}

The switch statement tests the value of a given variable or expression against a list of case values and when a match is found, a block of statements associated with that case is executed.

9Q) Explain Looping Statements.

Ans) The Looping statements are used to repeat set of statements based on a condition. The looping statements are 1)while 2) do while and 3) for.

while

```
while(condition)
{
    ----------
    ----------   } Statements
}
```

If the condition is true, then the body of the loop is executed. This process is repeated until the condition is false.

do-while

```
do
{
    ----------
    ----------   } Statements
}while(condition);
```

The program proceeds to evaluate the body of the loop first. At the end of the loop the test condition is evaluated. If the condition is true, then the body of the loop is executed. This process is repeated until the condition is false.

for

```
for(initialization ; test condition ; increment/decrement)
{
```



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

Initialization of the variable is done first. Next the condition is evaluated, if the condition is true, the statements are executed. Next the control goes to increment/decrement. Again the condition is evaluated. This process is repeated until the condition is false.

break: used to stop the loop.

continue: used to skip the next statements of continue statement and continues to next iteration of the loop.

10) Explain Input and Output in java (or) (a) Java.util.Scanner Class (b) System.out.printf() (c) String.format()

Ans) **Java.util.Scanner** : Scanner is a class in java. util package used for obtaining the input of the primitive types like int, double, etc. and strings. It is the easiest way to read input in a Java program.

To use the Scanner class, create an object of the class and use any of the available methods found in the Scanner class documentation.

Syntax: Scanner ObjectName = new Scanner(System.in);

Example : Scanner myObj = new Scanner(System.in);

We use the following different methods to read data of various types in Scanner class.

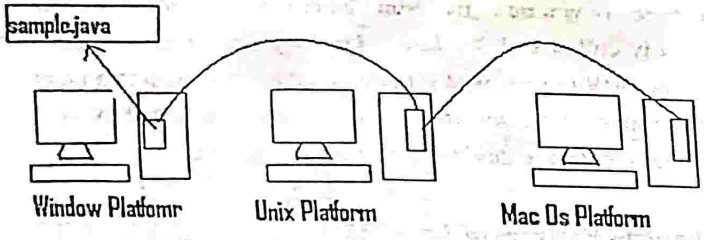
Method	Description
nextBoolean()	Reads a boolean value from the user
nextByte()	Reads a byte value from the user
nextDouble()	Reads a double value from the user
nextFloat()	Reads a float value from the user
nextInt()	Reads a int value from the user
nextLine()	Reads a String value from the user
nextLong()	Reads a long value from the user
nextShort()	Reads a short value from the user

Portable

Programs developed in the java, can be run any operating environment. Because of the memory space for the all variables are same in any environment.

Platform Independent

Programs develop in one operating system, and then we can run those programs in any operating system.



If you write one java file in Window platform and take that file and execute in UNIX platform then that java program will give the same result. That why java program is the independent language. The Sun Microsystems has given that "write once run anywhere". (Note: In all the platform Java must be installed).

Multithreaded

Performing more than one job at a time (concurrently) is the main goal of either multitasking or multithreading. In case of multitasking, multiple processes are required to perform multiple jobs. Whereas, in multithreading a single program performs multiple jobs concurrently. Java has built-in features to implement multithreaded applications. Java's contribution towards multithreading is that it has brought system level programming concept to application level.

Distributed

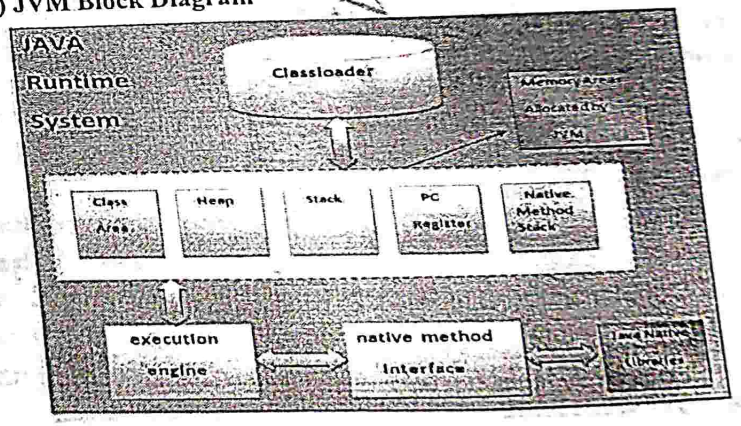
Java has rich libraries for network programming and distributed computing. As java is mainly meant for Internet based programming, it is no surprise that java is distributed. An application is said to be distributed if the business objects are geographically dispersed in the network and communicating one another.

Interpreted

Java is both compiled and interpreted. Java programs are complied and generate byte code. This byte code can be downloaded and interpreted by the interpreter. Java uses JIT (Just In Time) complier. The purpose of this JIT to increase the execution of the program.

2Q) Explain about JVM(Java Virtual Machine)?

Ans) JVM Block Diagram



Class loader

A class loader subsystem loads the .class file into the memory and verifies the byte code instructions and then allows required memory

to execute instructions. This allocated memory is divided in to five parts.

Method Area

Byte code instructions will be loaded into method area and it will allocate the memory for the instance variables. And it contains the class code and along with the method code.

Heap Area

Heap is second part of memory. It contains objects. Heap is used for creating objects.

Java Stack

Purpose of Java method is executed method parameters and temporary results method return results are stored in java stack.

PC Register

PC (Program Counter) these registers store the address of the instructions begins executed by the User Program.

Native Method Stack

Native Method Stack are method written in some other language like c or c++ ., depending up on the situation it will get executed.

Execution Engine

It contains interpreter and JIT (Just In Time) complier which translates the byte code instructions into machine language which are executed by user program. JIT complier and interpreter executes simultaneously. Native Method interface is useful to linkup with the native method or JVM (Native method is nothing but the OS methods).

3Q) Explain Structure of Java program. (or) Explain Parts of Java Program.

Ans)

- ✈ Documentation section
- ✈ Package statements
- ✈ Import statements
- ✈ Interface statements
- ✈ Class definition
- ✈ Main method class definition

```
{
    Statements;
}
```

The documentation section contains the name of the program, the author name, version number and other details. /**.....*/ is used to denote documentation.

Package is nothing but a directory, it contains classes/interfaces.

Import statement is similar to #include statement in c and c++. It is used to import classes/interfaces.

An interface is like a class but it contains method declarations.

A java program may contain any no. of classes. classes are primary and essential elements of a java program.

Every java program requires a main method. Program execution starts from main. Without main method we can't execute a java program.

Output statement

System.out.println is the output statement in java.

Syntax

System.out.println ("output string" + variable);

Here System is the class, out is an object of System class and println is the method. It always appends a new line to end of the string.

A simple java program

```
C:\> EDIT First.java
class First
{
    public static void main(String args[ ])
    {
        System.out.println("First java program");
    }
}
```

Here class is a keyword, it declares a class and First is an identifier, it specifies the class name.

Every class definition starts with opening braces ({) and ends with closing braces (}).

public, static, void and **main** are key words. **public** is an access specifier, it declares that the main is accessible to all other classes. **static** declares main as one that belongs to the entire class and not a part of any object. **void** is a type modifier, states that main does not return any value.

String args[ ] declares a parameter named args, which contains an array of objects of the class type **String**. It handles command line arguments.

Note: java extension is .java

Every statement in java program ends with semicolon (;)

To Compile:

C:\> javac First.java

To Execute:

C:\> java First

After compilation java compiler creates a **.class** file, by using class name.

The java interpreter executes **.class** file. i.e. we must specify class name at the time of execution.

4) Explain Naming Conventions in Java.

Ans) Java is a case sensitive language, so it follows some rules. The rules to be followed by the programmers by writing class name, method name, and variable declaration etc. in the java programs is called "Naming Conventions".

There are six rules in the java naming conventions…

1. Package Names in java are written in all small letters. Ex: java.awt:

java.io; javax.swing;

2. Class Names & Interface Names are start with capital letters.

Ex: String, DataInputStream, ActionListener

3. Method Names start with a small letters, then each word start with capital letters.

Ex: println (), readLine (), getNumberInstance ()

4. Variable also fallows the method Naming convention rule (i.e. Method name rules)

Ex: empName, emp_Net_Sal, cityName

5. Constant variable name should be written using all capital letters

Ex: PI, MAX_VALUE, FONT_BOLD

6. All Key words should be written in all small letters.

Ex: public, void, import

5Q) Explain Data Types in Java.

Ans) Java language has 8 primitive data types.

- int
- short
- byte
- long
- float
- double
- char
- boolean

int, short, byte and long are integral data types. All these data types are used to declare variables that can store non-decimal numbers.

Type	Size	Minimum Value	Maximum Value
byte	1Byte	-128	127
short	2 Bytes	-32768	32767
int	4 Bytes	-2147483648	2147483647
long	8 Bytes	-9223372036854775808	9223372036854775807
float	4 Bytes	3.4e-038	3.4e+038
double	8 Bytes	1.7e-308	1.7e+308
char	2 Bytes	It can hold only single character	
boolean	1 Byte	It can hold true or false keywords.	

Numeric values that are not integral are stored in floating-point numbers. float and double are used to store decimal numbers. If we need accuracy up to 7 digits after decimal point, we use float type. If we need accuracy up to 17 digits after decimal point, we use double type.

Floating-point literals are by default double type. If we want to specify a floating point literal, we should explicitly mention f or F.

Example: float a=45.6F;

Char primitive type variable occupy 2 bytes of memory in Java. Java supports Unicode characters. A Unicode is a specification or standards to include alphabet of all international languages into java character. Unicode character takes 2 bytes of memory. While assigning a character literal to a variable in Java, we have to enclose it in single quotes (similar to C and C++).

Example: char grade='A';

We can use arithmetic on char variables.

grade +=1; //Earlier grade holds 'A' and now it holds B.

Variables of type boolean can have only one of two values, true or false.

Example:

boolean ismanager=false;//ismanager variable holds false literal now.

6Q) Explain literals in Java.

Ans) A literal is a constant value written in source code. They appear literally in the code and may not change within a particular program run.

Literals in Java can be classified into five types. They are

1. Integral Literals
2. Floating-point Literals
3. Char Literals
4. String Literals
5. Boolean Literals

Integral Literals: An integral literal is a numeric value without any fractional or exponential part.

Example:
```
int binaryNumber = 0b10010;
int octalNumber = 027;
int decNumber = 34;
int hexNumber = 0x2F;
```

Floating-point Literals:

A floating-point literal is a numeric literal that has either a fractional form or an exponential form.

Example:
```
double myDouble = 3.4;
float myFloat = 3.4F;
double myDoubleScientific = 3.445e2;  // 3.445*10^2
```

Character Literals

Character literals are unicode character enclosed inside single quotes.

Example: char letter = 'a';

String literals

A string literal is a sequence of characters enclosed inside double-quotes.

Example: String str1 = "Java Programming";

String str2 = "Government College (A)";

Boolean Literals

In Java, boolean literals are used to initialize boolean data types. They can store two values: true and f

Operator	Meaning
+	Addition or unary plus
-	Subtraction or unary minus
*	Multiplication
/	Division
%	Modulo division

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

3. Logical operators

Operator	Meaning
&&	Logical AND
||	Logical OR
!	Logical NOT

4. Assignment operators (=)

Assignment operators are used to assign the value of an expression to a variable.

Ex : int a=10;

5. Increment and Decrement operators

Operator	Meaning
++	Increment (Adding 1 to value)
--	Decrement (Subtracting 1 from value)

Ex: 1) int a=b++ (post increment) 2) int a=++b (pre increment)

3) int a=b– (post decrement) 4) int a=--b (pre decrement)

6. Conditional operator (or) Ternary operator : (?:)

This operator is used to construct conditional expressions.

Syntax: expression1 ? expression2 : expression3 ;

Ex: int a=10,b=35;

Int x=(a>b) ? a : b;

Here if expression1 is true expression2 is evaluated. i.e x=a.

if expression1 is false expression3 is evaluated. i.e x=b.

7. Bitwise operators

These operators are used for testing the bits or shifting them to the right or left.

Operator	Meaning
&	Bitwise AND
!	Bitwise OR
^	Bitwise exclusive OR
~	One's complement
<<	Shift left
>>	Shift right
>>>	Shift right with zero fill

8. Special operators:

Operator	Meaning
.	Member selection operator
instanceof	Object reference operator
new	is useful to create an object to a class

Java Operator Precedence or Priority

Precedence	Operator	Type	Associativity
1	() [] .	Parentheses Array subscript Member selection	Left to Right
2	++ --	Unary post-increment Unary post-decrement	Left to Right
3	++ -- + - ! ~ (*type*)	Unary pre-increment Unary pre-decrement Unary plus Unary minus Unary logical negation Unary bitwise complement Unary type cast	Right to left
4	* / %	Multiplication Division Modulus	Left to right
5	+ -	Addition Subtraction	Left to right
6	<< >> >>>	Bitwise left shift Bitwise right shift with sign extension Bitwise right shift with zero extension	Left to right
7	< <= > >= instanceof	Relational less than Relational less than or equal Relational greater than Relational greater than or equal Type comparison (objects only)	Left to right

8	== !=	Relational is equal to Relational is not equal to	Left to right
9	&	Bitwise AND	Left to right
10	^	Bitwise exclusive OR	Left to right
11	|	Bitwise inclusive OR	Left to right
12	&&	Logical AND	Left to right
13	||	Logical OR	Left to right
14	? :	Ternary conditional	Right to left
15	= += -= *= /= %=	Assignment Addition assignment Subtraction assignment Multiplication assignment Division assignment Modulus assignment	Right to left

8Q) Explain branching statements in java (OR) Explain Conditional control statements in java.

Ans) A set of statements are executed by using a condition it is called conditional statements.

Conditional control statements are **if-else** and **switch**.

if-else statement

The if-else statement is conditional control statement. The Java *if statement* is used to test the condition. It checks Boolean condition: *true* or *false*. There are various types of if statement in java.

- if statement
- if-else statement
- nested if statement
- if-else-if ladder

Simple if

if(condition)

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

Here if condition is true, the statements of if block will be executed. Otherwise the statements will be skipped

if – else

```
if(condition)
{
----------
----------  Statements.
}
else
{
----------
----------  Statements.
}
```

Here if condition is true, the statements of if block will be executed. If condition is false the statements of else block will be executed.

Nested if – else

```
if(condition 1)
{
----------
----------  Statements 1.
if(condition 2)
{
----------
----------  Statements 2 .
}
}
```

Here if condition1 is true, the statements 1 is executed and Condition 2 is true statements 2 is executed otherwise the statements will be skipped

if-else-if ladder

```
if(condition1)
```



```
{
----------
----------  Statements1.
}
clse if(condition2)
{
----------
----------  Statements2.
}
else
{
----------
----------  Statements3.
}
```

Here if condition1 is true, the statements1 will be executed. If condition1 is false and condition2 is true the statements2 will be executed. If condition1 and condition2 are false the statements3 will be executed.

Switch

```
switch(expression)
{
 case value1:
----------
----------  Statements1
 case value2:
----------
----------  Statements2
--
--
 default:
----------
----------  Statements
```

Examples :

String name = myObj.nextLine();
int age = myObj.nextInt();
double salary = myObj.nextDouble();

System.out.printf()

The System.out.printf() function in Java allows users to print formatted data.printf() comes from the C programming language and stands for print formatted.

Syntax: System.out.printf(format, arguments);

Here format contains format specifiers and escape characters.

%s – formats strings
%d – formats decimal integers
%f – formats the floating-point numbers
%t – formats date/time values

escape characters

\b - Insert backspace
\n - Insert newline
\t - Insert tab
\\ - Insert backslash

Example :

1) System.out.printf("Hello %s!%n", "World"); Hello World!

2) int x = 100;

System.out.printf("Printing simple integer: x = %d\n", x);

String.format()

*The java string format() method returns the formatted string by given locale, format and arguments.*The format() method of java language is

like sprintf() function in c language and printf() method of java language.

There are two type of string format() method:

1) public static String format(String format, Object... args)

and,

2) public static String format(Locale locale, String format, Object... args)

Here,

locale : specifies the locale to be applied on the format() method.

format : format of the string.

args : arguments for the format string. It may be zero or more.

Example : String name="GC(A)";

String sf1=String.format("name is %s",name);

String sf2=String.format("value is %f",32.33434);

11Q) Explain about Arrays in java.

Ans) An array is a group of similar data elements. It allocates memory contiguously.

In java there are two types of arrays.

1). Single dimensional Arrays 2). Multi dimensional Arrays

Single Dimensional Array:

Syntax : datatype varname[ ]=new datatype[size];
datatype[ ] varname=new datatype[size];
datatype varname[ ];
varname[ ]=new datatype[size];

Here new is an operator, it allocates memory dynamically.

Size is the maximum no. of values an array can holds.

Ex. For dynamic initialization:

int a[ ]=new int[5];

for(int 1=0;i<5;i++)

a[i]=i;

Ex. for compile time initialization

int a[ ]={1,2,3,4,5};

Multi Dimensional Array:

Multi dimensional means 2D, 3D....etc. It is divided into rows and columns.

2D Arrays

Syntax:

1. datatype varname[ ][ ]=new datatype[row size] [column size];
2. datatype[ ][ ] varname=new datatype[row size][column size];

datatype varname[ ][ ];

varname[ ] [ ]=new datatype[row size][column size];

Ex: int a[ ][ ]=new int[2][3];

Here array 'a' holds 2X3 values. i.e 6.

Initialization:

```
int a[2][2]={{1,2},{4,3}};  (or)  for(int i=0;i<2;i++)
                                      for(int j=0;j<2;j++)
                                      a[i][j]=(i+j);
```

3D Arrays

Example : int a[ ][ ][ ]=new int[2][2][2];

Here array 'a' holds 2X2X2 values. i.e. 8.

Initialization:

```
int a[2][2][2]={{{1,2},{4,3}},{{5,6},{7,8}};
(or)  for(int i=0;i<2;i++)
                                      for(int j=0;j<2;j++)
                                      for(int k=0;k<2;k++)
                                      a[i][j][k]=(i+j+k);
```

arrayname.length

Array has a property length, it calculates the size of an array.

Ex: int a[ ]={2,2,3,5,6};

int b=a.legth;

here the value of b is 5.

12Q) Explain Command Line arguments.

Ans) The java command-line argument is an argument i.e. passed at the time of running the java program.

The arguments passed from the console can be received in the java program and it can be used as an input.

So, it provides a convenient way to check the behavior of the program for the different values.

Example: Lab program 1

UNIT – 2

1Q) Problems in Procedure Oriented Approach

Ans)

Procedural Oriented Programming is a programming language that follows a step-by-step approach to break down a task into a collection of variables and routines (or subroutines) through a sequence of instructions.

The key features of procedural programming are as follows:

Predefined functions: A predefined function is typically an instruction identified by a name.

Local Variable: A local variable is a variable that is declared in the main structure of a method and is limited to the local scope it is given.

Global Variable: A global variable is a variable which is declared outside every other function defined in the code.

Modularity: *Modularity is when two dissimilar systems have two different tasks at hand but are grouped together to conclude a larger task first.*

Parameter Passing: *Parameter Passing is a mechanism used to pass parameters to functions, subroutines or procedures.*

2Q) Explain principles of object oriented programming (or) Explain Features of Object-Oriented Programming System (OOPS).

Ans) An object oriented programming language is the one style of writing the programs. An OOP is the core of java. So to start learning the java it is recommended to understand the OOP concepts before you begin writing even a simple java program. The following are the important oops features…

1.) class
2.) object
3.) Encapsulation
4.) Inheritance
5.) Polymorphism

1.) Class

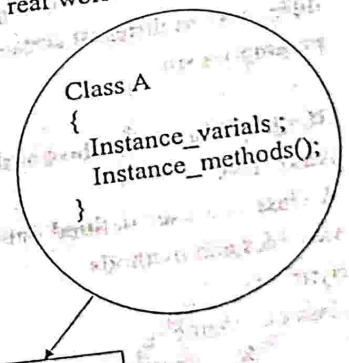
A class is model for creating objects. A class cannot exits physically. A class contains the all instance variables and functions/methods. Without creating a class we can't create object.

Note:- Without creating a single class you can't write a single sample java program.

2.) Object

An object is an anything that exits in the real world. An object is the instance of the class.

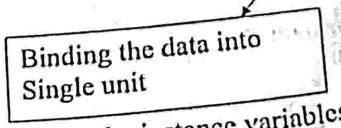
By using object only we can access the instance variables and methods.



3.) Encapsulation

Binding the data into one single unit is called the encapsulation.

Example is the class,

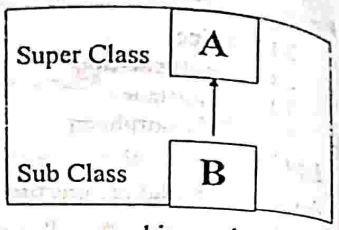


Where a class contains the all the instance variables and methods.

In the figure we declare the all the instance variables and method in one single class called A. There are some mechanism for hiding the complexity of the implementation of the class, i.e. each variable and method can also be declared as private or public or protected.

4.) Inheritance

A process of acquiring properties of from one class to another class is called the inheritance.



In the figure A,B are the two classes , Where A is the super class and B is the sub class, Hence now the B having the properties of A. By this inheritance we can achieve the code reusability. In java we have different types of inheritance.

5.) Polymorphism

Polymorphism is the Greek word "poly" means many "morphism" means forms , complete meaning is many forms. It is one feature that acts differently depending up situations. An object behaving differently in different situations is nothing but the object is exhibiting polymorphism.

3Q) Explain about class and Object.

Ans) class

A class is a user defined data type. It is the combination of member variables and methods.

Syntax:

```
class classname
{       Member variables;
          Methods;
}
```

Here class is a keyword.

A method is nothing but a function.

```
Ex:  class A
     {     int a;
           void read()
           {
                 a=23;
           }
     }
```

Object: An object is an instance of class or a blue print of class.

Syntax: 1) classname object=new classname();

2) classname Object;
Object =new classname(); obj=new A();

Ex: A obj=new A(); (or) A obj;

4Q) Explain about Methods

Ans) A method is a block of code or collection of statements grouped together to perform a certain task or operation. It is used to achieve the reusability of code. We write a method once and use it many times.

A method must be declared within a class. It is defined with the name of the method, followed by parentheses ().

Syntax : Access_specifier return_type Method_Name(parameters);

Example : public void myMethod(int a,int b)

```
{
      int x,y,z;
      x=a;
      y=b;
      z=a+b;
}
```

Example Program : Lab programs

5Q) Explain about Constructors in Java

Ans) A constructor is a specialized method in Java whose name and class name is the same and is called automatically as soon the object

is created. Primary purpose of a constructor is object initialization. If we don't write any constructor in a class, Java compiler provides a zero argument constructor for the class by default. We can write our own zero argument constructors also in a class. We can have parameterized constructor also in a class

Example Program : Lab program

6Q) Explain Static methods in Java.

Ans) If we apply static keyword with any method, it is known as static method.

- A static method belongs to the class rather than object of a class.
- A static method can be invoked without the need for creating an instance of a class.
- Static method can access static data member and can change the value of it.

Example Program : Lab program

7Q) Define Inheritance? Explain the types of inheritance?

A. Definition:

Inheritance is a process of creating new class from the existing class. The existing class is known as "base class" and the newly created is known as "derived class".

By process of inheritance the derived class gets all the features of base class and apart from this it can have its own features. The base class is not affected by the process of inheritance. Inheritance is most useful and essential characteristics of object oriented programming

"The major advantage of inheritance is code reusability i.e., reusing the properties of base class in the derived class."

Types of Inheritance:

The derived class inherits all the features of base class based on access specifier and level of inheritance. Based on this relationship inheritance can be classified in to 6 forms.

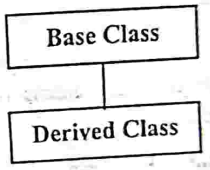
They are:

1. Single Inheritance
2. Multilevel Inheritance
3. Hierarchical Inheritance

Single Inheritance

Derivation of a class from only one base class is known as single inheritance.

In single inheritance there is one base class and one derived class.



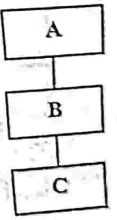
Syntax:

```
class derivedclass extends baseclass
{
    Body of derived class
}
```

Multilevel Inheritance: Derivation of a class from another derived class is known as multi-level inheritance.

In the figure Class A is base class and class B is derived from A.

Further the class C is derived from B. Here the Class C can access the members of both A and B.



The class B is called intermediate base class which provides a link for inheritance between A and C. The syntax of multi-level inheritance is:

```
class A
{
    Members of class A
```

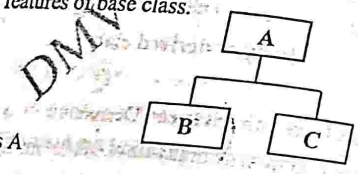
```
}
class B extends A
{
    Members of class B
}
class C extends C
{
    Members of class C
}
```

Hierarchical inheritance:

Derivation of several classes from single base class is called hierarchical inheritance.

Here "A" is base class and "B","C" are derived classes. Each derived class inherits the features of base class.

Ex:-




```
class A
{
    Members of class A
}
class B extends A
{
    Members of class B
}
class C extends A
{
    Members of class C
}
```


8Q) Explain Strings in Java.

Ans) Generally, string is a sequence of characters. But in java, string is an object that represents a sequence of characters. The java.lang.String class is used to create string object.

There are two ways to create String object:

- By string literal
- By new keyword

1) String Literal

Java String literal is created by using double quotes. For Example:

```
String s="welcome";
```

Each time you create a string literal, the JVM checks the string constant pool first. If the string already exists in the pool, a reference to the pooled instance is returned. If string doesn't exist in the pool, a new string instance is created and placed in the pool.

For example:

```
String s1="Welcome";
String s2="Welcome"; //will not create new instance
```

2) By new keyword

```
String s=new String("Welcome");
//creates two objects and one reference variable
```

In such case, JVM will create a new string object in normal (non pool) heap memory and the literal "Welcome" will be placed in the string constant pool. The variable s will refer to the object in heap (non pool).

Immutability of Strings

In Java, String objects are immutable. The string is immutable means that we cannot change the object itself, but we can change the reference to the object.

Example :

```
String s="Govt";
s.concat(" College"); //concat() method appends the string at the end
System.out.println(s); //will print Govt because strings are immutable
s=s.concat(" College");
System.out.println(s); // will print Govt College
```

String Class Methods

Java String class provides a lot of methods to perform operations on string such as concat(), split(), length(), replace(), compareTo(), substring() etc.

Method	Description
substring()	Returns a new string which is the substring of a specified string
compareTo()	Compares two strings lexicographically
concat()	Appends a string to the end of another string
isEmpty()	Checks whether a string is empty or not
length()	Returns the length of a specified string
split()	Splits a string into an array of substrings
toLowerCase()	Converts a string to lower case letters
toUpperCase()	Converts a string to upper case letters
replace()	Searches a string for a specified value, and returns a new string where the specified values are replaced

UNIT – 3

1Q) Explain Polymorphism in java.

Ans) Polymorphism is an important concept of object-oriented programming. It simply means more than one form.

Polymorphism with Variables

A polymorphic Variable is a variable that can reference more than one type of object. Polymorphic variables derive their power from interaction with inheritance, overriding and substitution. A common polymorphic variable is the implicit variable that maintains the receiver during the execution of a method.

Polymorphism using Methods

Method Overloading is a feature that allows a class to have two or more methods having same name, if their argument lists are different. Constructor overloading that allows a class to have more than one constructor having different argument lists.

Argument lists could differ in –

1. Number of parameters.
2. Data type of parameters.
3. Sequence of Data type of parameters.

Method overloading is also known as **Static Polymorphism.**

Example Program: Lab program

2Q) Explain final variables, final methods and final classes.

Ans) The **final keyword** in java is used to restrict the user. The java final keyword can be used in many context. Final can be:

1. variable
2. method
3. class

1. Final Variable

The final keyword can be applied with the variables, a final variable that have no value it is called blank final variable or uninitialized final variable. It can be initialized in the constructor only. The blank final variable can be static also which will be initialized in the static block only.

Ex: final int speedlimit=90;//final variable

2. Final Method

The final keyword in Java can also be applied to methods. A Java method with the final keyword is called a final method and it can not be overridden in the subclass.

Example : final void run()

```
{
    System.out.println("running");
}
```

3. Final class

Java class with the final modifier is called the final class in Java. The final class is complete in nature and can not be sub-classed or inherited. Several classes in Java are final e.g. String, Integer, and other wrapper classes.

Example : final class Bike

```
{
}
```


3Q) Explain Type Casting

Ans) Assigning a value of one type to a variable of another type is known as Type Casting. In Java, type casting is classified into two types.

They are,

1) Implicit type casting

2) Explicit Type casting.

Implicit Type casting: Automatic Type casting take place when,

- the two types are compatible
- the target type is larger than the source type

byte → short → int → long → float → double

Example:

```
public class Test
{
  public static void main(String[] args)
  {
    int i = 100;
    long l = i;     //no explicit type casting required
    float f = l;    //no explicit type casting required
    System.out.println("Int value "+i);
    System.out.println("Long value "+l);
    System.out.println("Float value "+f);
  }
}
```

Output:

Int value 100

Long value 100

Float value 100.0

Explicit type casting:

When we are assigning a larger type value to a variable of smaller type, then we need to perform explicit type casting.

double⟶float⟶long⟶int⟶short⟶byte

Example:

```
public class Test
{
  public static void main(String[] args)
  {
   double d = 100.04;
   long l = (long)d;  //explicit type casting required
   int i = (int)l; //explicit type casting required
   System.out.println("Double value "+d);
   System.out.println("Long value "+l);
   System.out.println("Int value "+i);
     }
  }
```

Output :

Double value 100.04
Long value 100
Int value 100

4Q) Explain Abstract methods and Abstract classes

Ans) Abstraction is a process of hiding the implementation details and showing only functionality to the user. Java supports two types of abstraction.

1. Abstract class

2. Abstract method

Abstract class: A class that is declared with abstract keyword, is known as abstract class in java. It can have abstract and non-abstract methods

Example : abstract class A
{
}

Abstract Method: A method that is declared as abstract and does not have implementation is known as abstract method.

Example : abstract void printStatus(); //no body and abstract

Example program:

```
abstract class Bike
{
  abstract void run();
}
class Honda4 extends Bike
{
void run()
{
  System.out.println("running safely..");
}
public static void main(String args[])
{
  Bike obj = new Honda4();
  obj.run();
}
}
```

OUTPUT
running safely..

5Q) Explain about Interfaces?

Ans) An interface in java is a blueprint of a class. It has static constants and abstract methods.

The interface in java is a mechanism to achieve abstraction. There can be only abstract methods in the java interface not method body. It is used to achieve abstraction and multiple inheritance in Java.

Example :

```
interface Animal
{
        public void eat();
        public void travel();
}
```

There are mainly three reasons to use interface. They are given below.

- It is used to achieve abstraction.
- By interface, we can support the functionality of multiple inheritance.
- It can be used to achieve loose coupling.

Example : Lab program

6Q) Explain Packages.

Ans) A java package is a group of similar types of classes, interfaces and sub-packages. Package in java can be categorized in two form, built-in package and user-defined package. There are many built-in packages such as java, lang, awt, javax, swing, net, io, util, sql etc.

Example :

- **java.lang** – bundles the fundamental classes
- **java.io** – classes for input , output functions are bundled in this package

Creating a package

Creating a package in java is quite easy. Simply include a package command followed by name of the package as the first statement in java source file.

Example :

```
package mypack;
public class Employee
{
  statement;
}
```

The above statement will create a package with name mypack in the project directory.

To compile the Java programs with package statements, we have to use -d option as shown below.

javac -d Destination_folder file_name.java

Example :

javac -d . Employee.java

Accessing/using a Package

import keyword is used to import built-in and user-defined packages into java source file so that the class can refer to a class that is in another package by directly using its name.

If we import packagename.classname, then only the class with name classname in the package with name **packagename** will be available for use.

Example : import mypack.employee;

If we use packagename.*, then all the classes and interfaces of this package will be accessible but the classes and interface inside the subpackages will not be available for use.

Example : import mypack.*

Subpackages in Java

Package inside the package is called the subpackage. It should be created to categorize the package further.

Example :

```
package mypack.p1;
```

7Q) Explain Access specifiers or Access Modifiers in Java.

Java provides four types of access specifiers that we can use with classes and other entities.

These are:

1) Default: Whenever a specific access level is not specified, then it is assumed to be 'default'. The scope of the default level is within the package.

2) Public: This is the most common access level and whenever the public access specifier is used with an entity, that particular entity is accessible throughout from within or outside the class, within or outside the package, etc.

3) Protected: The protected access level has a scope that is within the package. A protected entity is also accessible outside the package through inherited class or child class.

4) Private: When an entity is private, then this entity cannot be accessed outside the class. A private entity can only be accessible from within the class.

Access Specifier	Inside Class	Inside Package	Outside package subclass	Outside package
Private	Yes	No	No	No
Default	Yes	Yes	No	No
Protected	Yes	Yes	Yes	No
Public	Yes	Yes	Yes	Yes

8Q) Explain managing Errors and Exceptions.

Ans) An Exception is a condition that is caused by a runtime error in the program. An error may produce an incorrect output or may terminate the execution of the program.

There are two types of errors.

1. Compile time errors
2. Run time errors

All syntax errors will be detected and displayed by the java compiler and therefore these errors are known as compile time errors.

Ex: Misspelled variable and function names, Missing semicolons, improperly matches parentheses, etc.

A **runtime error** means an **error** which happens, while the program is running.

Ex: Dividing an integer by zero, converting invalid string to a number, accessing an element that is out of the bounds of an array, etc.

Exception handling:

The **exception handling in java** is one of the powerful *mechanism to handle the runtime errors* so that normal flow of the application can be maintained.

There are 5 keywords used in java exception handling.

1. try
2. catch
3. finally
4. throw
5. throws

try block

Java try block is used to enclose the code that might throw an exception. It must be used within the method.

catch block

Java catch block is used to handle the Exception. It must be used after the try block only.

finally block

Java finally block is a block that is used to execute important code such as closing connection, stream etc.

Java finally block is always executed whether exception is handled or not.

throw keyword

The Java throw keyword is used to explicitly throw an exception.

We can throw either checked or unchecked exception in java by throw keyword. The throw keyword is mainly used to throw custom exception.

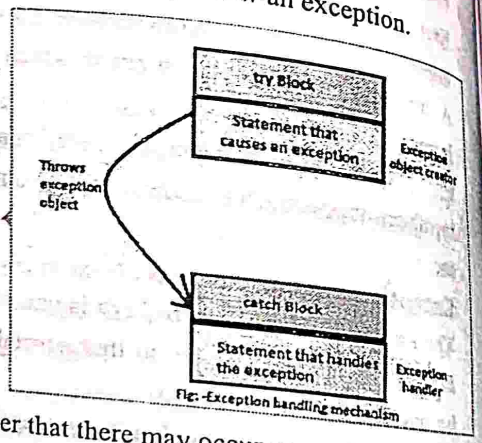


Fig: -Exception handling mechanism

throws keyword

The Java throws keyword is used to declare an exception. It gives information to the programmer that there may occur an exception so it is better for the programmer to provide the exception handling code so that normal flow can be maintained.

Example : Lab Program

UNIT - 4

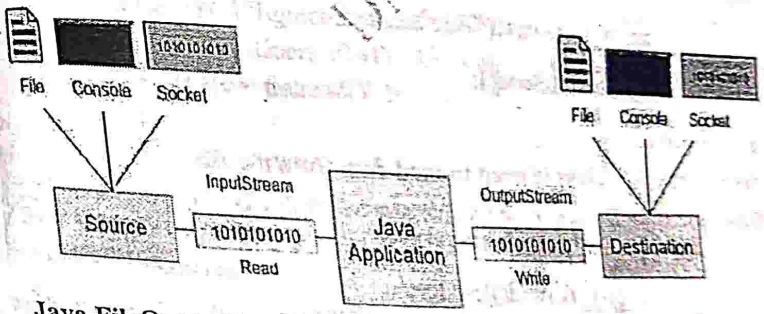
1Q) Explain managing input/output files in java (or) define stream? Explain

Ans) Java I/O (Input and Output) is used *to process the input* and *produce the output*. Java uses the concept of stream to make I/O operation fast. The java.io package contains all the classes required for input and output operations.

We can perform **file handling in java** by Java I/O API.

Stream

A stream is a sequence of data. In Java a stream is composed of bytes. It's called a stream because it is like a stream of water that continues to flow.



Java FileOutputStream

Java FileOutputStream is an output stream used for writing data to a file.

Example:
```
FileOutputStream fout=new FileOutputStream ("File1.txt");
fout.write(65);
fout.close();
```

Java FileInputStream

Java FileInputStream class obtains input bytes from a file. It is used for reading byte-oriented data (streams of raw bytes) such as image data, audio, video etc.

```
Example: FileInputStream fin=new FileInputStream("File1.txt");
         int i=fin.read();
         System.out.print((char)i);
         fin.close();
```

2Q) Write about file concepts in java (or) Write about a)FileWriter b) FileReader c) File Class d) File Copy

Ans)

Java FileWriter

Java FileWriter class is an abstract class which is used to write file character streams.

```
Example : FileWriter myWriter = new FileWriter("File1.txt");
          myWriter.write("File created using FileWriter");
          myWriter.close();
```

Java FileReader

Java FileReader class is used to read data from the file.

```
Example : FileReader fr=new FileReader("File1.txt ");
          int i;
          while((i=fr.read())!=-1)
               System.out.print((char)i);
          fr.close();
```

Java File Class

The File class is an abstract representation of file and directory pathname. A pathname can be either absolute or relative.

The File class has several methods for working with directories and files such as creating new directories or files, deleting and renaming directories or files, listing the contents of a directory etc.

```
Example: File file = new File("File1.txt");
         if (file.createNewFile()) {
           System.out.println("New File is created!");
         } else {
           System.out.println("File already exists.");
         }
```

File Copy

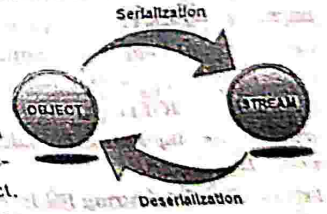
Using File Stream method we can copy data from one file to another. Here we use a file input stream to get input characters from the first file and a file output stream to write output characters to another file. This is just like seeing one file and writing onto another.

```
Example: FileInputStream fis=new FileInputStream(
                    "C:\\Users\\ DMV\\Desktop\\input.txt");
         FileOutputStream fos = new FileOutputStream(
                    "C:\\Users\\DMV\\Desktop\\output.txt");
         while ((c = fis.read()) != -1)
         {
              fos.write(c);
         }
```

3Q) Write about Serialization of Objects.

Ans) **Serialization**

Serialization in Java is a mechanism of writing the state of an object into a byte-stream.



The reverse operation of serialization is called deserialization where byte-stream is converted into an object. The serialization and deserialization

process is platform-independent, it means you can serialize an object on one platform and deserialize it on a different platform.

For serializing the object, we call the writeObject() method of ObjectOutputStream class, and for deserialization we call the readObject() method of ObjectInputStream class.

4Q) Write about Zipping and Unzipping Files in Java.

Ans) Zipping and Unzipping Files in Java

ZIP is a common file format that compresses one or more files into a single location. It reduces the file size and makes it easier to transport or store. A recipient can unzip (or extract) a ZIP file after transport and use the file in the original format.

Zipping : We can zip files in Java using the core Java libraries java.util.zip package.

Example:
```
FileOutputStream FoS = new FileOutputStream ( "
        dirCompressed.zip " ) ;
ZipOutputStream zipOut = new ZipOutputStream( FoS ) ;
```

Unzipping : We can unzip files in Java using the core libraries of java.util.zip package.

Example:
```
FileInputStream FiS = new FileInputStream( zipFilePath ) ;
ZipInputStream ZiS = new ZipInputStream( FiS ) ;
```

5Q) Write a program counting number of characters in text file using java.

Ans)
```
import java.io.BufferedReader;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.InputStreamReader;
public class Tester {
  private static final String FILE_PATH = "data.txt";
```



```
public static void main(String args[]) throws IOException {
                    FileUtil fileUtil = new
No. of characters in file:  FileUtil(FILE_PATH);
                    System.out.println("" +
fileUtil.getCharCount());
  }
}
class FileUtil {
  static BufferedReader reader = null;
  public FileUtil(String filePath) throws FileNotFoundException {
    File file = new File(filePath);
    FileInputStream fileStream = new FileInputStream(file);
    InputStreamReader input = new InputStreamReader(fileStream);
    reader = new BufferedReader(input);
  }
  public static int getCharCount() throws IOException {
    int charCount = 0;
    String data;
    while((data = reader.readLine()) != null) {
      charCount += data.length();
    }
    return charCount;
  }
}
```

OUTPUT
No. of characters in file:25

6Q) Explain Multi threading.

Ans) **Multithreading in java** is a process of executing multiple threads simultaneously. Thread is basically a lightweight sub-process, a smallest unit of processing. Multiprocessing and multithreading, both are used to achieve multitasking.

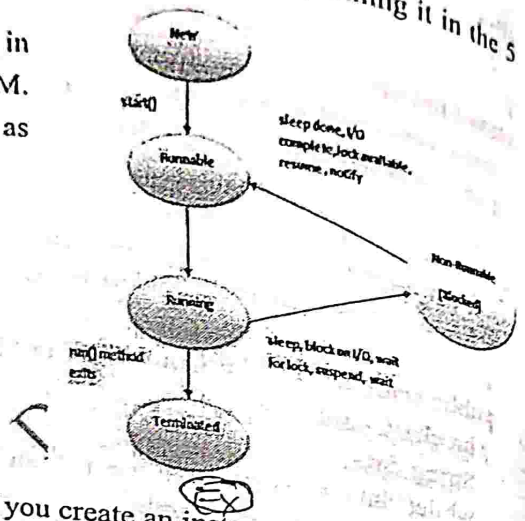
Life cycle of a Thread

A thread can be in one of the five states. According to sun, there is only 5 states in thread life cycle in java new, runnable, running, non-runnable and terminated. There is no running state.

But for better understanding the threads, we are explaining it in the 5 states.

The life cycle of the thread in java is controlled by JVM. The java thread states are as follows:

1. New
2. Runnable
3. Running
4. Non-Runnable (Blocked)
5. Terminated



1) New

The thread is in new state if you create an instance of Thread class but before the invocation of start() method.

2) Runnable

The thread is in runnable state after invocation of start() method, but the thread scheduler has not selected it to be the running thread.

3) Running

The thread is in running state if the thread scheduler has selected it.

4) Non-Runnable (Blocked)

This is the state when the thread is still alive, but is currently not eligible to run.

5) Terminated

A thread is in terminated or dead state when its run() method exits.

7Q) Explain Thread operations (or) Explain Thread Class Methods (or) a) Creating thread b) Running thread c) Terminating Thread

Ans)

Creating Threads

A thread can programmatically be created by two ways.

1. By extending Thread class

```
Example:    class Multi extends Thread
            {
            public void run()
            {
                System.out.println("thread is running...");
            }
```

2. By implementing Runnable interface.

```
Example:    class Multi3 implements Runnable
            {
            public void run()
            {
                System.out.println("thread is running...");
            }
```

Running thread

The run() method of thread class is called if the thread was constructed using a separate Runnable object otherwise this method does nothing and returns. When the run() method calls, the code specified in the run() method is executed. You can call the run() method multiple times.

Syntax : userThread.run();

The run() method can be called using the start() method or by calling the run() method itself.

Syntax : userThread.start();

Terminating Thread

Whenever we want to stop a thread from running state by calling stop() method of Thread class in Java. This method stops the execution of a running thread and removes it from the waiting threads pool and garbage collected. A thread will also move to the dead state automatically when it reaches the end of its method. The stop() method is deprecated in Java due to thread-safety issues.

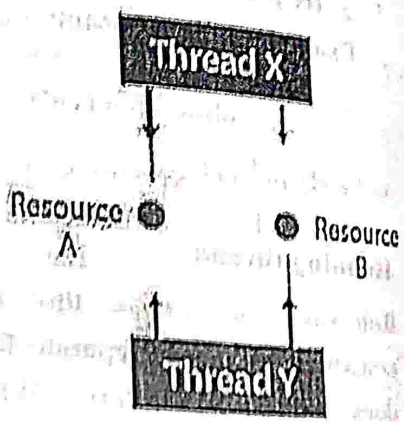
Syntax : userThread.stop();

Example : Lab Program

Note : Single Tasking Using a Thread, Multi Tasking Using Threads, Multiple Threads Acting on Single Object : For these topics write LAB Program

8Q) Explain Deadlock of Threads.

Ans) Deadlock in Java is a condition when two or more threads try to access the same resources at the same time. Then these threads can never access the resource and eventually go into the waiting state forever.



Deadlocks cannot be completely resolved. But we can avoid them by following basic rules mentioned below:

Avoid Nested Locks: We must avoid giving locks to multiple threads, this is the main reason for a deadlock condition. It normally happens when you give locks to multiple threads.

Avoid Unnecessary Locks: The locks should be given to the important threads. Giving locks to the unnecessary threads that cause the deadlock condition.

Using Thread Join: A deadlock usually happens when one thread is waiting for the other to finish. In this case, we can use join with a maximum time that a thread will take.

9Q) Explain Thread Priorities.

Ans) A thread scheduler assigns threads to specific processes based on their priority. A java thread comes with a pre-assigned priority. In addition to this, the java virtual machine can also assign priority to threads or explicitly given by the programmers. The value of thread priority is between 1 and 10 (inclusive).

3 constants defined in Thread class:

1. public static int MIN_PRIORITY
2. public static int NORM_PRIORITY
3. public static int MAX_PRIORITY

Default priority of a thread is 5 (NORM_PRIORITY). The value of MIN_PRIORITY is 1 and the value of MAX_PRIORITY is 10.

The 'getPriority()' method in Java helps in returning the priority of the thread bound as value to it.

The 'setPriority()' method changes the priority value of a given thread. It throws the IllegalArgumentException when the thread priority is less than 1 or greater than 10.

10) Explain Daemon thread in Java.

Ans) Daemon thread in Java is a service provider thread that provides services to the user thread. When all the user threads die, JVM terminates this thread automatically.

There are many java daemon threads running automatically e.g. gc, finalizer etc.

Daemon threads are useful for background supporting tasks such as garbage collection, releasing memory of unused objects and removing unwanted entries from the cache. Most of the JVM threads are daemon threads.

Creating a Daemon Thread:

```
NewThread daemonThread = new NewThread();
daemonThread.setDaemon(true);
daemonThread.start();
```


UNIT – 5

1Q) Explain Applet Concepts.

Ans) An applet is a Java program that runs in a Java-compatible browser such as Internet explorer. This feature allows users to display graphics and to run programs over the Internet. An applet allows web documents to be both animated and interactive.

There are many advantages of applet. They are as follows:

- It works at client side so less response time.
- Secured
- It can be executed by browsers running under many platforms, including Linux, Windows, Mac Os etc.

Creating an Applet: To create an applet, our program must import the Applet class. This class is found in the java.applet package. The Applet class contains code that works with a browser to create a display area.

Syntax : public class AppletClassName extends Applet

Example: public class WishUser extends Applet

<APPLET> tag : Applet tag <applet> is required to load and start an applet either in a browser window or in an appletviewer(provided by JDK).

Three attributes are required in the <APPLET> tag. Two of these attributes, width and height, specify the space the applet occupies on the screen. The third required attribute is code. The code attribute specifies the class file from which the applet is loaded.

Example:

```
<applet code="WishUser.class" width=300 height=300>
```

Note: At present, many current version of browsers such a Google Chrome and Mozilla Firefox(except Internet Explorer) have stopped displaying applets or recognizing <applet> tag, hence the only reliable tool to display an applet is appletviewer.

Applet Parameters

Parameters specify extra information that can be passed to an applet from the HTML page. Parameters are specified using the HTML's param tag.

The <param> tag is a sub tag of the <applet> tag. The <param> tag contains two attributes: name and value which are used to specify the name of the parameter and the value of the parameter respectively.

<param name="msg" value="Welcome to applet">

these two parameters can be accessed in the applet program using the getParameter() method of the Applet class.

String str=getParameter("msg");

A Simple Applet : Lab Program

2Q) Explain Applet with Swing Components

Ans)A Swing applet extends JApplet rather than Applet. JApplet is derived from Applet. Thus, JApplet includes all of the functionality found in Applet and adds support for Swing.

It is used for creating window based applications. It includes components like button, scroll bar, text field etc. Putting together all these components makes a graphical user interface.

```
import javax.swing.*;
import java.applet.*;
import java.awt.*;
import java.awt.event.*;
public class SApplet extends JApplet implements ActionListener {
```



```
TextField input,output;
Label label1,label2;
Button b1;
JLabel lbl;
int num, sum = 0;
public void init() {
  label1 = new Label("please enter number : ");
  add(label1);
  label1.setBackground(Color.yellow);
  label1.setForeground(Color.magenta);
  input = new TextField(5);
  add(input);
  label2 = new Label("Sum : ");
  add(label2);
  label2.setBackground(Color.yellow);
  label2.setForeground(Color.magenta);
  output = new TextField(20);
  add(output);
  b1 = new Button("Add");
  add(b1);
  b1.addActionListener(this);
  lbl = new JLabel("Swing Applet Example. ");
  add(lbl);
  setBackground(Color.yellow);
 }
 public void actionPerformed(ActionEvent ae ){
   try {
     num = Integer.parseInt(input.getText());
     sum = sum+num;
```

```
        input.setText("");
        output.setText(Integer.toString(sum));
        lbl.setForeground(Color.blue);
        lbl.setText("Output of the second Text Box : " +
output.getText());
      } catch(NumberFormatException e) {
        lbl.setForeground(Color.red);
        lbl.setText("Invalid Entry!");
      }
    }
}
```

3Q) Explain Animation in Applet

Ans) Applet is mostly used in games and animation. For this purpose image is required to be moved.

DisplayImage.java

```
import java.awt.*;
import java.applet.*;
public class DisplayImage extends Applet {
 Image picture;
 public void init() {
  picture =getImage(getDocumentBase(),"bike_1.gif");
 }
  public void paint(Graphics g) {
 for(int i=0;i<500;i++){
  g.drawImage(picture, i,30, this);
 try{Thread.sleep(100);}catch(Exception e){}
 }
 }
}
```


myapplet.html

```
<html>
<body>
<applet code="DisplayImage.class" width="300" height="300">
</applet>
</body>
</html>
```

4Q) Define JDBC? Explain Stages in a JDBC Program

Ans) JDBC (Java Database Connectivity) is an API(Application programming interface) used in java programming to interact with databases. The classes and interfaces of JDBC allow the application to send requests made by users to the specified database.

Stages in a JDBC Program

There are 5 steps to connect any java application with the database using JDBC. These steps are as follows:

1. Register the Driver class
2. Create connection
3. Create statement
4. Execute queries
5. Close connection

1) Register the driver class :-

The forName() method of Class class is used to register the driver class. This method is used to dynamically load the driver class.

```
Class.forName("oracle.jdbc.driver.OracleDriver");
```

2) Create the connection object

The getConnection() method of DriverManager class is used to establish connection with the database.

```
Connection con=DriverManager.getConnection(
"jdbc:oracle:thin:@localhost:1521:xe","system","password");
```

3) Create the Statement object

The createStatement() method of Connection interface is used to create statement. The object of statement is responsible to execute queries with the database.

```
Statement stmt=con.createStatement();
```

4) Execute the query

The executeQuery() method of Statement interface is used to execute queries to the database. This method returns the object of ResultSet that can be used to get all the records of a table.

```
ResultSet rs=stmt.executeQuery("select * from emp");
```

5) Close the connection object

By closing connection object statement and ResultSet will be closed automatically. The close() method of Connection interface is used to close the connection.

```
con.close();
```

5) Explain a) Working with Oracle Database, b)Working with MySQL Database

Ans) Working with Oracle Database

To connect java application with the oracle database, we need to follow the following steps. In this example, we are using Oracle 10g as the database. So we need to know following information for the oracle database:

1) Driver class: The driver class for the oracle database is oracle.jdbc.driver.OracleDriver.
2) Connection URL: The connection URL for the oracle10G database is jdbc:oracle:thin:@localhost:1521:xe where jdbc is the API, oracle is the database, thin is the driver, localhost is the server name on which oracle is running, we may also use IP address, 1521 is the port number and XE is the Oracle service name. You may get all these information from the tnsnames.ora file.

3) Username: The default username for the oracle database is system.
4) Password: It is the password given by the user at the time of installing the oracle database.

Working with MySQL Database

To connect Java application with the MySQL database, we need to follow the following steps.

In this example we are using MySql as the database. So we need to know following informations for the mysql database:

1) Driver class: The driver class for the mysql database is com.mysql.jdbc.Driver.
2) Connection URL: The connection URL for the mysql database is jdbc:mysql://localhost:3306/deptes where jdbc is the API, mysql is the database, localhost is the server name on which mysql is running, we may also use IP address, 3306 is the port number and deptes is the database name. We may use any database, in such case, we need to replace the deptes with our database name.
3) Username: The default username for the mysql database is root.
4) Password: It is the password given by the user at the time of installing the mysql database. In this example, we are going to use root as the password.

SQL:- working with MySQL databases can be both rewarding & challenging but with the right knowledge & tools, it becomes a powerful asset for storing, & retrieving, & managing data.

Question Bank

UNIT – I

1. Explain Java features.
2. Explain Data Types.
3. Explain Operators.
4. Explain looping statements
5. Define Array? Explain Types of arrays

UNIT – II

6. Explain Strings.
7. Explain Constructors.
8. Explain Features of OOPS.
9. Define Inheritance? Explain.

UNIT – III

10. Explain Polymorphism.
11. Define interfaces? Explain
12. Explain about packages
13. Define Exception? Explain

UNIT – IV

14. Explain Lifecycle of a Thread.
15. Write about FileOutputStream and FileInputStream.

UNIT – V

16. Define Applet ?Explain.
17. Define JDBC? Explain Stages in a JDBC Program.
18. Write about Working with MySQL Database