virtual machine. The virtual machine acts as intermidate b/w java Api & operating System thus, user can intract with underlying platform




## Command line arguments :-

1. using command line arguments we can provide the input to the program during the execution.

Scanner is one of the inbuilt class that support providing input at the time of execution. by using the methods

- nextInt() for integer
- nextFloat() for floating point number.
- nextLine() for Strings.

Scanned with CamScanner

Below java program demonstrate CLA concept.

```
class CLAdemo
{
  public static void main (String args[])
  {
    Scanner sc = new Scanner (System.in);
    System.Out.println ("enter 2 numbers");
    int a = sc.nextInt();
    int b = sc.nextInt();
    int c = a+b;
    System.out.println("sum = "+c);
  }
}
```

while executing the program CLAdemo. java we can provide command line arguments which are the 2 inputs stored at a & b at the time of execution

After we compile this using javac command

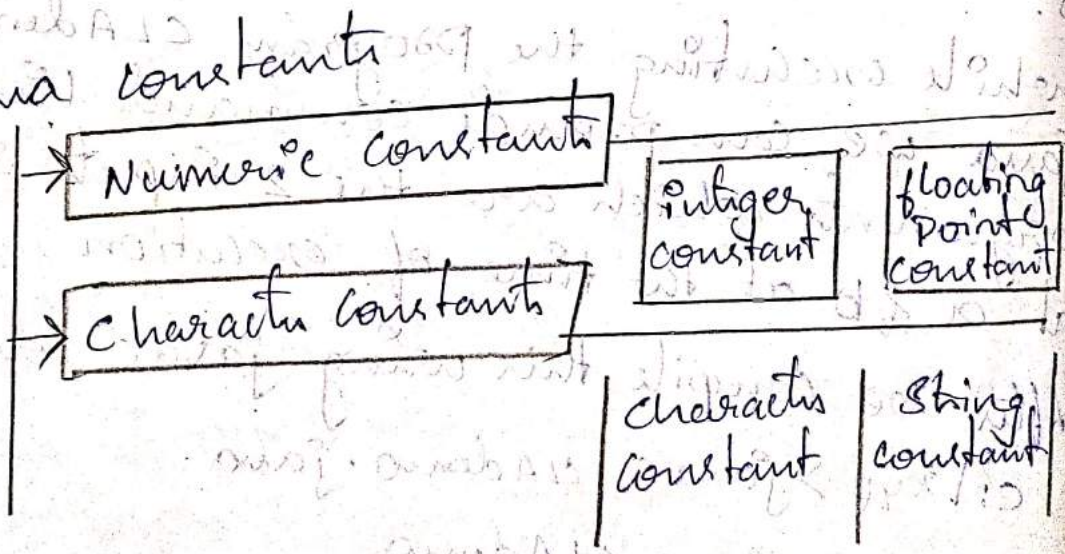
```
c:\xyz>javac CLAdemo.java.
c:\xyz>java CLAdemo.
enter 2 no! numbers.
   10   20
```

10 & 20 are the command line argument

Scanned with CamScanner

Programing style :- Java is a free form language which can be written in any style. The program can also written without indentation, Java system does not care for the manner in which program is written but, writing the code without proper indentation, is a bad programming practice.

Constants :- means fixed values that do not change during the execution of a program. In Java various types of constants are used are as showed below.




Numeric Constants - are used for representing the numerical value, the numeric values can be integer type / real

Scanned with CamScanner

type.

(a) Integer constant - Integer means sequence of digits (1 or more) without decimal point

Example : 12, 160, 01101

(b) Real constant - are the numbers with the decimal point. example: 85.45, 1.011

Character Constant - are non numerical values they can be either character constant or String constant.

(a) Character constant - A single character constant in enclosed within a pair of single quote.

Example : 'S', 'J'.

(b) String constant - A string is a collection of characters (character array) enclosed b/w double quotes.

Example : "Sahana", "xyz".

Variables :- A variable is an identifier that denotes storage location. A variable is a fundamental unit of storage in java. The variable are used in

Combination with identifiers, datatypes, operators & some values for initialization.
The variable must be declared before they must use.

Syntax: datatype name of variable = Const;

Example: int a = 30;
float 1 = 30.45;
Char init = 'J';
String name = "RJS";

following are some rules for variable declaration.

1. The variable name should not start with a digit.
2. NO special characters are allowed except underscore.
3. There should not be blank space in variable.
4. The variable name should not be a key word.
5. The variable name should be meaningfull.

Scanned with CamScanner

Datatypes :- are element of programming which informs the compiler about the type of command line arguments. i.e, input & output of a program.

In java various datatypes are used. they are listed below.

1. int - This is the most commonly used datatype. for defining integer type numerical data. The width of datatype is 4 byte. (32 bit). the declaration can be

int a, b;

2. float - To represent the real number, float datatype can be used, the width of the float datatype in java is same of int. i.e 4 bytes (32 bit). The declaration can be

float a, b;

3. Short - This datatype is also used for defining the signed numerical variables with the width of 2 byte

4. long - Some times when int is not sufficient for declaring some data then long datatype is used. The width is 8 bytes

5. Double - To represent the real numbers of large range that double datatype is used. it is width is 8 byte.

Scanned with CamScanner

6. Char - This datatype is used to represent character type of data. width of char in java is 2 byte

7. Byte - this datatype is infact smallest integer type its width is 1 byte

8. Boolean - is simple datatype which denote a value to be either ~~to~~ true or false. it width is 1 bit.

Scope of Variables :- Their are 3 types in Variables.

1. class Variable.
2. local Variable
3. instance Variable.

class Variables & instance Variables are declared inside the class.

The local Variable are declared inside the methods.

when the objects are instantiated at that time the instants Variables are created.

The instance Variable are associated with objects.

The class Variables are ~~gobal~~ global to the corresponding class. this Variable

Scanned with CamScanner

can be accessed by object of that class.
The local variables are not accessable to the method outside the method because their scope is limited within the curly brasses.

The area in which the variables are accessable is called scope

Below program illustrate 3 types of variable & its scope.

```
class exam
{
  String data, st_time, end_time;
  void get_data (string d, string st, string et)
  {
    data = d;
    st_time = st;
    et_time = et;
  }
  void display()
  {
    System.out.println("data & time of examin
    -ation is : "+data + st_time + et_time);
  }
}
```

```
class theoryexam.
{
  public static void main(String args[]);
  {
     exam OS = new exam();
     exam PEIC = new exam();
     exam java = new exam();
     OS.get_data("17-Apr", "2pm", "5pm");
     PEIC.get_data("19-Apr", "2pm", "5pm");
     java.get_data("23-Apr", "2pm", "5pm");
     OS.display();
     peic.display();
     java.display();
  }
}
```

Type casting :- It is the common practice to assign the value of one datatype element to another. Java allows the automatic conversion b/w these datatypes but. java performs type conversion b/w the comparable datatype elements. i.e if we assign an integer value to a float type

Scanned with CamScanner

element. then java automatically make this conversion. however

any datatype rather such a conversion is done with the comparitable datatypes elements.

Defination - Type casting means explicit conversion b/w incomparitable data types.

for casting the values we use the following syntax.

(type) variable name;

Example - float a = 7;
int b;
System.out.println("val of a is " + a);
b = (int) a;
System.out.println("val of b is " + b);

Special operator :- Their are 2 special operators.

1. Dot (.) operator
2. instanceof operator.

for determining whether the object belongs to particular class or not an instanceof operator are used.

Scanned with CamScanner

for example - ganga instanceof river. then it returns true otherwise false.

The Dot (.) operator is used to access instance variable & methods of class object.

for example - customer.name; customer.id;
obj.display(); , obj.Set_val();

Mathematical functions :- java provides a support for performing mathematical operation by means of mathematical function. the basic mathematical functions are defined in math class present in java.lang package. Various mathematical functions are

sin(double angle) -> returns the sin of angle in radians.

cos(double angle) -> returns the cosin of angle in radians

tan(double angle) -> returns the tan of angle in radians

asin(double angle) -> returns the inverse of sin value (i.e cosec).

Scanned with CamScanner

acos(double angle) -> returns the inverse of cosine value (i.e sec)

atan(double angle) -> returns the inverse of tan value (i.e cot)

pow(double a, double b) -> return the value $a^b$.

sqrt(double a) -> returns square root of a.

max(a,b) -> Computes the maximum of a & b.

min(a,b) -> computes the minimum of a & b.

log(double val) -> computes the logarithm value of a.

abs(val) -> Computes absolute value of val.

ceil(val) -> it returns the smallest whole number which is greater than or equal to val.

floor(val) -> returns the largest whole number which is lesser than or equal to val.

Labled loops :- (Break & continue). Some times when we apply loops we need to come out of the loop on occurence of particular condition, this can be achived

Scanned with CamScanner

by break statement.
following java progrein illustrate the use of break in the for loop.

```
class breakdemo
{
   public static void main (String x[])
   {
      for(int i=1; i<=20; i++)
      {
         if(i%10==0)
         {
            System.out.println("number"+i+
            "is divisible by 10");
            break;
         }
         else
         System.out.println("number"+i+"
         not divisible by 10");
      }
   }
}.
```

output -

Number 1 is not divisible by 10.

Scanned with CamScanner

Number 2 is not divisible by 10
Number 3 is not divisible by 10
Number 4 is not divisible by 10
Number 5 is not divisible by 10
Number 6 is not divisible by 10
Number 7 is not divisible by 10
Number 8 is not divisible by 10
Number 9 is not divisible by 10
Number 10. is divisible by 10.

In the above output the number after 10 will not be considered at all this is because when the control reaches at 10, if condition becomes true & then encounters break statement. this forces the control to take an exit from the for loop.

The Continue Statement :- This Statement is used to restart the current loop. if we want to jump to the outer loop then continue statement is used
Below program illustrates the use of continue statement in for loop.

Example :

```
Class dContinuedemo.
{
```

Scanned with CamScanner

```
public static void main (String x[])
{
System.out.println("In printing the odd
                    numbers");
for(int i=0; i<10; i++)
{
  if(i%2==0)
  continue;
  System.out.println(i);
}
}
}
```

Output:-

printing the odd no

1
3
5
7
9

Operators & expressions:- Various operators that are used in java are enlisted in the below table.

| Type | operator | meaning | Example |
|---|---|---|---|
| Arithmatic | + | addition. | c = a+b |
| | - | Subtraction | z = x-y |
| | * | multiplication | c = a*b |
| | / | divion | r = p/q |
| | % | modulo | k = m%n |

Scanned with CamScanner

| | | | |
|---|---|---|---|
| Relational | < | lesser than | a<b |
| | > | greater than | a>b |
| | <= | lesser than or equal | x<=b |
| | >= | greater than or equal | y>=z |
| | == | equal. | P==R. |
| | != | not equal to | S!=Q |
| logical | && | AND operator | p&&q |
| | \|\| | OR operator | x\|\|y. |
| Assignment | = | is assigned to | a=5 |
| increment | ++ | increment by 1 | i++ ++i |
| Decrement | -- | decrement by 1 | i-- --i. |
| conditional | ?: | Ternary operator where one expression denotes true condition & 2 expression denotes false condition. | (a>b)?a:b |

## Decision making, Branching & looping :

Programmer can take decision in their

Scanned with CamScanner

Program with the helps of control statement & control constructs. Various control constructs that can used in java

1. if Statement.
2. if else statement
3. while Statement
4. Do while Statement.
5. Switch case
6. for loop

if & if else Statement are branching control constructs.

if execute the statement enclosed only the condition is true where as if else statement is also a branching type. if execute the statement condition i.e true statement / false statement.

1. if statement is of 2 types.
   (a) simple if statement
   (b) compound if

Simple if Statement - the if statement is only one statement is followed by that if Statement

Syntax: if (condition)
        Statement;

Scanned with CamScanner

Example: if (i % 10 == 0)
System.out.println("no is divisb by 10")

Compound if Statement – if their are more than 1 Statement that can be executed when if condition is true. then it is called compound if statement.

Syntax: if (condition)
{
Statement 1
Statement 2
- - - - - -
- - - - - -
Statement n
}

Example: if (time != a)
{
Sy.out.pln ("Absent mark");
Sy.out.pln ("pay fine");
new fine = new-fine + 50;
Sy.out.pln ("fine payable = " + new fine);
}

2. if else Statement – In this Statement their are 2 sides of execution.

Scanned with CamScanner

Syntax : if (condition)
Statement 1 -> true statement
else
statement 2 -> false statement.

Example : if (a > b)
Sy.out.pln("a is big");
else
Sy.out.pln("b is big");

The if else statement can be of simple type or compound type

<u>while statement</u> :- while is another form of if statement which is used to have iteration of the statement for any number of times.

Syntax : while (condition)
{
Statement 1;
Statement 2;
.....
Statement n;
}

Example : int count = 1;
while (count <= 5)
{ Sy.out.pln("i am in line no " + count);

Scanned with CamScanner

}.

output :- I am in line no 1
I am in line no 2
I am in line no 3
I am in line no 4
I am in line no. 5.

Do-while :- This is similar to while statement but only difference b/w the 2 is that in case of do while statement. the statement inside the do while must be executed atleast once. this means that the statement inside the do -while body gets executed first. & then the while condition is checked. for execution of the statement.
where as in while statement first a fall the condition given in the while is checked first & then the statement inside the while body get executed when the condition is true.

Syntax :
```
do
{
statement 1;
statement 2;
. . . . .
statement n;
}
while (condition);
```

Scanned with CamScanner

Example :
```
int count=1, i=0;
do
{
  i=i+1;
  Sy.out.pln("the value of i = "+i);
  count++;
}
while(count<=5);
```

Output :-
value of i=1
value of i=2
value of i=3
value of i=4
value of i=5

Switch Case statement :- It is also called as single entry, single exit control construct an example that illustrate Switch Case statement is as shown below.

```
class SwitchCaseDemo.
{
  public.Satic void main(String x[])
  {
    char choic;
    System.out.pln("1:A");
    System.out.pln("2:B");
    System.out.pln("3:C");
```

Scanned with CamScanner

```
choice = (char) System.in.read ();
switch (choice)
{
  case '1': Sy.Out.pln("you selected A");
            break;
  case '2': Sy.out.pln("you selected B");
            break;
  case '3': Sy.out.pln("you selected c");
            break;
  default:
            Syet.out.pln("NOTA");
}
}
}
```

**for loop** :- for is a key word used to apply loops in the program. They are categorise -d into simple for loop & compound for loop.

Syntax: for (initialization statement; test condition; increment/decrement)

if required { }.

Example:

```
class forloopdemo
{
  public static void main (String a[])
```

Scanned with CamScanner

```
{
    for(i:0; i<=5; i++)
    System.out.println("The value of i:" +i);
    }
}
```

Output :- The value of i 1
The value of i 2
The value of i 3
The value of i 4
The value of i 5.

Scanned with CamScanner

7/1/20. 38 marks

UNIT-02.

# Classes, objects & methods, String and StringBuffer classes.

**Introduction :-** Object oriented programs are often easy to understand, correct & modify. Java is the OOP language. In any OOP language anything is encapsulated in a class. The objects are the methods of a class. So, that 2 objects can communicate with each other.

**Defining a class :-**

1. A class can be defined as an entity in which data & function [methods] put together.
2. The concept of class is similar to the concept of structures in C
3. A class is declared by using a keyword class.

The general form of class is as shown below.

Scanned with CamScanner

```
Class classname.
{
    type var 1, var 2 . . . . ;
    type method name (parameter list)
    {
        . . . . . .
        . . . . . .
    }
    type method name ( )
    {
        . . . . . .
        . . . . . .
    }
}
```

fields declarations :- (Variable)

1. The data lies within the class & datafields are accessed by the methods of that class.
2. The data fields are also called as Instance variables. ~~bec~~ of member variable because they are created when the object gets instantiated
3. for Eg :-

Scanned with CamScanner

```
class test
{  int a;
   int b;

}.
```

## Methods declaration:-

1. All the methods have the same general form as the method main(). most of the methods are specified as either static or public.
2. Java classes do not need the method main(). but, if we want particular class as a starting point for our program then the main method is included in the class.
3. The general form of method is

```
type methodname (parameter list)
{
- - - - -
- - - - -
}
```

4. The type specify the datype of data return -ed from the method; if the method does not return anything then its datatype must be void.
5. for returning the ~~object~~ data from the method the keyword return is used

Scanned with CamScanner

Creating objects :- objects are nothing but instances here. A block of memory gets allocated for this instances variable for creating objects in java the operator new is used

Example : test obj = new test ();

In the statement we instantiate one object obj. It can be represented graphically as shown as below.

Test

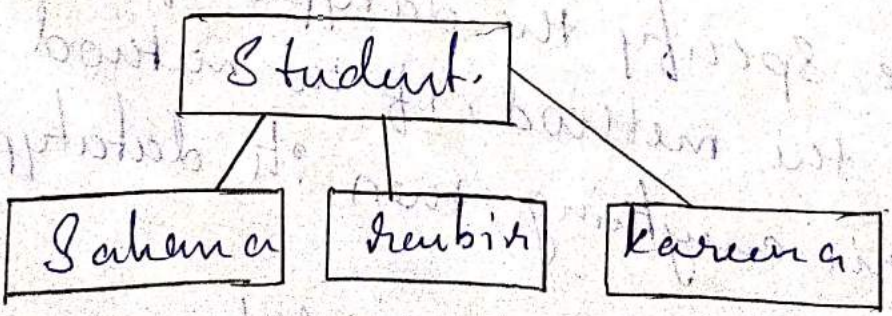
obj

we can create as many object as possible for a single class which is as shown below.

Student Sahana = new Student ();

Student Ranbir = new Student ();

Student Kareena = new Student ();




Scanned with CamScanner

## Accessing class members :-

1. There are 2 types of class members.

they are
data member, member function (methods)

2. These class members can be accessed using (.) Dot operator.

3. for accessing the data members the syntax is :-

```
object name . variable name = value;
Object name . methodname ();
objectname . methodname (parameter list);
```

4. for accessing the methods of the class. the Syntax is :-

```
object name . methodname ();
object name . methodname (parameth list);
```

for Example - obj°. radius = 10;
obj°. find_area (5);

Program example which makes use of class, methods & object.

```
class rectangle.
{
    int length, breadth;
    void find - area ()
    { int result = length * breadth;
      System.out.println ("The area of rectangle = " + result
                          + "sqrt");
    }
}.
```

Scanned with CamScanner

```
class rectdemo
{
    public static void main (String a[])
    {
        rectangle obj = new rectangle();
        obj.length = 10;
        obj.breadth = 20;
        obj.find_area();
    }
}.
```

Output :- The area of rectangle is 200 sqr units

Constructors :- The constructor is a specialized method used for initializing the data members of the class. Note that name of constructor & name of the class must be same.

The constructor is invoked whenever an object of its class gets created, it is called constructor because it creates the values for the data fields of the class.

The below Java program illustrates making use of a constructor.

```
Class rectangle
{ double length, breadth;
  rectangle()
```

Method :- kind of procedure to do some activity

```
{
    length = 25;
    breadth = 10;
}
rectangle (double l, double b)
{
    length = l;
    breadth = b;
{
void show_area ()
{
  double area = length * breadth;
  System.out.println ("The area of rectangle is"
                                    + area);
}
}
class boxdemo
{ public static void main (String a[])
  {
    rectangle obj1 = new rectangle ();
    rectangle obj2 = new rectangle (10,20);
    obj1.show_area ();
    obj2.show_area ();
  }
}
```

Scanned with CamScanner

## Properties of Constructor :-

1. Name of the constructor must be same as name of class.
2. The constructor should be declared in the public mode only
3. Constructor gets invoked automatically when object gets created.
4. The constructor should not have any return type even void datatype also
5. The constructor cannot be used as a member of union or structure.
6. The constructor can have default arguments
7. The constructor cannot be inherited but derived class (sub class) can invoke the base class (super class)
8. It can make use of new operator for allocating the memory. delete operator for releasing the memory.
9. multiple constructor can be used by the same class
10. When we declare constructor explicitly then we must declare the object of that class.

Scanned with CamScanner

## Method Overloading – Overloading is a mechanism in which we can use many methods having the same method names but, can pass different no of parameters of different types of parameters.

for Example – int, Sum (int a, int b)

double Sum (double a, double b)

int Sum (int a, int b, int c)

That means by overloading mechanism we can handle different numbers of parameters of different types of parameters by having same method name.

Below java program illustrate method overload.

```
class Sum
{
  Void Show_Sum (int a, int b, int c)
  {
    int Value = a+b+c;
    System.Out.pln("Sum of the integer = " + Value);
  }
  Void Show_Sum (double a, double b)
  {
    double Value = a+b;
    System.Out.println("Sum of the double = " + Value);
  }
```

Scanned with CamScanner

```
    void show_Sum (int a, int b)
    {
      int Sum = a + b;
      System.out.println ("Sum of =" + Sum);
    }
  }
class operation
{
  public static void main (String a[])
  {
    Sum add = new Sum();
    add.show_Sum(10, 11);
    add.show_Sum (2, 4, 8);
    add.show_Sum (2.4, 2.2);
  }
}.
```

Output :- Sum of = 21
Sum of integer = 14
Sum of double = 4.6

Scanned with CamScanner

Static members :- Can be static data number of static method. The static members are those members which can be accessed without using object.

following java program illustrate the use of static members.

```
class staticprogdemo
{
  static int a=10;
  static void fun(int b)
  {
    System.Out.println("b = "+b);
    System.Out.println("a = "+a);
  }
}
class Anotherclass
{
  public static void main(String a[])
  {
    System.Out.println("a = "+staticprogdemo.a);
    staticprogdemo.fun(20);
  }
}.
```

Output :- b = 20
a = 10.

Nesting of methods :- any method belonging to a class can be invoked using the object of that class but, some times we can invoke that method by simply using the name of it., this is called nesting of methods. following is the simple java program which illustrate the idea of nesting of methods.

```
class nestmetdemo
{
  void first_method()
  {
    System.Out.println("I am in first method");
  }
  void second_method()
  {
    first_method();
    System.Out.println("I am in second method");
  }
}
class nesttest
{
  public static void main(String a[])
  {
    nestmetdemo obj = new nestmetdemo();
    obj.second_method();
  }
}
```

Scanned with CamScanner

output :- I am in first method.
I am in second method.

Inheritance : Extending a class :- inheritance means taking some properties from parents for instance if your mother have blue eyes & your eyes are also blue then it is said that the colour of your eyes is inherited from your mother.
In java inheritance means derived class borrows some properties of base class, at the same time the derived class may have some additional properties. which are not there in the base class.
The old class is called the base class & the new class generated from the old class is called derived class.
The inheritance can be achived by incorporating one class into another using the keyword extends.
In oop, inheritance is referred as 'is-a' relation.
Various type of inheritance are
1. single level inheritance
2. multi level inheritance.
Java directly support the above 2 type of

Scanned with CamScanner

inheritance & java also supports another type of inheritance & it is
3. Multiple inheritance.

→ Single level inheritance -

* The class which is inherited is called the base class of super class, parent class, old class.
* The class that does the inheriting is called derived class, sub class, child class, new class
* The sub class can be derived as shown below.

```
class name of subclass extends name of superclass
{
  variable declaration;
  method declaration;
}.
```

Note that; the keyword extends represents that the properties of super class are extended to the sub class. thus, the subclass will now have the properties of its own class as well as properties of super class. following is a simple java program that illustrates the concept of single level inheritance

```
class A
{
  int a;
```

Scanned with CamScanner

```
void set_a(int a)
{
 this.a = a;
}
void show_a()
{
 System.Out.println("a = " + a);
}
}
class B extends A.
{
 int b;
 void set_b (int b)
 {
  this.b = b;
 }
 void show_b()
 {
  System.Out.println ("B = "+b);
 void main()
 { int c = a * b;
}   system.out.println("product is "+c);
class } SingleInheritDemo
{
 public static void main (String a[])
 {
  B obj_b = new B();
  obj_b.set_a(50);
  obj_b.set_b(60);
  obj_b.show_a();
  obj_b.show_b();
```

Scanned with CamScanner

obj-b.mul();

}

}

Output :- a=50

b=60

product is 3000.

--> Multi level inheritance -

is a kind of inheritance in which the derived class itself derives sub class further. In the following program we have created a base class A from which sub base B is derived. Then is a class C which is derived from class B, in the function main we can access any of the field in the class hierachy by creating the object of class C.

```
class A
{ int a;
  void set_a(int i)
  {
    a=i;
  }
  void show_a()
  {
    System.out.println("A="+a);
  }
}
```

Scanned with CamScanner

```
class B extends A
{ int b;
  void set_b (int j)
  {
     b = j;
  }
  Void show_b ()
  {
    System.Out.println ("B = "+b);
  }
}
class C extends B
{ int c;
  void set_c (int k)
  {
    c = k;
  }
  void show_c ()
  { System.Out.println("c = "+c);
  }
  void mul()
  {
    int res = a*b*c;
    System.Out.println("value of product = "+c);
  }
```

Scanned with CamScanner

```
}
class multinheritdemo.
{ public static void main(String s[])
  {
    c obj-c = new c();
    obj-c.seta(5);
    obj-c.set_b(6);
    obj-c.set_c(10);
    obj-c.show_a();
    obj-c.show_b();
    obj-c.show_c();
    obj-c.mul();
  }
}
```

Output :- a=5
b=6
C=10
Value of product = 300

Overriding methods :- method Overriding is a mechanism in which a sub class inherits the methods of super class & sometimes the sub class modifies the ~~method~~ implementation of a method defined in

a Super class.
i.e, the method of Super class which gets modified in the Sub class. As the same name & type signature.
The Overriden method must be called from the Sub class.
Below Java program illustrates the concept of method overriding.

```
class vehicle
{
  void run()
  {
    System.Out.println("Vehicle is running");
  }
}
class Bike extends Vehicle.
{
  void run()
  {
    System.Out.println("Bike is running Safely
                        & Smoothly");
  }
}
class Overridden0.
{
public static void main (String s[])
  {
    Bike b = new bike();
```

Scanned with CamScanner

```
        b.run();
    }
}
```

Output :- Bike is running safely & smoothly

```
class Bank
{   int getrateofintrest()
    {   return 0;
    }
}
class SBI extends Bank
{   int getrateofintrest()
    {   return 6;
    }
}
class Axis extends Bank
{   int getrateofintrest()
    {   return 7;
    }
}
```

Scanned with CamScanner

```
class ICICI extends bank.
{ int getrateofinterest()
  { return 8;
  }
}
class test
{
  public static void main(String arg[])
  {
    SBI s = new SBI();
    Axis a = new axis();
    ICICI i = new ICICI();
    s.getrateofinterest();
    a.getrateofinterest
    System.out.println("interest in SBI bank = " +
                        s.getrateofinterest);
    System.out.println("interest in axis bank = " +
                        a.getrateofinterest);
    System.out.println("interest in ICICI bank = " +
                        i.getrateofinterest);
  }
}
```

Output:
interest in SBI bank is 6
interest in axis bank is 7
interest in ICICI bank is 8

Scanned with CamScanner

## Final Variables & methods :- The final keyword can be applied at 3 places.

1. for declaring Variables
2. for declaring methods
3. for declaring Class.

A Variable can be declared as final then it cannot be modified further. the final Variable is always a constant.

Example: final int a=10;

in the above declaration a is fixed to 10 & can't change or modify the Variable a

Similarly the final keyword can be also be applied to the method. then method over riding is avoided. that means the methods those are declared with the keyword final cannot be overridden.

A class can also contain Keyword final. in this case inheritance is avoided i.e, we can't derive new class from existing old class.

Consider the following java programs which make use of the keyword final for declaring the method

Scanned with CamScanner

```
class test.
{ final void fun()
  {
    System.Out.println("In programming with
                        java");
  }
}
class test1 extends test.
  void fun()
  { System.Out.println("object oriented program
                        language");
  }
}
```

Output :- fun() in test1 Cannot fun() in test;
overridden method in final
final void fun.
^
1 error.

Final declaring the class :- if we declare particular class as final, no class can be derived from it. following Java program is Example for final class.

```
final class test
{
  void fun()
```

Scanned with CamScanner

```
    {
        System.Out.println("This function from is in
                                    base class");
    }
}
class test1 extends test
{
    void fun()
    {
        System.Out.println("This function is derived");
    }
}
```

Output :- Cannot inherit from final test
class test1 ^ extends test
1 error.

Finalize() method :- java has a facility of automatic garbage collection hence, even though we allocate the memory & forget to deallocate it then objects that are no longer used get freed & finalized method we will specifies more actions that must be performed before an object is destroyed.

"Garbage collection runs periodically checking for objects that are no longer referenced

by any running state or indirectly other referenced objects.

To add finalizer to a class simply define a method finalize() method.

Syntax: void finalize()
{
finalization code;
.....
}

Note:- finalize method is called just before garbage collection. but it is not called when an objects gets out of an scope.

Abstract methods & classes:- Abstraction is a process of hiding implementation details & showing only functionality to user. Another way it show only important things & hiding internal details.

example: Sending sms we just type the text & send the message we do not know the internal details of message delivery.

Abstract class in java:- A class that is declared as abstract is known as abstract class. it needs to be extended & its methods

implemented it cannot be instantiated.

Syntax: Abstract method

abstract void printabstr();

In the above syntax the method that is declared abstract & does not have implementation details is known as abstract method.

Syntax: Abstract class.

Below java pgm illustrate the abstract method & abstract class.

```
abstract class Bike  —— (1)
{
  abstract void run();
}
class Honda extends Bike
{
  void run()
  { System.out.println("Running Smoothly");
  }
  public static void main(String s[])
  { Honda b = new Honda();
    b.run();
  }
}
```

Scanned with CamScanner

Abstract class Bank — (2)

```
{ abstract int getrateofin();

}
class SBI extends Bank.
{ int getrateofin()
  { return = 7;
  }
}
class PNB extends Bank.
{ int getrateofin().
  { return = 8;
  }
}
class Kasbank extends Bank.
{ int getrateofin()
  { return = 9;
  }
}
class Bankhomeloans.
{
  public static void main (String ki[])
  {
    Bank B;
```

Scanned with CamScanner

```
b = new SBI();
System.out.println("Interest rate is: "
                   + b.getrate of in());

b = new PNB();
System.Out.println("Interest rate is: "
                   + b.getrate of in());

b = new Karbank();
System.Out.println("Interest rate is: "
                   + b.getrate of in());
}
}
```

Output :- Rate of interest is 7.
Rate of interest is 8.
Rate of interest is 9.

Methods with variable arguments (varargs) :- If we want to pass variable length argument to the method then varargs are used.
following is the syntax of using varargs

```
accessspecifier static void method_name (object
... arguments)
{
   body of method;
}
```

Scanned with CamScanner

The ellipsis (...) is used while representing the varargs type. With the help of 3 dots I can specify variable length arguments for a particular method.

Following simple Java program make use of varargs

```
public static class varargdemo.
{
  public static void main (String args[])
  {
    test("Adil", "Bansanti");
    test("charita", "Devanand", "elish");
  }
  public static void test(String... student)
  {
    for(String name : student)
    System.out.println("welcome" +name);
  }
}
```

Output :- welcome Adil
welcome Bansanti
welcome charita
welcome Devanad.
Welcome elish.

Scanned with CamScanner

Visibility control :- in the technic that makes access to data fields, methods, classes. the access modifiers used in java are of 3 types

* public.
* private
* default modifiers.

Public access modifier allows class, methods, data fields accessable from any class

private access modifier allows class, methods & data fields accessable only from within own class

if public & private not use they by default the class, methods & data fields are accessable by any class on the same package. This is called package private & package access.

A package is essentially grouping of classes

Example :- Illustrate visibility control.

```
package test;
public class class1
{  public int a;
   int b;
   privat int c;
   public void fun1()
   { System.Out.println("I am in fun 1");
   }
   void fun2()
   { System.Out.println("I am in fun 2");
   }
   privat void fun3()
   { System.Out.println("I am in fun 3");
   }
}
Public class class2
{
```

Scanned with CamScanner

```
public static void main (String a[])
{
    class1 obj = new class1 ();
    obj.a; // allowed
    obj.b; // allowed
    obj.c; // not allowed.
    obj.fun1 (); // allowed
    obj.fun2 (); // allowed
    obj.fun3 (); // not allowed.
}
}
```

```
package another_fol;
public class class3
{ public static void main (String s[])
  {
    Class 1 obj = new class1 ();
    obj.a; // allowed
    Obj.b; // not allowed.
    obj.c; // not allowed.
    obj.fun1 (); // allowed
    obj.fun2 (); // not allowed
    obj.fun3 (); // not allowed.
} }
```

Scanned with CamScanner

String & String buffer classes –

Strings : String is a [inbuilt class] in java [collection of characters,] String define the object we can make use of string to denote collection of characters.

Let us look at sample program in which string object is used inorder to handle some text.

String is also an inbuilt class in java present in language package. (.lang)

class Stdemo.

```
public static void main(String s[])
{
  char b[] = {'Y', 'E', 'L', 'L', 'O', 'W'};
  String C = "yellow book";
  String Color = new String [b];
  System.Out.println (Color);
  System.Out.println (C);
}
```

String methods : following are some commonly used methods by string class.

Scanned with CamScanner

| Method | Description. |
|---|---|
| 1. s.charAt(int position) | returns the character present at the position. |
| 2. s.indexOf('char') | it returns the first occurance of character passed in the string s |
| 3. s.length() | it gives the length of string s |
| 4 s1.equals(s2), | if s1 & s2 are both equal then it returns true. |
| 5. s1.equalIgnoreCase(s2) | By ignoring case, if s1 & s2 are equal then it returns true. |
| 6. s1.concat(s2). | it returns the concatenated string of s1 & s2 |
| 7. s.toUpperCase() | This method changes the content of s to upper case |
| 8. s.toLowerCase() | This method changes the content of s to lower case |
| 9. s.trim() | The trim method eliminates the unwanted white spaces. |
| 10. s1.compareTo(s2) | if s1 less than s2 -> it returns positive. if s2 > s2 -> it returns negative. |

Scanned with CamScanner

if $s_1 = s_2$ -> Then returns 0.

Below Java program illustrate use of string methods.

```
class stringmethodsno.
{
public static void main (String s[])
{ String s1 = "Black board";
  String s2 = "Dusters";
  String s3 = "Attendence";
  String s4 = "pen";
  String s5 = "chalk piece";
  System.out.println("The length of string s1 is "+s1.length());
  System.out.println("character passed at position 4 : "+s1.charAt(4));
  System.out.println("The occurance of character t in the
                      is "+s3.indexOf('t'));
  String s6 = "pen";
  System.out.println("The string s4 & s2 are equal :
                      "+s4.equals(s6));
  System.out.println("The string s4 & s6 are equal "+s4.equals
                      ignoreCase(s6));
  System.out.println("change of case of s5 "+s5.toUpperCase
                      Case());
  System.out.println("change of case of s4: "+s4.toLowerCase
                      Case());
```

Scanned with CamScanner

```
String s7 = " is not a car.";
System.out.println("After concatinat new string is:"
                    + s2.concat(s7));

if(s1.compareTo(s2));
System.out.println("s1 is big");
else
System.out.println("s2 is big");
String s8 = "  See the difference";
System.out.println("Before trim s8 is:" + s8);
System.out.println("After trim is:" + s8.trim());
}
}
```

StringBuffer class() –

The string buffer is a class which is alternative to the String class but StringBuffer is more flexiable to use than the String class. that means using StringBuffer we can insert some Components to the existing String or modify the existing String but, in case of String class once the String is defined it remains fixed.

String buffer is also called as a StringBuilder following are simple methods of StringBuffer class.

| Method Name. | Description. |
| --- | --- |
| 1. append (String str) | appends the string to the buffer. |
| 2. Capacity ( ) | it returns the capacity of string buffer. |
| 3. charAt (int index) | it returns a specific character from the sequence which specified by the index. |
| 4. delete (int start, int end) | it deletes the characters from the string specified by the starting & ending index. |
| 5. insert (int offset, ~~String s~~ char ch) | it inserts the character at the position specified by the offset. |
| 6. length ( ) | returns the length of StringBuffer. |
| 7. setCharAt (int index, char ch) | Character specified by index from StringBuffer is set to ch. |
| 8. setLength (int newlength) | ~~The character specified~~ it sets the length of String Buffer. |
| 9. toString ( ) | it converts the string representing data with StringBuffer. |

Scanned with CamScanner

10. replace(int start, int end, string str) — it replaces the characters specified by the new string

11. reverse() — The character sequence is reversed

with a java program that illustrate all the methods of string buffer class.

```
Class StringBufferDemo.
{
  public static void main (String s[])
  {
    StringBuffer sb = new StringBuffer("Two chairs");
    Sys.Out.pln("the length of sb is"+sb.length());
    Sys.Out.pln("The capacity of sb is"+sb.capacity());
    Sys.Out.pln("The character at position" +sb.charAt(7));
    sb.setCharAt('s','n')
    Sys.Out.pln("After setCharAt method" +sb);
    sb.setLength(3);
    Sys.Out.pln("After setLength sb is"+sb);
    Sys.Out.pln("inserting string & char in b/w sb" +
                sb.insert(3,"cusion");
    Sys.Out.pln("append method is used here".
                +sb.append("chairs");
```

Scanned with CamScanner

System.out.println("delete method used here" + sb.delete(0,10));

Sys.out.println("After reverse method" + sb.reverse());

}

}

Vectors - Vector is an inbuilt class in java that implements the dynamic array. this vector class stores any number of objects of any datatype.

Vector class is defined in java.util package.

in Vector class we can store the elements of any datatype that means in a single Vector we can store integer, string, double or any other datatype elements all together.

Vector can be created like this.

Vector myvector = new Vector(); // declaring Vector without size.

Vector myvector = new vector(5); // declaring with size 5

Difference b/w an array & Vector.

Scanned with CamScanner

| Arrays | Vectors. |
|---|---|
| 1. Array contains all the elements of similar datatypes | 1. Vector contains the elements that may be of varing datatypes. |
| 2. The size of array is fixed. | 2. The size in Vector is varing, we can change the size of vector when required. |
| 3. we can store the simple elements directly in the array | 3. we cannot store the simple elements directly into the database |
| 4. Arrays we can directly add the simple datatype element in the array | 4. we cannot directly add the simple datatype element in the vector. we need to add an object. |
| 5 | |

following are some of the most commonly used methods of Vector class.

| Methods | Description. |
|---|---|
| 1. void addElement(object) | for adding some elements in the vector this method is used. |
| 2. void insertElementAt(object obj, int pos) | for inserting elements in the vector specified by ith position |

Scanned with CamScanner

| | |
|---|---|
| 8. void removeElement(Object obj) | This method removes the specified element |
| 9. void removeAllElements(); | This method is for removing all elements from vector. |
| 10. void removeElementAt(int pos); | The element specified by its position gets deleted from vector. |
| 11. capacity(); | It returns the capacity of the vector. |
| 12. size(); | It returns the total no of elements present in vector |
| 13. void firstElement(); | It returns the first element of the vector. |
| 14. void lastElement(); | It returns the last element of the vector. |
| 15. void isEmpty(); | It returns true if it is empty otherwise false |
| 16. void indexOf(); | It returns the index of corresponding element in the vector. |
| 17. void setSize(); | This method is for setting the size of the vector. |

Below java program illustrate methods of vector class.

Scanned with CamScanner

```
import java.util.*;
class Vectordemo
{
public static void main (String s[])
{
Vector v = new vector ();
v.addElement("white");
v.addElement("Board");
v.addElement(4);
v.insertElementAt(1, "ceramic");
v.insertElementAt(0, "color");
Sys.Out.pln("size of vector in "+v.size());
Sy.Out.pln("capacity of vector in "+v.capacity());
Sy.Out.pln("first element in vector "+v.firstElement());
Sy.Out.pln("last element in vector "+v.lastElement());
V.removeElement(color);
V.removeElementA(0);
for(int i=0; i<=V.size(); i++)
Sys.Out.pln("element in the vector are "+v.element
(i));
v.removeAllElement();
for(i=0; i<=v.size(); i++)
Sys.Out.pln("elements in the vector are"
+v.elementAt(i));
}
}
```

Wrapper classes :- are those classes that allows primitive datatypes to be accessed as objects. the wrap. class is one kind of ~~class and~~ wrap. around primitive datatype

following table shows various primitive datatypes & the corresponding wrapper classes.

| primitive datatype | wrapper class. |
|---|---|
| 1. void. | Void |
| 2. byte | Byte |
| 3. boolean | Boolean. |
| 4. char | Character. |
| 5. int | Integer |
| 6. double | Double. |
| 7. long | Long. |
| 8. short | Short. |
| 9. float | Float. |

Methods to handle wrapper classes are enlisted in the following table.

| Methods used | Description. |
|---|---|
| 1. Integer val = new Integer (int val) | Any object is created for Integer variable & int value method is used allow the wrapper class Integer. |
| 2. Float val = new Float (float value) | Any object is created to float variable, float value is used to allow Float wrapper class |
| 3 Double val = new Double (double value) | An object is created for double variable, method double value used to allow the wrapper class double |
| 4. Characte val = new Character (char 'ch') | An object is created for character variable, 2 String method is used allow the character wrapper class. |

Below Program Illustrate Java class.

```
class wrapdemo
{ public static void main (String s[])
  Integer i = new Integer (24);
  Float f = new Float (24.1);
```

Scanned with CamScanner

```
Character c = new Character('A');
System.out.println("The value stored in i is" + i.intValue());
System.out.println("The value stored in f is" + f.floatValue());
System.out.println("The value stored in c is" + c.toString());
}
```

Output :- The value stored in i is 24
The value stored in f is 24.1
The value stored in c is A.

Enumerated types :- enum in java is a datatype that contains fixed set of constants it can be used for months of year

```
enum months {JAN, FEB, MAR, APR, MAY, JUN, JUL, AUG,
             SEP, OCT, NOV, DEC},

enum Season {SUMMER, SPRING, WINTER, RAINY},

enum directions {NORTH, SOUTH, WEST, EAST},

enum week {SUN, MON, TUE, WED, THU, FRI, SAT}.
```

The java enum constants are static & final implicitly.

Java enums can be thought of as class that have fixed set of constants.

The enumerated datatypes can be denoted by keyword enum, enum helps to define the user defined datatypes.

Scanned with CamScanner

The value can also be assigned to the elements in the enum datatypes.
Below program demonstrate the use of enum datatype

```
class enum_demo
{
  pu enum Days{SUN, MON, TUE, WED, THU, FRI, SAT};
    public static void main(String s[])
    {
      Days d = Days.MON;
      switch (d)
      {
        case SUN:
                System.out.pln("Sunday");
                break;
        case MON:
                System.out.pln("monday");
                break;
        Default:
                System.out.pln("otherday");
      }
    }
}
```

output:- Monday.

Scanned with CamScanner

Annotations :- java annotations is a tag that represents the meta data i.e, attached with class, interface, methods are fields to indicate additional information which can be used by java compiler & JVM

Following are the rules that must be followed while using annotation.

1. annotation declaration should start with the symbol @ (at the rate) following with the interface keyword, following with the annotation name.
2. Method declaration should not have any throws clauses
3. Method declaration should not have any parameters.
4. Return types of methods should be one of the following.

a) primitive, b) string, c) class, d) enums.
e) array of above types.

Example java program that illustrate how to use annotations in java.

```
@Override
Java Tiffin
void Something()
```

```
    {
        System.out.println("already finished my breakfast");
    }
    @Override
}
class todays extends tiffin
{
    void eatSomething()
    {
        System.out.println("I will have lathu");
    }
}
class annotdemo
{
    public static void main(String s[])
    {
        todays to = new todays();
        to.eatSomething()
    }
}
```

5/1/20 UNIT-03. 16 marks.

# Interfaces.

troduction:- Java does not support the
ncept of multiple inheritance. that means
or more classes cannot be derived from
single parent class. but there are large
mber of applications in which deriving of
ultiple classes from a single parent class
order to overcome this limitation of Java
terface concept has been introduced.

finition of Interface:- An Interface is defined
s datastructure ~~with~~ which contains data
mbers (variables) & member function without
ody (abstract methods).

## ifferences b/w classes & interfaces:-

| classes | Interfaces. |
|---|---|
| The class is denoted by using a keyword class. | 1. The interface is denoted by using a keyword interface. |
| The class contain data embers & methods, methods are defined in class mplementation. Thus class ntain executable code. | 2. The interface may contain data members & methods, but methods are not defined. Thus interface does not contain |

Scanned with CamScanner

| | |
|---|---|
| | executable code. |
| 3. By creating an instance of a class (object) the class members can be accessed. | 3. we cannot create an instance of interface |
| 4. The class can use various access specifier like public, private, default modifier | 4. The interface make use of only public access specifier |
| 5. The mebers of a class can be constant or final | 6. The members of interface are always declared as final. |

The interface can be defined using following syntax:-

```
access specifier interface interface name.
{
    return type method 1(param 1, ..... param n);
    return type static final variable name = constant value;
}
```

Example:-

```
public interface Book.
{ int price = 75;
  void write_note();
}
```

in the above example book is interface name which contains data member price & assigned

value 75. By default interface data members are final & a method with no body hence with node method is abstract method, the body for with note method is excepted to be defined within the class defination..

below java program illustrate implementation of interface.

```
public interface area
{
  float pi = 3.14f;
  public void compute(float x, float y);
}
class rectangle implements area.
{
pub void compute (float x, float y);
 {
   return x*y;
 }
}
class circle implements area
 public void compute (float x, float y)
 {
   return pi*x*x;
 }
}.
```

```
class Interdemo
{
public static void main (String s[])
{
    area a;
    rectangle rect = new rectangle();
    circle cir = new circle();
    a = rect;
    System.Out.println("area of rectangle "+a.compute
                                        (10,10));
    a = cir;
    System.out.println("area of circle "+a.compute
                                        (5,10));
}
}
```

Output:- Area of rectangle is 100
Area of circle is 78.5

(Accessing Interface)

Extending Interface:- Interfaces can be extended similar to the classes that means we can derive sub Interface from the Super Interface like how we derive child class from parent classes. To extend the class of Interfaces, the keyword extends is used.

Scanned with CamScanner

Syntax :-

interface nameofSubinterface extends nameof Super Interface

{

.....(body of interface)

}

Example :-

interface A

{ int val = 10;

}

interface B extends A

{

void print_val();

}.

That means in interface B, the print val method can access the value of variable Val.

Similarly more than 1 interface can be extended as shown below.

interface A

{

==

==

}

interface B

Scanned with CamScanner

{

}

Interface C extends A, B

{

}

Below Java program illustrate the extending Interfaces.

interface area

{

float pi = 3.14f;

double compute (float x, float y);

}

interface display extends area.

{ void display-result (double result);

}

class rectangle implements display

{

double compute (double x, double y)

{ return x * y;

}

double display-result (double result)

```
{ System.out.println("Area rectangle is " + result);
}
class InterfaceDemo.
{ public static void main(String s[])
  {
    Rectangle rect = new rectangle();
    double result;
    result = rect.compute(10.20);
    rect.display_result(result);
  }
}
```

output :-

Area of rectangle is 200

28/1/20. 23 marks

UNIT-04

# Packages

## Using System package :-

1) Java contains a collection of a package which in collective called as Java API
2) every package contains various class that are required for implementation
3) inside this class various methods are defined
4) while developing a java program. the java developer make use of this methods
5) Their are 2 ways of accessing the class stored in the core package

1st way - we import the java package class using the keyword import.

Supose, we want to use the Date class stored in util package. we have to write Below statement at the begining of the program.

package ↑ class ↑

import java.util.Date;

Scanned with CamScanner

2nd way - There are some situations in which we want to make use of several class stored in a package then we can with it as

import java.util.*;

here * means any class in corresponding package. thus, import statement are used at the beginning of the program. using the operator the class present inside the package can be refered..

Naming Conventions :-

for naming the packages some standard rules must be supported.

→ normally packages begin with lower case letters & class name begin with upper case letters. (only first character) for accessing the method belonging to a particular class of some package, the dot operator is used.

Example - double val = java.lang.Math.max(x, y);

class. method

→ The class math invokes the method max.

this method finds the Max variable of 2 variables x, y. This class present in package demo.

-> Every package name should be unique because duplicate package name may cause run time errors

-> According to naming convention of packages. it is suggested to make use of domain names. as prefix to predefined package names.

(Creating a package)
(Using a package)

To Create a package :- following are the steps which show how to creat a package in java.

Step 01 - Creat a folder named my-package & write the following code in notepad & Saved the file in this folder.

```
package my-package
public class A
{ int a;
    Void Set_val(int i)
    {
```

Scanned with CamScanner

```
    {
    void show_val().
    {   System.Out.println("the value of a "+a);
    }
}
```

Now the package created can be accessed in another class as shown below.

```
import java.my_package.A;
class ~~blockdemo~~ packdemo.
{ public static void main (String s[])
    {
        A a = new A();
        a.set_val(50);
        a.show_val();
    }
}
```

Save the above code by the name packdemo.java

Step 02 - Compile the above program as follows

f:\test > cd my_package

f:\test\my_package > javac A.java

Now make sure that the class file A.class is generated in the folder my_package

Step 03 - Now execute the program packagedemo.java as shown below

f:\test > cd my_package

f:\test\my_package > javac packdemo.java

f:\test\my_package > java packdemo

Value of a is 50.

Adding a new class to a existing package :-

We can add some class into the existing package My package for this purpose Some of the 3 steps to be followed

Step 01 - write a class (in a separate java file) which is to be added in the existing package.

Suppose we wish to add our new class

class B into the existing package my package, Create a B.java file & store the code in package as shown below.

```
package My-package ;
public class B
{
    public void show()
    {
        System.Out.println("class B is now
        added in this package");
    }
}
```

Step 02 - The test program. packdemo.java can be edited as follows.

```
import java.My-Package.*;
class packdemo
{
    public static void main (String s[])
    {
        A a = new A();
        B b = new B();
        a.seta (50);
        b.show();
        a.show_a();
    }
```

Scanned with CamScanner

Step 3 – Issue the following command to execute the package with newly added class B

D:\test\pgm> cd my-package ↵

D:\test\pgm\my-package> javac A.java

D:\test\pgm\my-package> javac B.java

D:\test\pgm\my-package> javac packdemo.java

D:\test\pgm\my-package> java packdemo

Output:-

Class B is now added in the package

value of a = 50.

Hiding the classes:- A class can be hidden by declaring as non public if we try to access a method of a hidden class then error message is displayed by the compiler. following are the steps that shows how to hide a

particular class

step 01 — Create a folder & name it as Sports & write a following java program in this folder.

```
package sports;
public class Team_events
{
  void events_boys()
  {
    System.out.println("volleyball, cricket");
  }
  void events_Girls()
  {
    System.out.println("Throw ball");
  }
}
```

Now adding a new ~~class~~ non public class to the existing package, sports.

```
package sport;
class Athletics
{
  void when_n_where()
```

Scanned with CamScanner

```
{
    System.out.println("7th feb @ St. John's Ground");
}
}
```

Step 02 — write another java pgm in which you will access the method of above declared class. ~~This java pgm should not be~~

```
import java.sports.*;
class hideclassdemo
{
    public static void main(String s[])
    {
        Team_event te = new Team_event();
        Athletics a = new Athletics();
        te.Event_boys();
        te.event_girls();
        a.when_n_where();
    }
}
```

Scanned with CamScanner

because in the pgm hideclassdemo.java we are accessing non public class athletic i.e. hidden class cannot be accessed outside the package.

D:\sposh> javac Athletic.java.

D:\sposh > javac teamevents.java.

D:\sposh > javac hideclassdemo.java.

hideclassdemo.java :7 sposh.Athletic is not public in sposh; cannot be accessed

```
Athletic at= new Athletic();
^
```

hideclassdemo.java :7 sposh Athletic is not public in sposh. cannot be accessed

```
Athletic at= new athletic();
                ^
```

2 errors.

Static import :- is a new feature added in Java. in order to allow the static member class it is necessary to qualify references with the class it came from that means in order to allow the static member of a class, it is a must to write the member along with its belonging class name. for example

```
class StaticImportDemo
{
    public static void main (String s[])
    {
        double a = 25;
        double val = Math.sqrt (a);
        System.out.println ("The Square root of
                            "+a+ "is" + val);
    }
}
```

Output :-

The Square root of 25.0 is 5.0

Scanned with CamScanner

In order to avoid unnecessary use of static class members like Math & System. we should use static import. The above code can be changed by using static import. Here is the difference b/w 2 pgm.

```
import static java.lang.System.out;
import static java.lang.~~System~~Math.Sqrt;
class staticimportdemo
{
    public static void main (String s[])
    {
        double a=25;
        double val = Sqrt(a);
        Out.println("The Square root of a
                    "+a"."is"+val);
    }
}.
```

Output :-

The Square root of 25.0 is 5.0

10.02.2017

## Unit-IV: Multithreaded programming Max.Marks: 23

Introduction, Creating threads, extending the thread class, stopping and blocking a thread, Life cycle of a thread, Using Thread methods, Thread exceptions, Thread priority, Synchronization, implementing Runnable interface, Inter-thread communication.

**Introduction:** Multithreading is the concept in Java, therefore in Java we can write the programs that perform multitasking. It allows the user to handle multiple tasks together.

**Definition:** "Thread is a tiny program running continuously." it is sometimes called as light-weight process.

Differences between thread and process

| Thread | Process |
|---|---|
| 1. Thread is a light-weight process. | 1. Process is a heavy weight process. |
| 2. Threads do not require separate address space for its execution. It runs in the address space of the process to which it belongs to. | 2. Each process requires separate address space to execute. |

Differences between Multithreading and Multitasking

| Multithreading | Multitasking. |
|---|---|
| 1. Thread is a fundamental unit of multithreading | 1. program or process is a fundamental unit of multitasking environment. |
| 2. Multiple parts of a single program gets executed in multithreading environment | 2. Multiple programs gets executed in multitasking environment. |

Scanned with CamScanner

| | |
|---|---|
| 3. During multithreading the processor switches between multiple threads of the program. | 3. During multitasking the processor switches between multiple programs or processes. |
| 4. It is cost effective because cpu can be shared among multiple threads at a time | 4. It is expensive because a particular process uses cpu other processes has to wait. |
| 5. It is highly efficient | 5. It is less efficient in comparison with multithreading. |
| 6. It helps in developing efficient application programs. | 6. It helps in developing efficient operating system programs. |

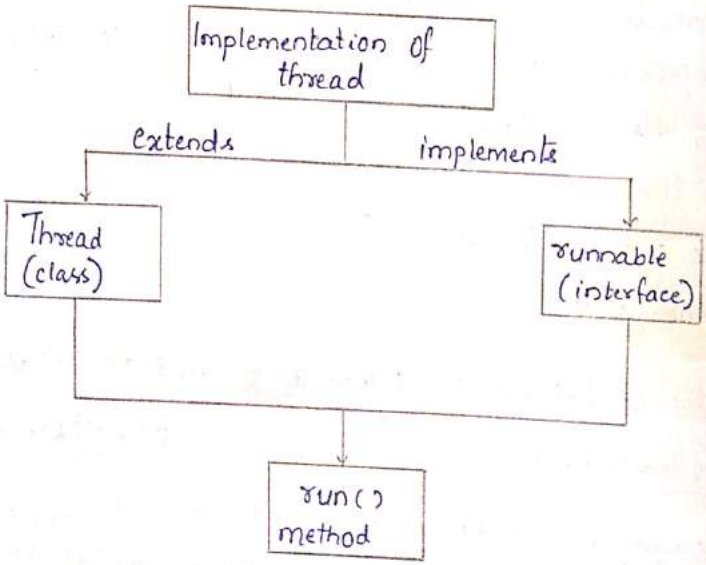
## Creating threads :

The threads can be in various states such as new, runnable, waiting, blocked & terminated state

In Java we can create thread programs using two approaches -

1. using Thread class
2. using runnable interface.

below figure shows the thread implementation using 2 approaches.




As shown in the above diagram, there are 2 ways to create thread programs in java

1. by extending Thread class & the other is by
2. by implementing runnable interface.

Scanned with CamScanner

The run() method is the most important method in both the ways. The run() method can be written as follows.

```
Public void run()
{
    // statements for implementing Thread
}
```

for invoking the Thread's run method the object of a Thread class is required. This object can be obtained by creating and initiating a thread using the start() method.

## 1. Extending the Thread class:

The procedure to extend a thread class is as given below.

Step 1: The main class must extend the Thread class.

Step 2: A thread is created in the constructor using the start() method. this method initiates the execution of Thread.

Step 3: Override the method run() by writing the code required to execute a thread.

The below java program illustrates creating a single Thread by extending the Thread class.

```
Class Threaddemo extends Thread → Extend the inbuilt Thread class to your class
{
    String str = " ";
    Threaddemo (String s)      → In the constructor the Thread is created
    {
        str = s ;
        start () ;             → start the thread execution.
    }
    public void run()          → In the run method the code for Thread execution is written.
    {
        System.out.println (str);
    }
}
public class Thread_Create
{
    public static void main (String a[])
    {
        Threaddemo t = new Threaddemo ("One Thread Created");
    }
}
```

Scanned with CamScanner

Now save & run the program as shown below

javac Thread_Create.java

java Thread_Create

One Thread Created ---> is the output

## Stopping and Blocking The Thread.

**Stopping a Thread:** A Thread can be stopped from running further by issuing the following statement

Thread object.stop(); i.e th.stop();

By this statement the Thread enters into dead state. from stopping state a Thread can never return to a runnable state.

**Blocking a Thread:** A Thread can be temporarily stopped from running state. This is called blocking or suspending of a Thread.

following are the ways by which Thread can be blocked

1. sleep() - By sleep method a thread can be blocked for some specific time. When the specified time gets elapsed Then the Thread can return to a runnable state.

2. suspend() - By suspend method the Thread can be blocked until further request comes. When the resume() method is invoked Then the Thread returns to the runnable state.

3. wait() - The Thread can be made suspended for some specific conditions. When the notify method is called then the blocked Thread returns to the runnable state.

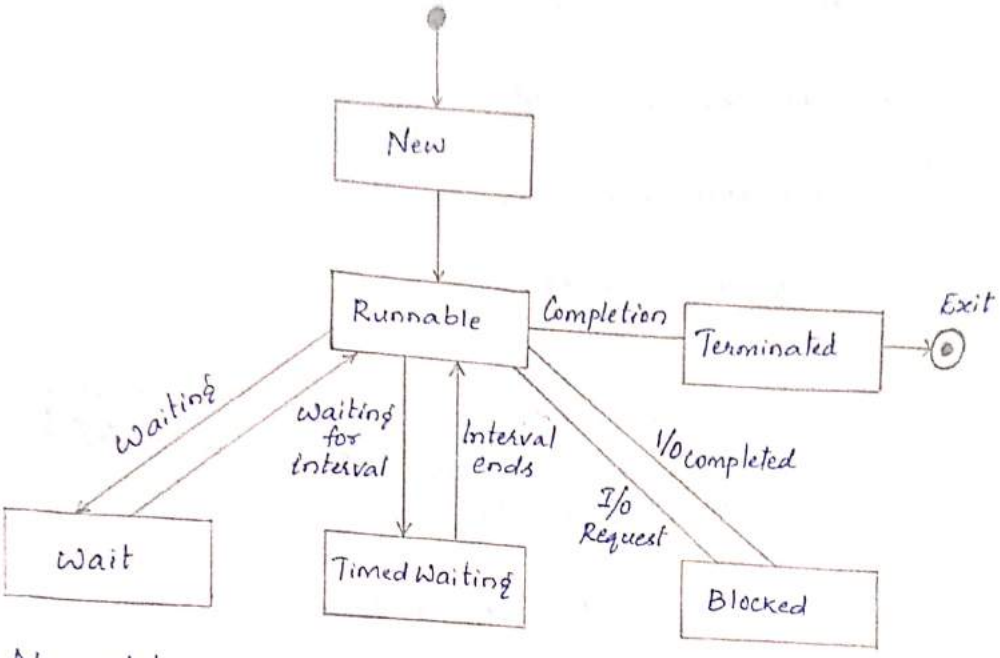
The difference between the suspending & stopping thread is that if a thread is suspended Then its execution is stopped or blocked temporarily and it can return to a runnable state. But in case, if a Thread is stopped then it goes to a dead state and can never return to runnable state.

Scanned with CamScanner

## Life cycle of a Thread :

Thread life cycle specifies how a thread gets processed in the Java program. By executing various methods.

Following figure represents how a particular thread can be in one of the state at any time.




**New state :**

When a thread starts its lifecycle it enters in to the new state or a create state.

**Runnable state :**

This is a state in which a thread starts executing

**Waiting state :**

Sometimes one thread has to undergo in waiting state because another Thread starts executing.

**Timed waiting state :**

There is a provision to keep a particular threading waiting for some interval. This allows to execute high prioritized Thread first. After the timing gets over, the Thread in waiting state enters in runnable state.

**Blocked state :**

When a particular thread issues an input/output request then Operating system sends it in blocked state until the I/O operation gets completed. After the I/O completion the Thread is sent back to the runnable state.

**Terminated state :**

After successful completion of the execution of the Thread in

Scanned with CamScanner

## Using Thread methods:

In this Thread class, some methods are used here. They are yield(), sleep(), stop(), wait(), resume(), notify().

The below Java program example which illustrates the use of yield(), stop() & sleep() methods of Thread class.

```
Class One extends Thread
{
    Public void run()
    {
        for(int i=1; i<5; i++)
        {
            if(i==1)
            yield(); // temporarily the execution is suspended and
                     transfers control to another Thread.
            System.out.println("class one: i= "+i);
        }
        System.out.println("Exiting one");
    }
}
Class Two extends Thread
{
    public void run()
    {
        for(int j=1; j<5; j++)
        {
            if (j==3)
            stop(); // stopping or killing the Thread completely.
            System.out.println("class Two: j=" +j);
        }
        System.out.println("Exiting Two");
    }
}
```

Scanned with CamScanner

```
Class Three extends Thread
{
    Public Void run()
    {
        for(int k=1; k<5; k++)
        {
            if(k==2)
            {
                sleep(1000);
            }
            System.out.println("Class Three : k = "+k);
        }
        System.out.println("Exiting Three");
    }
}
Class Threadmain
{
    public static void main(String a[])
    {
        One t1 = new One();
        Two t2 = new Two();
        Three t3 = new Three();
        System.out.println("Starting Thread One");
        t1.start();
        System.out.println("Starting Thread Two");
        t2.start();
        System.out.println("Starting Thread Three");
        t3.start();
        System.out.println("End of demo");
    }
}
```

Output:

```
Starting Thread One
Starting Thread Two
class One : i=1
class Two : j=1
class Two : j=2
Starting Thread Three
End of demo
Class one: i=2
Class one: i=3
Class one: i=4
Class Three: k=1
Exitting One
Class Three : k=2
Class Three : k=3
Class Three : k=4
Exitting Three
```



Scanned with CamScanner

## Thread Exceptions:

Any error causing situation in Java is handled by exception handling mechanism.

for error prone block of statements Java throws the exception. Such a block is written within try-catch block.

for instance the sleep() statement should be written within the try-catch block.

For example:

```
try
{
    // block of statements to be executed
}
catch (InterruptedException e)
{
   // when exception occurs it is handled here
}
catch (ThreadDeath e)
{
   // When exception occurs it is handled here
}
catch (IllegalArgument e)
{
   // When exception occurs it is handled here
}
```

## Thread priority:

In Java threads scheduler selects the threads using their priorities. The thread priority is a simple integer value that can be assigned to the particular thread. These priorities can range from 1 (lowest priority) to 10 (highest priority).

There are two commonly used functionalities in thread scheduling

- setPriority()
- getPriority()

Scanned with CamScanner

The function setPriority is used to set the priority to each thread.

Thread_Name.setPriority(Priority_val);

The priority_val is a constant value denoting the priority for the thread. It is defined as follows

MAX_PRIORITY = 10
MIN_PRIORITY = 1
NORM_PRIORITY = 5

The function getPriority is used to get the priority of the thread.

Thread_Name.getPriority();

## Synchronization:

When two or more threads needs to access shared memory, then there is some way to ensure that the access to the resource will be by only one thread at a time. The process of ensuring one access at a time by one thread is called synchronization.

The synchronization is the concept which is based on monitor. Monitor is used as mutually exclusive lock or mutex. When a thread owns this monitor at a time then the other Threads cannot access the resources. Other threads have to be their in waiting state.

In Java every object has implicit monitor associated with it. for entering in Object's monitor, the method is associated with a keyword synchronized. when a particular method is in synchronized state then all other threads have to be there in waiting state.

There are two ways to achieve the synchronization.

1. Using Synchronized Methods
2. Using Synchronized blocks (statements).

let us make the method synchronized to achieve the synchronization by using following Java program.

Scanned with CamScanner

```
Class Table
{
    Synchronized void printTable(int n)
    {                                          //Synchronization method.
        for(int i=1 ; i<=5 ; i++)
        {
            System.out.println(n*i);
            try
            {
                Thread.sleep(400);
            }
            catch(Exception e)
            {
                System.out.println(e);
            }
        }
    }
}
Public class TestSynchronization
{
    Public static void main(String a[])
    {
        final Table obj = new Table(); //Only one object
        Thread t1 = new Thread()
        {
            Public void run()
            {
                Obj.printTable(5);
            }
        };
        Thread t2 = new Thread()
        {
            public void run()
            {
                obj.printTable(100);
            }
        ,
```

Scanned with CamScanner

```
class mythread implements Runnable
{
  Public void run()
  {
    sopln("Creating Thread using Runnable
          - interface");
    for(int i=1; i<=10; i++)
    { sopln("RJS polytechnic "+i);
    }
    sopln("end of Thread");
  }
}
class pgm
{ psvm(String args[])
  { mythread x = new mythread();
    Thread th = new Thread(x);
    th.start();
  }
}
```

```
    t1.start();
    t2.start();
  }
}
```

Output:

```
5
10
15
20
25
100
200
300
400
500
```

## Implementing Runnable interface:

Implementing Thread program Using Runnable interface is Preferable than implementing it by extending the Thread Class because of the following 2 reasons.

1. If a class extends a Thread class then it cannot extends any other class which may be required to extend.
2. If a class Thread is extended then all its functionalities gets inherited. This is an expensive operation.

Following Java program shows how to implement Runnable interface for creating a single thread.

```
Class Threaddemo implements Runnable
{
    Thread t;
    String str = " ";
    Threaddemo(String s)
    {
        t = new Thread(this); // assign value to object of Thread
        str = s;
        t.start();  //start the Thread
    }
```

Scanned with CamScanner

```
    Public void run() // execution of Thread
    {
        System.out.println(str);
    }
}
Public class Threadusingrunnable
{
    Public static void main(String a[])
    {
        Threaddemo t = new Threaddemo("One Thread Created");
    }
}
```

// Invoking Constructor

## Inter-thread communication:

Two or more threads can communicate with each other by exchanging the messages. This mechanism is called the inter-Thread communication and those are

notify() - If a particular thread is in the sleep mode then that thread can be resumed using the notify call (method).

notifyall() - This method resumes all the Threads that are in suspended state.

wait() - The calling Thread can be sent into a sleep mode.

Scanned with CamScanner

12.03.2017

# Unit VI Managing Errors and Exceptions

Max Marks: 23
Min " : 15.38

Contents:

Introduction, Types of Errors, Exceptions, Syntax of Exception handling code, Multiple Catch statements, Using Finally statement, Throwing our own Exception.

**Introduction:** Mistakes in the program lead to errors or Unexpected output. Errors are basically the bugs in the program due to which program go wrong. Sometimes due to the errors in the program, the program may stop abruptly or may cause the entire system to crash. To avoid such situations it is very important to handle the errors properly or to manage the possible error conditions in such a manner that the program will not cause abrupt termination of the program or crashing of entire system.

## Types of Errors:

There are 2 Types of errors
1. Compile time error
2. Run time error.

**1. compile time error:** The errors that are detected by the Java compiler during the compile time are called the compile time errors. When compiler issues the errors it will not generate the .class file. Hence we must eliminate all the compile time errors first.

The most commonly occuring compile time errors are
1. Missing Semicolons.
2. Wrong spelling of Keywords and identifiers.
3. Missing brackets of classes and methods.
4. Missing double quotes in the string.

Scanned with CamScanner

5. Use of undeclared variables
6. Use of = instead of ==
7. Incompatible types in assignment statement.
8. illegal reference to the Object.

Example :

```
Class Compileerrordemo
{
    public static void main (String a[])
    {
        System.out.println("Hello Java programmers");
    }
}
```

Output :

```
Compileerrordemo.java:5: Unclosed string literal
System.out.println("Hello Java programmers);
                   ^
Compileerrordemo.java:6: ')' expected
}
^
2 errors.
```

Runtime errors :

Sometimes the program is free from the Compile time errors and Creates a .class file. But it does not yield the results as per our expectations. In such situation we declare that the program has run time errors.
The runtime errors are basically the logical errors, That get Caused due to wrong logic.

The most commonly occuring runtime errors are -
1. Accessing the array element which is out of the index.
2. In an expression, divide by zero.
3. Trying to store a value into an array of incompatible type
4. passing the parameters that is not in the valid range.
5. Converting invalid string to numbers.

Scanned with CamScanner

6. Trying to illegally change the state of thread
7. Trying to access the character which is out of bounds of a string.

Example:

```
Class   Runerrordemo
{
   Public static Void main(String args)
   {
       int a,b,c;
       a=10;
       b=0;
       c=a/b;
   }
}
```

Output:

Exception in thread "main" java.lang.ArithmeticException: / by zero at
Runerrordemo.main(Runerrordemo.java:8)

17/3/17

## Exceptions:

Exception is an unusual condition that can occur in the program. This condition is caused due to run time error in the program.

In Java, whenever the exception occurs then it creates an exception object and throws it.

If the exception object is not caught and handled properly then the interpreter will display an error message, and ultimately the program will terminate. Even though the exception occur and if we want to continue the execution of remaining code then it is necessary to handle the error condition properly by displaying the appropriate messages. This mechanism is called the exception handling mechanism/Technique.

Scanned with CamScanner

Following are the tasks that need to be carried out for handling the error code.

1. Find the problem - This is called hit the exception
2. Inform that the exception has occured - By throwing the exception.
3. Receiving and displaying the information about the error - This is called catching of exception.
4. Taking Corrective action. That means handling of exception.

Various common exception types and causes are enlisted in the following table.

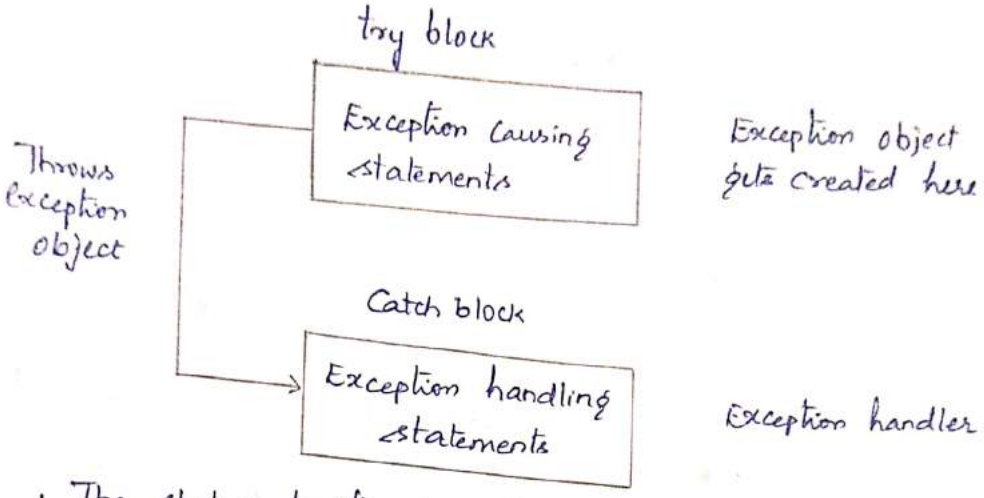
| Exception | Description |
|---|---|
| 1. ArithmeticException | This is caused by error in Math operation. Example : Divide by zero. |
| 2. NullPointerException | Caused when an attempt to access an object with a Null reference is made. |
| 3. IOException | When an illegal input/output operation is performed Then this exception is raised. |
| 4. IndexOutOfBoundsException | An index when gets out of bound, This exception will be caused. |
| 5. ArrayIndexOutOfBoundsException | Array index when gets out of bound, This exception will be caused. |
| 6. ArrayStoreException | When a wrong object is stored in an array This exception occurs. |
| 7. EmptyStackException | An attempt to pop. The element from empty stack is made then This exception occurs. |
| 8. NumberFormatException | When we try to convert an invalid string to number this exception gets caused. |

Scanned with CamScanner

9. RuntimeException : To show general runtime Error this Exception must be raised.

10. Exception : This is most general type of exception.

## Syntax of Exception handling Code:




- The statements that are likely to cause an exception are enclosed within a try block. for these statements The Exception is thrown.
- There is another block defined by the keyword catch which is responsible for handling the exception thrown by the try block.
- As soon as exception occurs it is handled by the catch block.
- The catch block is added immediately after the try block.

Following is an example of try-catch block.

```
try
{
    // exception gets generated here
}
Catch (type_of_Exception e)  → e is the Exception object
{
  // exception is handled here
}
```

Scanned with CamScanner

- If any one statement in the try block generates exception then remaining statements are skipped and the control is then transferred to the catch statement.

```
class RunErrDemo
{
    public static void main(String a[])
    {
        int a, b, c;
        a = 10;
        b = 0;
        try
        {
            c = a/b;
        }
        Catch(ArithmeticException e)
        {
            System.out.println("In Divide by zero");
        }
        System.out.println("In The value of a:" +a);
        System.out.println("In The value of b:" +b);
    }
}
```

Output:

```
Divide by Zero
The Value of a : 10
The Value of b : 0.
```

Note that even if the exception occurs at some ~~time~~ point, the program doesnot stop at that point.

Program for IndexOutOfBoundsException:

When we use an index which is beyond the range of index then IndexOutOfBoundsException occurs.

Following Java program illustrates it.

Scanned with CamScanner

```
Class Excepprog
{
 Static Void fun(int a[])
 {
   int c;
   c= a[0]/a[2];
 }
 Public static void main(String b[])
 {
     int  a[] = {10, 5};
     fun(a);
  }
}
```

After we compile & interpret the below message is displayed

Exception in thread "main" Java.lang.ArrayIndexOutOfBoundsException:2

Therefore the catch block is added immediately after the try block.

Try catch block for ArrayIndexOutOfBoundsException.

Now in order to handle this exception we can write the following modified code.

```
Class Excepprog
{
  static void fun(int a[]) throws ArrayIndexOutOfBoundsException
  {
     int c;
     try
     {
        c= a[0]/a[2];
     }
     Catch(ArrayIndexOutOfBoundsException e)
     {
       System.out.println("Caught exception:" +e);
     }
  }
  public static Void main(String c[])
  {
      int a[] = {10, 5};
      fun(a);
   }
```

Scanned with CamScanner

Then we will get the following output.

Caught exception: java.lang.ArrayIndexOutOfBoundsException

## Multiple Catch statements:

It is not possible for the try block to throw a single exception always.

There may be the situations in which different exceptions get raised by a single try block statements and depending upon the type of exception thrown it must be caught.

To handle such situation multiple catch blocks may be for the single try block statements.

The syntax for single try and multiple catch is

```
try
{
  ... // exception occurs here
}
catch(Exception_type1 e)
{
  // exception type 1 is handled here
}
catch(Exception_type2 e)
{
  // exception type 2 is handled here
}
catch(Exception_type3 e)
{
  // exception type 3 is handled here
}
```

Program example →

Scanned with CamScanner

```
Public class Multicatchdemo
{
  Public static void main(String c[])
  {
      try
      {
          int a[] = new int[5];
          a[5] = 30/0;
      }
      Catch(ArithmeticException e)
      {
        System.out.println("Task1 is completed");
      }
      catch(ArrayIndexOutOfBoundsException e)
      {
        System.out.println("Task 2 completed");
      }
      Catch(Exception e)
      {
        System.out.println("Common Task completed");
      }
      System.out.println("Rest of The code");
  }
}
```

Output: task1 completed
Rest of The code.

## Using Finally statement:

Sometimes because of execution of try block the execution gets break off. And due to this some important code (which comes after throwing off an exception) may not get executed.
That means, Sometimes try block may bring some Unwanted things to happen.

The finally block provides the assurance of execution of some important code that must be executed after the try block.

Scanned with CamScanner

Even though there is any exception in the try block the statements assured by finally block are sure to execute. These statements are sometimes called as clean up code.

The syntax of finally block is

```
finally
{
    //clean up code that has to be executed finally
}
```

The finally block always executes. The finally block is to free the resources.

Below java program which shows the use of finally block for handling exception.

```
Class finallydemo
{
    public static void main(String d[])
    {
        int a=10, b=-1 ;
        try
        {
            b = a/0;
        }
        Catch (ArithmeticException e)
        {
            System.out.println("In catch block: " +e);
        }
        finally
        {
            if(b!=-1)
                System.out.println("finally block executes without
                        occurrence of exception");
            else
            System.out.println("finally block executes on
                        occurrence of exception");
        }
    }
}
```

Scanned with CamScanner

Output:

In catch block: java.lang.ArithmeticException: / by zero.
finally block executes on occurrence of exception.

In the above program, on the occurrence of exception in try block the control goes to catch block, the exception of instance ArithmeticException gets caught. This is divide by zero exception and therefore / by zero will be printed as output. following are the rules for using try, catch and finally block.

1. There should be some preceeding try block for catch or finally block. Only catch block or only finally block without preceding try block is not at all possible.
2. There can be zero or more catch blocks for each try block but there must be single finally block present at the end.

19.03.2017

## Throwing out own Exception:

We can throw our own exceptions using the keyword throw. The syntax for throwing out own exception is

Throw new Throwable's subclass

Here the Throwable's subclass is actually a subclass derived from the exception class.

for example:

Throw new ArithmeticException();
↖ Throwables subclass.

below java program which illustrates this concept. →

Scanned with CamScanner

```
import java.lang.Exception;
class MyOwnException extends Exception
{
    MyOwnException(String msg)
    {
        super(msg);
    }
}

class MyExceptDemo
{
    public static void main(String args[])
    {
        int age;
        age=15;
        try
        {
            if(age<21)
            throw new MyOwnException("Your age is less than
                actual age to vote");
        }
        catch(MyOwnException e)
        {
            System.out.println("This is my exception block");
            System.out.println(e.getMessage());
        }
        finally
        {
            System.out.println("Finally block : End of The Pgm
        }

    }
}
```

Scanned with CamScanner

In this code, The age value is 15 and in the try block. The if condition throws the exception if the value is less than 18. As soon as the exception is thrown the catch block gets executed. Hence as an output we get the first message "This is My Exception block".

Then the control is transferred to the class MyOwnException defined at the top of the program. The message is set and it is "your age is less than actual age to vote". This message can then printed by the catch block using the System.out.println statement by means of e.getMessage(); here e is the exception object and getMessage is the inbuilt method of Exception class that displays the thrown message in the catch block.

At the end the finally block gets executed.

Scanned with CamScanner