Chapter- 1
Introduction to Java

1) What is Java? Explain the features of java. [2012, 2013, 2015, 2016]
Ans:- Java is pure object-oriented & high level programming language developed by 'Sun-Microsystems' USA in 1991. Java is high level programming language, which consists of objects & classes.

Features of Java
i) Java is Simple:- Java was developed by taking the best points from other programming languages, primarily C and C+ +.
- Error tasks such as pointers and memory management have either been eliminated or are handled by the Java environment automatically rather than by the programmer.
- Since Java is primarily a derivative of C + +.

ii) Java is Object-Oriented: - It is pure object oriented language. Everything in java is object. All program code & data resides within a objects & classes.
- Java comes with an extensive set of classes arranged in packages.
- In object-oriented programming (OOP), data is treated as objects to which methods are applied. Java's basic execution unit is the class.

iii) Java is Distributed: - It is designed as distributed languages for creating application on network.
- It has ability to share both data & program.
- Commonly used Internet protocols such as HTTP and FTP as well as calls for network access are built into Java.

iv) Java is Platform independent: It means we can run write & compile the java code in one platform (eg: windows) and can execute in any other supported platform.
(eg: Linux, sun solaries etc.)

v) Java is Interpreted: - When Java code is compiled, the compiler outputs the Java Byte code which is an executable for the Java Virtual Machine.
- The Java Virtual Machine does not exist physically but is the specification for a hypothetical processor that can run Java code.
- Java interpreter generates machines code that can be directly executed by the machine that is running Java program.

vi) Java is Robust and Secure:- Java is robust language. It provides many safeguards to ensure reliable code.
- It has strict compile-time & run-time checking for data types.
- Java manages memory automatically by using an automatic garbage collector.
- The garbage collector runs as a low priority thread in the background keeping track of all objects and references to those objects in a Java program.

Security: Java systems not verify all memory access but also ensure that no viruses are communicated through an applet.
- The absence of **pointers** in java ensures that programmer cannot give access to memory locations without proper authorization.
- Java compiler produce only correct java code, there is still possibility of the code being tampered between compilation & runtime.

vii) Java is Architecturally Neutral: - The Java compiler compiles source code to a stage which is intermediate between source and native machine code.
- This intermediate stage is known as the byte code, which is known as neutral.
- The byte-code conforms to the specification of hypothetical machine called ‘java virtual machine.

viii) Java is Portable: - Java ensures portability in two ways. Java compiler generator byte codes instructions that can be implemented on any machine.
- By porting an interpreter for the Java Virtual Machine to any computer hardware/ operating system, one is assured that all code compiled for it will run on that system. This forms the basis for Java's portability.

ix) Multithreaded and interactive: - Java supports developing programmes that handle multiple tasks simultaneously. This feature improves the interactive performance of graphical system.
- Java run-time comes with tools that support multiprocessor synchronization and constructs smoothly running interactive system.

2) Differentiate between Java & C++. [2012, 2014]
Ans: -

Java	C & C ++
1) Java doesn’t support pre-processors like #define, #include etc..	1) C & C++ supports all pre-processors like #define, #include,, etc..
2) Java doesn’t support Method Overriding.	2) C & C++ support Method Overriding.
3) Java is Pure Object- oriented languages.	3) C & C++ are Procedure-oriented languages.
4) Templates are not supported by java and in java there is no global variable concept.	4) Templates are not supported by C and C++.
5) Java uses Applets and servlets for internet services.	5) C & C++ do not provide applets and servlets services.

3) Explain Java Development Kit (JDK) in brief.
Ans: -JAVA DEVELOPMENT KIT (JDK):
- JDK comes with a collection of tools that are used for developing and running java programs.

They include:-
1) javac:- The javac compiler is used to compile the java source code files into byte codes.
2) Java interpreter: - The java interpreter is used to execute java byte codes. It takes as an argument the, name of the class file to execute or the name java archive file (jar).
3) appletviewer:- applet viewer is used for viewing java applets. It enables us to test java applets without using web browsers.
4) java (java disassembles):- java converts byte code file into program description.
5) javah:- It produces header files for use with native network.
6) javadoc:- Creates html documentation from java source file.

4) Explain general program structure of Java. [2012, 2013, 2015, 2016]
Ans:-
```
 // comments sample
         import.java.io.*;
        class sample
        {
        public static void main (String args[])
        {
       System.out.println("Hello world");
       }
}
```

a) Comments in java:- This are not executable statements, which returns for programmers reference but not for program reference.
In java we can have two types,
1) // it is used for single line commenting
Ex: // this line commented.
2) /* */ used for multiline commenting.
Ex:- /* java is difficult */

b) Import statements: - java doesn't support header files like C. Java provides another way of accessing the recourses which are stored in separate files in different packages by importing the packages we can refer the classes, methods in our program.
- We can write as many as import statements.
Ex :- import ABCD.*; //means import all classes from package ABCD import ABCD. tjava.*; //means import only java class of package EEPB.

5) What is Class? Explain with example.
Ans:- Class is a user-defined data type or blueprint using which we can create objects. In java, without class, we can write the programs.
Ex:
```
 import.java.io.*;
 class sample
{
  public static void main (String args[])
 {
System.out.println("Hello world");
```

```
        }
    }
```

public static void main(String args[ ]):- Java uses bit different type of the main method
→**public :-** public is an access specifier if we declare any method as public it is globally accessible.
→**Static:-** Java is pure object oriented programming if we want to use any method compulsory we have to create.
→**void :-** It is a keyword returns nothing.
→**Main:-** Here all programs execution starts from main only.
→**String args [] :-** It is command line argument list sent t the method main at run time.

6) What is Token? Explain its types. [2012, 2015]
Ans:- The smallest individual unit of program are called as '**Token'.**
Java includes

5types of Tokens:-

a) Keywords b) Identifiers c) Literals d) Operators e) Separators

a) Keywords:- Keywords are predefined or reserved words have a specific feature that is defined in the programming language.
- Keywords, operators, separators combined together forms a java program.
- The Java language has a rich set of keywords i.e. 53.

- **Following is the list of keywords present in java**:

Keywords	abstract	assert
Boolean	break	byte
Case	catch	char
Class	const	continue
default	do	double
float	for	goto
if	implements	import
package	private	protected
true	try	void

b) Identifiers:- Are names to refer the program objects in other words identifiers are user defined tokens.
- They are used in naming the classes, variables, methods, objects, packages, interfaces in the program.
- While choosing identifies the following rules to be maintained.
a) Identifiers may be alphabetic or alphanumeric but not numeric.
b) Identifier should not start from the number.
c) They should not be keywords.

Ex: Invalid Identifiers: la, avg#, try, 123, roll-no Valid Identifiers: al, Avg, try, ABC, roll_name.

c) Literals:- The compiler needs to translate the character strings "3.1" and "43" into numerical values of a particular type.
- Such explicit values in a computer language are called "liberals" for the obvious reason that they are literally equivalent to their stated value.
- Java supports 4 types of Literals:-
a) Floating Point Literals b) Integer Literals c) Characters and String Literals
d) Special Literals.

d) Operators: - An operator, basically is a symbol that takes the arguments as produces the result.

e) Separators: - Separators are symbols which divide our program into groups even they help in arranging the code. Separators in Java
; Semi-colon indicates the end of a statement.
→( ) Parentheses used in several places including:-
→{} Curly braces: - enclose the fields and methods of a class.
→ [] Array declaration:- array specification.

7) What is JVM? Explain it. [2013, 2014, 2015]
Ans: JAVA VIRTUAL MACHINE (JVM):- The JVM is the environment in which Java programs execute. It is software that is implemented on top of real hardware and operating system.
- When the source code (.java files) is compiled, it is translated into byte codes and then placed into (.class) files, The JVM executes these byte codes, So Java byte codes can be thought of as the machine language of the JVM.
- A JVM can either interpret the byte code one instruction at a time, or the byte code can be compiled.
- The JVM must be implemented on a particular platform before compiled programs can run on that platform.

8) What is a scope of variable? Explain the types of variables.
Ans:- SCOPE OF VARIABLES
- A variable's scope is the region of a program within which the variable can be referred to by its simple name.
- Scope is distinct from visibility, which applies only to member variables and determines whether the variable can be used from outside of the class within which it is declared.

3 Categories:-1) Instance Variables (Non-Static Fields):- Technically speaking, objects store their individual states in "non-static fields", that is, fields declared without the static keyword.
- Non-static fields are also known as instance variables because their values are unique to each instance of a class.

2) Class Variables (Static Fields): - a class variable is any field declared with the static modifier; this tells the compiler that there is exactly one copy of this variable in existence; regardless of how many times the class has been instantiated.

3) Local Variables:- Similar to how an object stores its state in fields, a method will often store its temporary state in local variables.
- The syntax for declaring a local variable is similar to declaring a field (for example, int count = 0 ;)
- There is no special keyword designating a variable as local; that determination comes entirely from the location in which the variable is declared.

9) Explain Data types used in Java.
Ans: DATA TYPES: - This indicates the type of data. That is used to process in program.
- The Java programming language is statically-typed, which means that all variables must first be declared before they can be used.

<u>8 Basic Data types are</u>
i) Byte: - The byte data type is an 8-bit signed two's complement integer.
- It has a minimum value of -128 and a maximum value of 127 .
- The byte data type-can be useful for saving memory in large arrays.

ii) Short: - The short data type is a 16-bit signed two's complement integer.
- It has a minimum value of -32,768 and a maximum value of 32,767.

iii) Int: - The int data type is a 32-bit signed two's complement integer.
- It has a minimum value of -2,147,483,648 and a maximum value of 2, 147, 483, 64.

iv) Long: - The long data type is a 64-bit signed two's complement integer.
- It has a minimum value of -9,223,372,036,854,775,808 and a maximum value of
9, 223, 372, 036, 854, 775, 807.

v) Float: - The float data type is a single-precision 32-bit floating point. As with the recommendations for byte and short, use a float (instead of double) if you need to save memory in large arrays of floating point numbers.

vi) Double: - The double data type is a double-precision 64-bit floating point. Its rang of values is for decimal values, this data type is generally the default choice.

vii) Boolean: - The Boolean data type has only two possible values: true and false. t this data type for simple flags that track true/false conditions.

viii) Char:-The char data type is a single 10-hit Unicode character.
- It has a minimum value of 11u0000' (or 0) and a maximum value of \ (or 65,535).

10) What is Constant? Explain its types
Ans:- Constants :- Constant means never changing once values are assigned to the constants at declaration time cannot be changed at the time of execution.
Java supports several types of constants those are:

1) Numeric Constants -->
This 2 types:- a) Integer constants b) Real constants.

a) Integer constants: This are may be decimal or hexadecimal. This can range in magnitude from negative to positive 231-1. Constants larger than these permissible magnitudes will produce unpredictable results.
ex: 037, 0 0435. 0551

b) Real Constants:- Quantities are represented by number containing fractional parts like 17.543. Such numbers are called as 'Real constants'
- A real number may also be expressed in exponential notation.
For example: The value of 215.65 may be written as 2.1565e2 in exponential notation. E2 means multiply by 102.
General Form is: mantissa e exponent

2) Character Constants: It consists of a single character enclosed in single quotes. Control characters can be placed in character constants by a # character followed by an integer constant (decimal or hexadecimal) corresponding to the desired ASCII value.
Ex: 'a' '1' ';' '#10' '#$1A'

11) What are Variables? Explain declaration of variable.
Ans:- VARIABLES:
- A variable is an identifier that denotes a storage location used to store a data value.
- Constants that remains unchanged during the execution of a program.
1) Average 2) Height
Some conditions of variables in java, that are:-
→ They must not begin with digit
→ It should not be keyword.
→ White space not allowed.

Declaration of variables
- In Java, variables are the names of storage locations. After designing variable names, we must declare to the compiler.

Declaration does three things:-
1) It tells the compiler what the variable name is.
2) It specifies what type of data the variable will hold.
The general form of declaration of a variable is:-

Type variable!, variable2 ... variable N
Variables are separated by commas; a declaration statement must end with semicolon.

Chapter-2
Operators and Expressions

1) Explain Different Operators used in Java.

Ans: i) Arithmetic Operators:

Arithmetic operators are used in mathematical expressions in the same way that they are used in algebra. The following table lists the arithmetic operators.

Operator	**Description**	**Example**
+	→Adds values on either side of the operator.	A + B
-	→Subtracts right-hand operand from left-hand operand.	A – B
*	→ Multiplies values on either side of the operator.	A * B
/	→ Divides left-hand operand.	B /A
%	→ Returns remainder.	B % A

ii) The Relational Operators: These operators are used to check the Condition.

There are following relational operators supported by Java language.

Operator	**Description**	**Example**
1) == (equal to)	Checks if the values of two operands are equal or not, if yes then condition becomes true.	(A == B) is not true.
2) != (not equal to)	Checks if the values of two operands are equal or not.	(A != B) is true.
3)> (greater than)	Checks if the value of left operand is greater than the value of right operand, if yes then condition becomes true.	(A > B) is not true.
4)< (less than)	Checks if the value of left operand is less than the value of right operand, if yes then condition becomes true.	(A < B) is not true.
5)>= (greater than or equal to)	Checks if the value of left operand is greater than or equal to the value of right operand, if yes then condition becomes true.	(A >= B) is not true.

iii) The Logical Operators:- Used to check TRUE OR FALSE.

Operator	Description	Example
1) && (logical and)	Logical AND operator. If both the operands are non-zero, then the condition becomes true.	(A && B) is false
2) || (logical or)	If any of the two operands are non-zero, then the condition becomes true.	(A || B) is true
3) ! (logical not)	Use to reverses the logical state of its operand. If a condition is true then Logical NOT operator will make false.	!(A && B) is true

iv) The Assignment Operators:

Operator	Description	Example
1) =	Simple assignment operator. Assigns values from right side operands to left side operand.	C = A + B will assign value of A + B into C
2) +=	Add AND assignment operator. It adds right operand to the left operand.	C += A is equivalent to C = C + A
3) -=	Subtract AND assignment operator. It subtracts right operand from the left operand	C -= A is equivalent to C = C – A

Chapter – 3
Decision Making and Branching

1) Explain All Control statements used program.

Ans:- 1) SIMPLE IF

The if-then or simple if, statement is the most basic of all the control flow statements.

Syntax	**For example**
if (<<condition>>)	**Program to check whether a person is eligible for voting */**
{	Class person
--------	{
}	public static void main (String arg[])
Statements;	{
	int age=19;
	if(age>=18)
	}
	System.out.println("Eligible for voting");
	}
	}

2) IF...ELSE.... (IF-THEN-ELSE STATEMENT)

The if-then-else statement is control statement that provides a secondary path of execution when an "if" clause evaluates to false.

Syntax:	**For example**:
if (<<condition>>)	**Program to check whether a student is passed or failed ?**
{	Class student
-------------;	{
Statements;	public static void main (String arg[])
}	{
Else	int per=40;
{	if(per>=35)
-----------------;	{
Statements;	System.out.println("The Student is passed");
}	}
	else {
	System.out.println("The Student is Failed");
	}
	}
	}

3) IF...ELSE IF...ELSE (LADDER IF STATEMENT):-

- The ladder if statement provides a multiple path execution when an "if" clause evaluates to false instead of going directly to else we can check some other conditions by specifying them in a else if ( ) clause.
- The only difference with simple if ... else is in multiple conditions if one is true but also next if statements will be checked but in ladder if any one condition will be executed.

Syntax:-

```
if (
<<condition>> )
{
Statements;
}
else if (
<<condition >>
)
{
Statements;
}
```

For example:-

/* The program assigns a grade based on the value of a test score; an a score of 90% or above, a B for a score of 80% or above, and so on.*/

```
Class grade
{
public static void main(String[]
args)
{
int testscore = 76;
char grade;
if (testscore >= 90)
{
grade = 'A';
}
else if (testscore >= 80)
{
grade = 'B'
}
 else if (testscore >= 70)
{
 grade = 'C';
}
else if (testscore >= 60)
{
grade = 'D';
}
else ( grade = 'F';  }
System.out.println("Grade = " +
grade);
}
}
```

4) NESTED IF STATEMENT

- If statement within another if statement is referred to as Nested If.

Syntax:-

```
if ( <<condition>> )
{
if ( <<condition>> )
{
Statements;
}
else
{
Statements;
} }
Else {
{
Statements; }
```

2) Explain The ? : OPERATOR with example.

Ans:- THE ? : OPERATOR

The ? Operator is used to rewrite the program

In Java we write

```
if (a > b)
{
max = a;
else
{
 max = b;
```

}
- Using the conditional operator you can rewrite the above example in a single line like this
max = (a > b) ? a : b;
(a > b) a: b;
- Here, is an expression which returns one of two values a or b.
- The condition, (a > b) is tested. If it is true the first value, a, is returned. If it is false, the second value, b, is returned.
- Whichever value is returned is dependent on the conditional test, a > b. The condition can be any expression which returns a **Boolean value**.

For example:

/* The programfind biggest of two numbers using? : Operator*/

```
class Big
{
        public static void main(String args[])
        {
        int a=20,b=30;
        string res="";
        res= a>b ? "A is Greater" : "B is Greater";
        System.out.println(res);
        }
 }
```

3) Explain looping Statements in Java?

Ans:- A loop is repeatedly executes the same set of instructions until termination met.

Types are:

1) <u>While Loop</u>:- This is java's most fundamental loop statement, it repeats a statements hile its controlling is become true.

- While loop is an entry conditioned loop.

Syntax:-

```
While (condition)
            {
             Statements;
            }
```

- The while statement evaluates expression, which must return a Boolean value.
- If the expression evaluates to true, the while statement executes the statement(s) in the while block.

Example:

Program to print 1 to 10 using while loop:

```
class number
{
public static void main(String[] args)
{
 int count = 1;
while (count < 11)
{
```

```
Systems.out.println("Count is: " + count);
count++;
}
}
```

2) DO WHILE LOOP

- It is Similar to the while statement, the do-while statement is a loop statement provided by Java.
- The difference between the do-while statement and the while statement is that in the while statement, the loop body is executed only when the condition stated in the statement is true. In contrast, in the do-while loop, the loop body is executed at least once.

Syntax:-	**Example:-**

do
{
Statements;
While (condition);
}
 	
- Each iteration of the **do-while** loop first executes the body of loop.
	Program to print 1 to 5 using do while:
	class number
	{
	public static void main(String[] args)
	{
	int n=0;
	Do
	{
	Systems.out.println(n);
	n++;
	}
	While (n<=5)
	} }

3) FOR LOOP:- The for statement provides a compact way to iterate over a range of values.
- Programmers often refer to it as the "for loop" because of the way in which it repeatedly loops until a particular condition is satisfied.

Syntax:

```
for (initialization; termination; increment)
{
Statements;
}
```

JUMPING STATEMENTS

a) The break statement

- The break statement has two forms:- labeled and unlabeled. We can also use an unlabeled break to terminate a for, while, or do-while loop, as shown in the following program

Example:-

Program to demonstrate Break statement:

```
Class sample
{
public static void main(String[] args)
{
For(int i=1; i<=10; i++)
{
If(i==5)
Break;
System.out.println(i);
          }
     }
}
```

b) The continue statement:- It skips current iteration of for, while, or do-while loop. The unlabeled form skips to end of the innermost loop's body.
- it evaluates Boolean expressions that controls the loop.

Example:-
Program to demonstrate Break statement:

```
Class sample
{
public static void main(String[] args)
{
For (int i=1; i<=10; i++)
{
If (i%2==0)
Continue;
System.out.println(i);
          }
      }
 }
```

Chapter -4
Packages and interfaces

1) What is package? Explain Java API packages.
Ans: - Packages are java’s way of grouping a variety of related classes, interfaces and sub-packages based on the functionality. Packages act as container for classes.
- Packages are both a naming and a visibility control mechanism.

Creating and Accessing the packages:-
For creating packages involves following steps:
a) Declare the package at beginning of the file.
Package <packageName>;
b) Define the class that is to be put in the package & declare it public
    public class MyClass {_ _ _ _}
c) Store listing as <ClassName>java file in subsidiary created.
d) Compile the file. This creates .class file in subsidiary.

Syntax:- package <pkg1>,<pkg2>.<pkg3>,…….;

Accessing the Packages:-
- A java package can be accessed using fully qualified class-name of the class or through ‘**import**’ statement that we use. This is done by using package name.
  Java.awt.color;
- This is also done using ‘Import’ statement at top of file.
    import<packagename>.<classname>;
  OR
    import<packagename>.*;

<u>**Java API Packages**</u>**:**

Name of the Packages	Contents
1)java.lang	1) Languages support classes.classes for primitive types strings, threads, exceptions.
2)java.util	2) Classes for vectors, hashtables, data etc.
3) java.io	3) input/output support classes.
4) java.awt	4) Classes for windows, graphics, lists etc.
5) java.net	5) Classes for networking.
6) java.applet	6) Classes for creating and implementing applets.

Adding a class to a package

- It is simple to add a class to an existing package. Consider the following package.

```
 Package P1
 Public class A
 {
// body of  A
  }
```

- The package P1 contains one public class by name. Suppose we want to add another class B to this package. This can be as follows:

1) Define the class and make it public.

```
  Package P1
Public class B
 {
// body of  B
  }
```

2) Store this as B.java under directory p1
3) Compile B.java file and creates B.class file. Place it in directory P1.

Chapter-5
Interfaces

1) What is Interface? How do you design interfaces?

Ans:- Java provides alternate approach known as interface to support the concept of 'multiple inheritance'.

- An interface is kind of class & interfaces are way to declare type consisting only of 'abstract methods & final fields'.

- Interfaces define only **abstract** & **final** fields.

General form:-
```
interface <InterfaceName>
{
<Final Fields>;
<Abstract Methods>;
}
```

Extending Interfaces:-

- One interface can inherit another by the use of keyword extends.

- When class implements an interface that inherits another interface.

Syntax:-
```
interface <Interface1> extends <Interface2>
{
<body of interface>;
}
```

Implementing Interfaces:- interface describes in pure abstract form. But it interested only if class implements it.

Syntax:-
```
class <ClassName> implements <InterfaceName>
{
 // body of class
}
```

Chapter -6
Threads

1) What is thread?
Ans:- A thread is a sequential flow of control within a program. It is a fundamental unit of program execution.

2) What is multithreading?
Ans:- The program containing two or more parts that can run concurrently is known as multithreading.
- On other hand, multithreading is ability of a single program to perform more than one task at same time called multithreading. Each part is called 'Thread'.

3) Explain the method of creating Threads.
Ans:- Thread:- Creating thread in java is simple. Threads are implemented in the form of objects that contain method called run ().

- Java defines **2 ways** in which threads can be created.
1) Declare a class that is sub class of **class Thread** define in java.lang package. Inside the sub class, override the run() method define in the Thread class.

Consider the following syntax:
Syntax: -

```
class MyThread extends Thread
{
// class definition
--------------------
 public void run()
  {
// implementation
  }
}
```

2) Declare a class that implements the **'Runnable in interface'**
- The easiest way to create a class that implements the Runnable Interface. It abstracts a unit of executable code.

Syntax:-

```
class MyThread implements Runnable
 {
// class definition
   ----------------
 Public void run()
  {
 // implementation
 }
}
```

Extending Thread Class:
- The second way to create a thread is to create new class that extends **Thread,** and then create instance of class.
- The extending class must override run() method.

Example:-
```
class SimpleThread extends thread
                 {
                 Public SimpleThread(String str)
                 {
                   Super(str);
                }
              Public void run()
                {
             For(int i=0; i<10; i++)
             {
             System.out.println(i++ “” + getName());
              Try   {
               Sleep (500);
                }
               catch (InterruptedException e) { }
                 }
                System.out.println(“DONE” + getname());
                }  }
             Public static void main(String args[])
             {
              new SimpleThread(“1234”).start();
            }
}
```

STOPPING AND BLOCKING A THREAD
a) Stopping:-Whenever we want to stop thread from running further, we use Stop() method.
    Syntax:- MyThread.sleep();
- This statement causes the thread to move to the dead state automatically when it reaches end of its method.

b) Blocking: - Threads can be blocked temporarily from entering into the runnable.
1. sleep():- Blocked for specified time.
2. suspend():- Blocked until further orders.
3. wait() :- Blocked until certain condition occurs.

4) Explain Life cycle of Thread.
Ans:- Life Cycle of Thread :-
Consider following stages in its life cycle.
1) New Born State:- The following statement creates new Thread but does not start it.
                              Thread myThread=New MyThreadClass();

- When a thread creates new thread, it is merely empty thread object. No system resources have been allocated for it.
- Calling any method besides start or stop when a thread in this state makes no sense and couses an ‘IllegealThreadStateException’.

2) Runnable and Running state:

Consider:-

```
Thread myThread=new MyThreadClass();
myThread.start();
```

- The start method creates the system resources necessary to run the thread, schedules the thread to run, and calls the thread’s run method.
- In this stage Thread is in ‘Runnable’ rather than “running” because the thread might not actually be running.
- The java run time system must implement a scheduling schema that shares processor between all runnable thread.

3) Blocked State:- A Thread become Block, when one of these events occurs:

a) Someone invokes its sleep method.
b) Someone invokes its suspend method.
c) The thread uses its wait method to wait.

Example:-

```
try{
        Thread.sleep(1000);
            }
catch (InterruptedExceptionc e)
      {
                  }
```

- During the 10 seconds that myThread is asleep; even if processor becomes available myThread does not run. After 10 seconds are up, my Thread becomes runnable again.

4) Dead State:- A thread can died in 2 ways; either from natural cases, or by being killed(stopped).
- A thread dies natural when its run method exists normally.

Syntax:-

```
public void run()
{
Int i=0;
While (i<100)
 {
i++;
System.out.println(“i=”+i);
}  }
```

- A thread with this run method dies naturally after the loop and run method complete.

myThread.stop():- The stop method throws a ThreadDeath object at thread to kill it.
- The thread will die when it actually receives the ThreadDeath exception.

Different Thread Methods are follows:

1) public void start()	- Starts thread in separate path of execution.
2) public void run()	- Here Thread object was initiated using separate runnable target run().
3) public final void setName(String name)	- Changes name of Thread object.
4) public void interrupt()	- interrupts this thread, causing it to continue execution if it was blocked.

4) Explain Thread Priority (scheduling).

Ans: - Every java has priority that helps operating system determine the order in which threads are scheduled?

ThreadName.setPriority(int number);

Java priorities are in range between:

MIN_PRIORITY	Constant of 1
MAX_PRIORITY	Constant of 10
NORM_PRIORITY	Constant of 5

- Threads with higher priority are more important to program and should be allocated processor time before lower priority threads.

Program example for Thread Priority:-

```
Class A extends Thread
{
public void run()
{
for(int i=1;i<=0;i++)
{
System.out.println("i="+i);
}
}}
class B extends Thread
{
public void run()
{
 for(int j=1;j=0;j++)
{
System.out.println("j="+j);
}
}
}
class C extends Thread
{
public void run()
{
for(int k=1;k=0;k++)
{
System.out.println("k="+k);
}
}}
class ThreadPrior
{
public static void main(String args[])
{
A thA=new A();
B thA=new B();
C thA=new C();
thA.SetPriority(1);
thB.SetPriority(3);
thC.SetPriority(9);
thA.start();
thB.start();
thC.start();
System.out.println("End of main");
}
}
```

5) Explain Thread Synchronization.

Ans:- When two threads need to use the same object, there is a possibility of interleaved operation that can corrupt the data. Java enables us to overcome this problem using technique called as Synchronization.
- The process by which this synchronization is achieved is called 'Thread synchronization'.
- This is **general form** of the synchronized statement:-

```
Synchronized (object)
{
// statements-------
}
```

- Here, object is a reference to the object being synchronized.
- A synchronized block ensures that a call to a method that is member of object occurs only after the current thread has successfully entered.

Program example for Thread synchronization:

Chapter – 7
Exception Handling

1) What is Exception? Explain exception handling Mechanism with syntax.
Ans:- Exceptions provides clean way to check for errors without clustering code. An exception is an abnormal condition that arises in a code sequence at Runtime.
- Exceptions can be generated by Java run-time system or manual code.
Exception Handling:- If we want to program to continue with the execution of remaining code, we should try to catch the exception object thrown by the errors condition & then display appropriate massage for taking corrective actions. This task is known as 'Exception handing'
Exception Handling performs following Tasks:
a) Find the problem (Hit the exception).
b) Inform that error has occurred (Thrown the exception).
c) Receive the error information (catch).
d) Take corrective actions (handling the exception).

General Form of Exception Handling:

```
try{
// code o monitor for error
}
catch (exception-type e1)
{
// exception handler for type 1
}
catch (exception-type e2)
{
 // exception handler for type 2
  - - - - - - - -  -
Finally {
// Block to be executed before try ends
}
```

2) What are the types of errors? And explain briefly the error handling in java.
Ans:- Error may produce incorrect output or may terminate the execution of program or may cause system to crash.

2 types of Errors:
1) Compile time error: - Compile time errors are all syntax errors detected & displayed by the java compiler. Whenever the compiler displays an error, it will create the .class file.
- Most Compile time errors are **'Missing Semicolon, braces'**.

2) Run time error:- A program may compile successfully by creating the .class file. However, it may not run properly. Such produce wrong results due to wrong logic or due to errors.
- Most run time errors are **'dividing the integer by Zero'**.

<u>Error Handling in Java:-</u>If we want to program to continue with the execution of remaining code, we should try to catch the exception object thrown by the errors condition & then display appropriate massage for taking corrective actions. This task is known as 'Error handing'

5 keywords are:

1) Try 2) catch 3) throw 4) throws 5) finally

3) Explain try, catch & finally blocks.

Ans:- <u>Try</u>:- Exception are caught by enclosing code in try block. Whenever there is chance of exception occur in the program, it is better to handle it explicitly. This can be done with help of Try block.

- Try block can have one or more statements that can generate an exception.

Syntax:- Try	**Example program of Try block:**

{
------------- 	
------------- 	
}
catch (exception- type e)
{
// statements
}
// statements
}
	Class DivideError
	{
	Public static void main(String args[])
	{
	int a=16, b=2, c=2;
	int x,y;
	try
	{
	X=a/(b-c);
	}
	catch (arithematicException e)
	{
	system.out.println("Divide by zero");
	}
	y=a/(b+c);
	system.out.println("y="+y);
	}
	}

Catch Block:- A catch block catches the exception thrown by the try block & handles it properly. The catch is added immediately after the try block.

- The catch block can have one or more statements that are necessary to process the exception.

Syntax:- catch (Exception type1)

```
            {
               --------------
               --------------    }
```

<u>Example program of catch block</u>:

```
class arithematicException
{
 Public static void main(String args[])
{
 int a=10,b=0, c;
 system.out.println("computing divison");
try  {
c=a/b;
system.out.println("result :" +c);
}
catch (Exception e)
{
system.out.println("Exception :" +e.getMassage());
}
}
```

Finally Block:- Java supports another statement finally that can be used to handle an exception that is not caught by any of previous catch statements.
- Finally block can be used to handle any exception generated within try block.
- Finally block creates its own reason to leave by executing break or return or by throwing an exception.
- Finally creates a block of code that will be executed after a try/catch block.

Syntax:- finally

```
            {
             ----------
            -----------
            Statements
            }
```

Example program of finally block:-

```
class  FinallyDemo
{
        Public static void main (String args[])
        {
        int a=10, b=0, c;
        system.out.println("computing divison");
        try
        {
                c= a/b;
                system.out.println("Result :"+c);
        }
                catch (Exception e)
                {
                system.out.println("Exception :"+e.getMassage());
                }
                Finally
                {
```

```
                system.out.println("Executing");
                 }
        }
  }
```

4) Explain with program Multiple Catch Blocks used in Exception Handling.

Ans:- It is possible to have more than one exception in catch block.

```
          Try {-------------------}
          catch {-----} {-------}
          catch {-----} {-------}
                  .
                  .
```

- More than one exception could be raised by single piece of code to handle this, we can specify two or more catch blocks.
- After one catch statement is executed, rest of the catch statements are ignored.
- In java, multiple catch() blocks are used to handle different exceptions.

Syntax:-

```
try  {
      Statements
       }
    catch (ArithmeticExcption e)
{
   statements
}
  catch (Exception e)
    statements
               }
```

<u>Example program of Multiple Catch block</u>:-

```
         class MultiCatch
         {
         public static void main (String args[])
         {
         try {
          int a=args.length;
          system.out.println("a= "+a);
         int b= 42/a;
         int c[] = {1};
         c [42]= 99;
         }
         catch (ArithmaticException e)
          {
         system.out.println("Divide by 0; "+e);
         }
         catch (ArrayIndexOutOfBoundsException e)
         {
         system.out.println ("Arrey index oob: "+e);
         }
         system.out.println ("After try/catch blocks");
         }
}
```

5) Explain Throw and Throws mechanism with syntax and program.

Ans:-a) Throw:- Exceptions are thrown using throw statement, which takes an object as its parameters.

```
        throw new <Throwable-subclass>;
```

- The flow of execution stops immediately after throw statement. The nearest enclosing try block is inspected to see if it has a catch statement that matches type of exception. If it does, control is transferred to that statement. If not, then next enclosing try statement is inspected and so on...
- Throw statement used explicitly throw an exception.

Example program of Thrown statment:-

```
   class DemoThrow
    {
        static void meth()
        {
        try
        {
         throw new NullPointerException("demo");
        }
                catch(NullPointerException e)
                {
                system.out.println("caught inside meth ()" );
                throw e;         // rethrow the exception
                }
         }
        public static void main (String args[])
        {
        try {
        catch (NullPointerException e)
                 {
                  system.out.println("Recught : +e);
                }
            }
        }
```

b) Throws: if a method is capable of causing a exception that it doesn't handle, this must specify this behaviour so that caller of that method can guard themselves against that exception. For this we use **throws** clause.

General Syntax:

```
 type method-name(parameter-list) throws exception-list
                       {
                        // body of method
                          }
```

- Here, exception list is comma-separated list of exception that a method can throw.

6) List all Java's Runtime Exceptions.
Ans:-

Exceptions	**Meaning**
1) ArithmeticException	**-** Arithmetic errors, such as divide-by-zero.
2) ArrayStoredException	- Assignment to an array element of an incompatible type.
3) ArrayIndexOutOfBoundsException	- Array index is out-of-bounds.
4) ClassCastException	- Invalid test.
5) IndexOutOfBoundsException	- some type of index is out-bounds

Chapter – 8
Applet Programming

1) What is Applet? How it differs from Java Applications? [2012]
Ans:- Applets are small or tiny java programs developed for internet applications and that can be embedded into HTML pages. Java applets loaded into servers can be download via internet and executed using browser which is java enabled.
- Applets are used to make web site more dynamic.

Example:
```
import java.applet;
 import java.awt.*;
 -------------------
Public class MyApplet extends Applet
    {
 --------------------------
    }
```

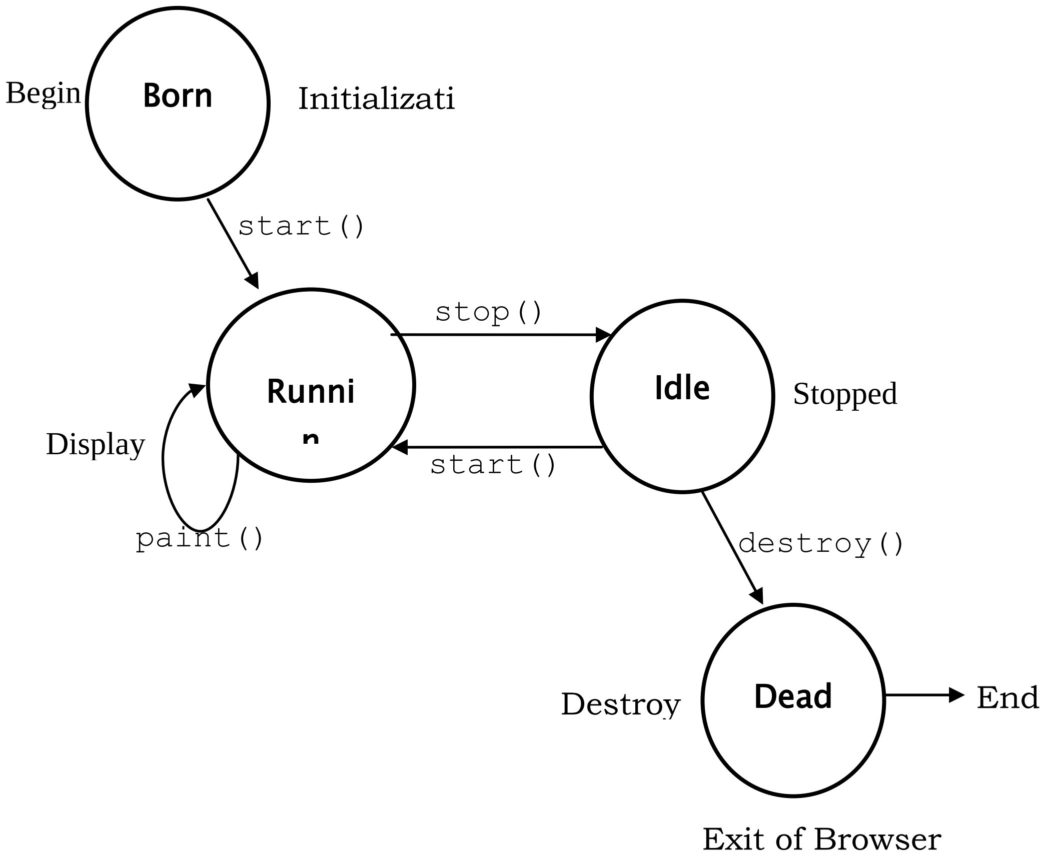
How applets are differs from Java Applications?
i) Applications must run in local machine, where as Applets needs no explicit installations on local machine.
ii) Applications must be run explicitly within a JVM, where as Applets loads and runs itself automatically in java-enabled browser.
iii) Applications starts execution with its main method(),where as Applets starts execution with its intit() method.
iv) Application can run with or without GUI, but Applet must run within a GUI.
v) Applets don’t have method main().

Types of Applets:
a) Local Applet:- An applet developed locally and stored in local system called local Applet. When web page trying to find local applet, it doesn’t need to use internet.
- It simply searches the directories on the local system.

b) Remote Applet:- It is developed by someone else and stored on remote computer connected to internet.

2) Explain Life Cycle of an Applet.
Ans:- An applet defines the structure with 4 states during the execution.
i) Born state
ii) Running state
iii) Idle state
iv) Dead state

Diagram (Applet Life cycle)



i) Born State:- When the class is executed the init() function is automatically invoked. At this stage applet is said to be born. This state is also called initialization state.
- During applet life cycle, the born state occurs only once in applet's life cycle.
- it is achieved by calling init() method of applet class.
Public void init() {-------------------}

ii) Running state:- Applet enters the running state when system calls start() method. It occurs automatically after the applet in initialized.
- It can also occur if applet is already in idle state. Start() method may be called more than once.
public void start() {------------------}

iii) Idle state:- An Applet become idle when it is stopped from running, stopping occurs when we leave the page containing currently running applet.
- We can call stop() method explicitly.
public void stop() {------------------}

iv) Dead state:- An Applet is said to be dead when it is removed from memory. It occurs automatically by invoking destroy() method.
- It can happen only once in applet's life cycle.
public void destroy() {------------------}

v) Display state:- Applet moves to this state when it has to perform some output operations on the screen. The paint() method is called to accomplish this task.

```
                    public void paint (Graphics g) {-----------------}
```

3) Explain Applet Tag with its Attributes.

Ans:- The APPLET tag is used to start applet from both HTML document and from an applet viewer.
- Applet tag supports number of standard attributes.

```
<APPLET
            [CODEBASE = codebaseVRL]
            CODE = appletFile
           [ALT =  alternateText]
           [NAME = appletInstanceName]
            Width = pixels height = pixels
          [ALIGN = alignments]
           [VSPACE = pixels] [HSPACE = pixels]
       >
          [<PARAMNAME = AttributeName1 VALUE = AttributeValue1>]
          [<PARAMNAME = AttributeName2 VALUE = AttributeValue2>]
          . . . . . . . . . . .
       </APPLET>
```

Attributes are:

1) CODEBASE: It specifies base URL of applet code, which is the directory that will be searched for applet's class.

2) CODE: It is required attribute that gives the name of the file containing applet's compiled.class code.

3) ALT: It is an optional attribute used to specify a short text massage display.

4) PARAM NAME: Allows us to specific applet arguments in an HTML page.

4)Explain Passing Parameter to Applet from HTML with program.

Ans:- We can pass user-defined parameters to an applet using <PARAM....> tags. Each <PARAM....> tag has a name and value attribute.
- Inside applet code, the applet can refer to that parameter by name using getParamater() to find its value.
- Method returns the value in the form of String object.
→ To set up and handle parameter, we need to do two things:
    - Include appropriate <PARAM...> tag in the HTML tag.
    - Provide code in the applet to parse these parameters
- Parameters are passed to an applet when is it loaded. We can define init() method in the applet.

Program example:
```
import java.io.*;
import java.applet.*;
public class DemoParam extends Applet
{
String str;
public void init()
{
Str= getParamater(“String”);
if (str==null) str=”Java”;
str =”Hello” + str;
}
public void paint(Graphics g) {
g.drawString(str, 100,100);
}
 }
```
HTML Code:
```
<APPLET CODE = “DemoParam.class”
   WIDTH = 200 Height = 200>
<PARAM NAME = “String” value = “Applet”>
</APPLET>
```

4) Write a Java Applet program to demonstrate Mouse Events.
Ans:
```
import java.awt.*;
          import java.applet.*;
          import java.awt.event.*;
        public class MouseEvents extends Applet implements MouseListener,  MouseMotion
Listener
       {
       string msg= “  ”;
       int mouseX=0, mousey=0;
       public void init();
       {
      addMouseListener(this);
      addMouseMotionListener(this);
      }
      public void paint(Graphics g)
       {
public void mouseClicked (MouseEvent me)
      {
      mouseX=0, mouseY=10;
      msg = “Mouse Clicked”;
      repaint();
      }
public void mouseEntered(MouseEvent me)
```

```
       {
        mouseX=0, mouseY=10;
        msg = “MouseEntered”;
         repaint();
       }
public void mousePressed(MouseEvent me)
         {
         mouseX=me.getX();
         mouseY= me.getY();
         msg = “Down”;
         repaint();
         }
public void mouseRealeased(MouseEvent me)
         {
         mouseX=me.getX();
         mouseY= me.getY();
         msg = “Up”;
         repaint();
         }
public void mouseDreagged((MouseEvent me)
      {
         mouseX=me.getX();
         mouseY= me.getY();
         msg = “*”;
showStatus(“dragging mouse at “+mouseX+” , “+mousey);
         repaint();
         }
public void mouseMoved(MouseEvent me)
         {
         Showstatus(“Mousing mouse at “+me.getX()+”, “me.getY());
         }
}
```

5) Write a Java Applet program to demonstrate KeyBoard event.
Ans:-

```
         import java.awt.*;
          import java.applet.*;
          import java.awt.event.*;
        public class simpleKey extends Applet implements KeyListener
         {
         String msg = “ ”;
         int X=10,  Y=20;
         public void init()
         {
                  addKeyListener(this);
                  requestFocus();
```

```
    }
    public void paint(Graphics g) {
    g.drawString(msg, X,y);
    }
    public void KeyPressed(KeyEvent ke)
    {
            showStatus(“KeyDown”);
    }
    public void KeyReleased(KeyEvent ke)
    {
            showStatus(“KeyUp”);
    }
    public void KeyTyped(KeyEvent ke)
    {
            msg +=ke.getKeyChar();
            repaint();
    }
  }
```

Chapter-9
Stream Classes

1) What is Stream Classes? Explain its types.
Ans: A stream represents the sequence or flow of data, or channel of communication with writer at one end and a reader at other end.
- A stream is abstraction that either produces or consumes information.
- A stream is linked to physical devices by java I/O system.

Types:
1) Input Streams: An input Streams extracts or reads the input data from the source and sends it to program.

Two types:
a) Byte stream Classes: Input stream classes that are used 8-bits bytes. though there are many classes related to byte streams but most frequently used classes are FileInputStream and FileOutputStream.

b) Input stream classes: input stream classes are the base class of all input streams in the java IO API. InputStream subclasses include the FileInputStream, BufferedInputStreams.

<u>Methods</u>	Description
1) int read()	- Reads the single byte data from the stream.
2) int read(byte[])	- Reads data into an array of bytes. it is determined by length of byte array passed.
3) long skip(long)	- Jumps over the specified number of bytes in streams.
4) int available()	- Returns the number of bytes that can be read without blocking.

2) Explain File Input Stream classes.
Ans: This used to obtain input bytes from a file. It is used for reading streams of raw bytes such as image data.
Program for input Stream class:
```
import java.io.*;
class SimpleRead
{
public static void main(String Agrs[])
{
try{
FileInputStream fin = new FileInputStream("abc.txt");
int i=0;
while (i=fin.read())!=1)  {
System.out.println(char);
}
fin.close();
}
catch(exception e)
{
System.out.println(e);
}
}
```

Important Method in Files class:
1) String getName() → The getName method gets the name of file & returns it as string.
2) String getPath() → The getPath method returns the path of file.
3) String getParent() → The getParent method returns the parent directory of file.
4) long length() → The length method returns length of file in bytes.

3) What are Piped Streams?
Ans: Pipe provides the I/O based mechanism for communicating data between different threads. The safe way to use piped stream is with two Thread: one is reading & one for writing.
- Writing on one end of pipe blocks the thread when the pipe fills up.
- The piped input stream contains the buffer, decoupling read operations from write operations within limits.
- A pipe is said to be broken if a threads that was providing data bytes to the connected piped output stream is no longer available.

Chapter – 10
Classes, Arrays, Strings and Vectors

1) What is class? Explain with example.
Ans:- Class is a user-defined data type or blueprint using which we can create objects. In java, without class, we can write the programs.
- Classes contain all the features of particular set of objects.
- Object is also called instance or copy of the class.
Syntax: class class_name

```
            {
            fields declaration[member variables];
            Constructor declarations; [special initialization methods]
            Methods declarations [member functions];
            }
```

- This is a class declaration. The class body contains all the code that provides for the life cycle of the objects created from the class.

2) What is Constructor? Explain its types.
Ans: Constructor is a special member function or method of a class. Which is used to initialized the objects.
- The name of the class & constructor must be same.
- Constructors have no return type. Like methods, we can give access specifier to the constructors.
Syntax: class [class_name]

```
           {
            Data members;
            Public [class name] (paramaterts...)
           {
           Statements ;
          }
         }
```

Types:i) Simple constructors
ii) Default constructors
iii Parameterized constructors
iv) Copy constructors

3) What is method overloading?
Ans:- Overloading methods provides more than one method with the same name but with different signature type to distinguish them.
- Overloading is the technique of giving the extra or enhanced meaning for the existing one.
- Method Overloading is the example of OOp feature Polymorphism.
Ex: void add(int x, int y)...

```
    void add(int x, int y. int z). . .
    void add(double x, double y)
```

Two rules to Overloaded Methods:

i) Return type of methods can be same, but argument list of overloaded methods must differ in number or in data type.
ii) Argument list of calling statement must differ enough to allow unambiguous determination of proper method to call.

Program example for Method Overloading:

```
Class Object
{
Double findArea(int r)
}
return (meth.PI*r*r) ;
}
double findArea(int b, int h)
{
return(0.5*b*h);
}
}
```

```
class Test()
{
public static void main(String args[])
{
double A=0.0;
Object circle=new Object();
A=circle.findArea(7);
System.out.println("Circle Area : "+A);
Object triangle=new object()
System.out.println("Triangle Area : "+A);
}
}
```

4) Differentiate between Overloading V/s Overriding.
Ans:

<u>**Overloading**</u>	<u>**Overriding**</u>
1) Same method names but different arguments may or may not be same return type written in the same class itself. 2) Method name same but signature is different in the class.	1) Same method names with same arguments and same return types associated in class & its subclass. 2) To re-define the basic class method in the derived class is called overriding.

Chapter-11
Inheritance

1) What Multiple Inheritances? How you implement in Java?
Ans: Multiple inheritances is a feature of Opp's, in which an object or a class can inherit characteristics & features from more than one parent object or parent class.
- It is distinct from single inheritance, where an object or class may only inherit from one particular object or class.
- Deriving more than one class from parent class in known as '**Multiple Inheritance**'.

Example:
```
class A
      int x;
      int y;
     int get (int p, int q)  {
     x=p;  y=q; return(0);
    }
   void Show()
   {
   System.out.println(x);
   }   }
   class B extends A
   {
System.out.println("B");
         }   }
         class C extends B
         {
         void display()   {
         }
         public static void main(String args[])
         A a=new A();
         a.get(5,6);
         a.Show();
         }
      }
```

2) What is Super Keyword? Explain its uses.
Ans:-Super keyword used to invokes super class constructors or parent class constructor
- The Super keyword is used to access the members of the super class that has been hidden by a member of a subclass.
- Whenever a sub class needs to refer to its immediate super-class, it can do so by use of the keyword 'Super'.

Using super to call super-class Constructors
A subclass can call a constructor defined by its superclass by use of following form of super.

```
super (arg-list);
```

- Here, arg-list specifies any arguments needed by the constructors in the superclass.super() must always be first statements executed inside a subclass constructor.

3)What is Final Keyword? Explain its uses.
Ans:- It is used to creates the equivalent of named '**constants'**& final is used to apply inheritance.
Ex: final int a=10;

Uses of final keyword:

a)Using final to prevent Overloading:

- To avoid a method from being overridden, specify **final** as a modifier at the start of its declaration.
- Method declared as final cannot be overridden.

Following examples shows the Final keyword:

```
 class A{
        final void main()
                {
                System,out,println("this is a final method");
                }
                }
            class B extends A {
             void method()
            System,out,println(" Illegal");
            }
}
```

- Here math() is declared as final, it can not be overridden in B. suppose if we are trying do so, we will get compile-time error.

b) Using final to prevent Inheritance:

- Here we want to prevent class from being inherited. to do this, class declaration with **final.**

Example: final class A

```
          {
        // . . . .
         }
      // the following class is illegal
        class B extends A
        {
        // . . . . .
        }
```

4) What is Narrowing and Widening?

Ans: **Narrowing:-** Converting Larger data type to smaller data type without losing data called Narrowing.

Widening:- Converting smaller data type to Larger data type with losing data called Widening.

5) Explain Mathematical Functions used in java.

Ans: sin(x), cos(x), tan(x), cosec(x), sec(x)
sqrt(), avg(), max(), min(), pow(x,y), abs(a)

6) What are Abstract methods and classes?

Ans: Class such as number, which represents an abstract concept and should not be initiated, is called abstract class. An abstract class is class that can only be subclasses
- it cannot be initiated.

- To declare the abstract class, we use keyword 'abstract' before the class keyword in class declaration:

```
abstraction class Number
{
. . . . .
}
```

- A abstract class may contain abstract methods.

7) Explain Visibility Controls / Access specifiers used in Java

Ans: The 3 access specifiers **Private**, **Public**, and **Protected** provide a variety of ways to produce the many levels of access required by these categories.
- Anything declared public can be accessed from anywhere.
- Anything declared private cannot be seen outside of its class.
- When a class is declared as public, it is accessible by any other code. If a class has a default access, then it only be accessed by other code within class.
- The protected fields or methods cannot be used for class and interfaces. fields, methods and constructors declared protected in super class can be accssed only by subclass in other package.

Chapter-12
Arrays, Strings and Vectors

1) What is single dimensional Arrays? Explain Creating and Declaration of Array

Ans: A list of items can be given one variable name using only one subscript. And such variable is called a single or one dimensional array.
- One-dimensional array is data structure which allows a collective name to be given to group of elements which have the same type.

Declaration: type var-name[];
- Here the type declares the base type of the arrays.

Example: int A = new int [5];

Creating an Array: like any other variables, arrays must be declared and created in computer memory before they are used.

3 steps involved:
i) Declaration
ii) Allocation
iii) Initialization

i) Declaration:- Like other variable in java, an array must have a specific type like byte, int, string or double. only variables of appropriate type can be stored in an array.

```
int[] k;
float [] t;
String[] names;
```

ii) Creation of Arrays: When we create an array we need to tell the compiler how many elements will be stored in it?

```
      k= new int[2];
      t= new float[5];
     names = new String[50];
```

- We create an array with operator 'new'. It is reserved word in java that is used to allocate not just an array.

iii) Initialization: Array can be initialized when they are declared. An array initialize is a list of comma-separated expressions.
- The array will automatically create large enough to hold number of elements.
- Individual elements of array are referenced by the array name and by an integer which represents their position in the array.

```
class Array
 {
public static void main(String args[])
{
 int month_days[] = {31,28,31,30,31,31,30,31,31};
}
System.out.println("April has " + month_days[3]= "days.");
}
}
```

2) Explain Multidimensional Arrays.(declaration, initialization)

Ans: An array with values of two or more values is called multidimensional array.

- To declare a multidimensional array variable, specify each additional index using another set of square brackets.

Declaration: int mulD [] [] [] = new int [9] [6] [4];

```
mulD[0] = new int[5];
mulD[1] = new int[5];
mulD[2] = new int[5];
mulD[3] = new int[5];
```

- When we allocate the memory for a multidimensional array, we need only specify the memory for the first dimension. we can allocate remaining dimensions separately.

Chapter-13
Strings

1) What is String? Explain Constructors used in strings.
Ans: Strings are used to represent the sequence of characters. String class is used to represent a string.
- The string type is used to declared string variables.

Constructing New Strings:

```
String str1 = new String();
String str2 = new String("A new string");
StringBuffer buf = new StringBuffer{"buffer"};
```

There are 6 string constructors are shown below:

Constructors	**Purpose**
1) Strings()	- Creates empty string.
2) String(String)	- Creates string from specified string.
3) String(char[])	- Creates string from an array of character.
4) String(char[],int, int)	- Creates string from specified subset of character in array.
5) String(byte[],int)	- Creates string from specified byte array.
6) String(StringBuffer)	- Creates string.

2) Explain Basic String Methods with program.
Ans:

Method	**Purpose**
1) concat(String)	- Concatenate one string into another.
2) compareTo(String)	- Compare two strings. returns 0 if they are equal.
6) copyString()	- Copies one string into another.
3) length()	- Returns length of the string.
4) toLowerCase()	- Converts entire string to lowercase.
5) toUpperCase()	- Converts entire string to uppercase.
7) charAt(int)	- Returns the character to location.
8) valueOf(char)	- Returns string containing one specified character.

Program to Demonstrate All String Methods:

```
import java.io.*;
class StringOperations
{
public static void main(String args[])
{
String first = “ “, second= “”;
DataInputString dis= new DataInputString(System.in);
System.out.println(“String Operation”);
 System.out.println();
System.out.println(“Enter the first string:”);
first = dis.readLine();
System.out.println(“The strings are: “+first+ “,” “+second);
System.out.println(“Enter the second string:”);
second=dis.readLine();
System.out.println(“The strings are: “+first+ “,” “+second);
System.out.println(“The length of the first string is: “+second.length());
System.out.println(“The Concatination of first and second string is: “+first. concat(“ ”
+second);
System.out.println(“The UpperCase of  “+first+” is: “+ first.toUpperCase());
System.out.println(“The LowerCase of  “+first+” is: “+ first.toLowerCase());
String str= dis.readLine();
char c = str.charAt(0);
System.out.println("The " + c + " occurs at position " + first.indexOf(c) + " in " + first);
System.out.println("Replacing a with o in "+first+" is: "+first.replace("aVo"));
boolean check=first.equals(second);
if(!check)
System.out.println(first + " and " + second + " are not same.");
else
System.out.println(first + " and " + second + " are same.");
}
}
```

3) What is String Buffer Class?

Ans: StringBuffer is peer class of String that provide much of the functionality of strings. StringBuffer represents growable and writable character sequences.
- StringBuffer may have character and substrings inserted in middle.

StringBuffer Constructors:

i) StringBuffer()
ii) StringBuffer(int size)
iii) StringBuffer(String str)
iv) StringBuffer(CharSequence char)

Some of StringBuffer Methods:

Methods	Purpose
1) Capacity()	i) Returns current capacity of StringBuffer
2) charAt(int)	ii) Returns character located at index.
3) ensureCapacity()	iii) Ensures the capacity of StringBuffer at least specified amount array.

Chapter-14
Vectors

1) What is a Vector class? How do u create Vector class?
Ans: Vectors are commonly used instead of arrays, because they expand automatically when new data is added to them.
- Vectors can hold only objects & not primitive types (eg. int). If we want to put a primitive types in a vector put it inside an object.

Vectors posses a number of advantages over Arrays:
i) It is convenient to use vectors to store objects
ii) A vector can be used to store a list of objects that may vary in size.
iii) We can add & delete objects easily.

Creating Vectors:
a) Create a Vector with default initial size
Vector v = new Vector();
b) Create a Vector with an initial size
Vector v = new Vector(300); // to add elements to end of vectors

Chapter-15
Wrapper Class

1) Write a note on Wrapper Class.
Ans: Wrapper class is wrapper around a primitive data type. It represents data types in their corresponding class instances.
- All primitive wrapper class in Java are immutable. i.e once assigned a value to wrapper class instance cannot be changed.

Features of Wrapper Class:
i) All the methods of Wrapper class are Static.
ii) The Wrapper class does not contain constructors.
iii) Once the value is assigned to wrapper class instance it cannot be changed.