ANNEXURE - I

ANNA UNIVERSITY
CHENNAI - 25

College Code	4	1	0	8								
College Name	G.K.M. college of Engineering Technology.											
Register Number	4	0	7	1	0	1	0	5	0	0	9	
Name of the Candidate	Dinesh T											
Degree	B.E/EEE											
Branch	Electrical and Electronics Engineering							Semester	5th Sem.			
Question Paper Code	X	6	0	3	8	6						
Subject Code	C	S	2	3	1	1						
Subject Name	Object Oriented Programming.											
Date	17 (DD)	7 (MM)	2021 (YY)	Session	FN	AN ✓						
No. of Pages used	28	In words	Twenty eight pages only.									
All particulars given above by me are verified and found to be correct												
Signature of the Student with date	Dinesh / 17/7/2021.											

For Office Use Only

Instructions to the Candidate: Put Tick mark (✓) for the questions attended in the tick mark column against each question

PART – A			PART – B & C									
Question No.	✓	Marks	Question No.		(i) ✓	(i) Marks	(ii) ✓	(ii) Marks	(iii) ✓	(iii) Marks	Grand Total (in words)
1	✓		11	a	✓						
2	✓			b							
3	✓		12	a	✓		✓				
4	✓			b							
5	✓		13	a	✓		✓				
6	✓			b							
7	✓		14	a							
8	✓			b	✓		✓				
9	✓		15	a	✓						
10	✓			b							Grand Total
			16	a							
				b							
Total											

Declaration by the Examiner: Verified that all the questions attended by the student are valued and the total is found to be correct

Date	Name of the Examiner	Signature of the Examiner

PART - B

11)

a) Object Oriented Programming:-

*) Object
*) classes
*) Data Abstraction
*) Data encapsulation.
*) Inheritance.
*) polymorphism.
*) Dynamic binding
*) Message Passing.

*) Object:-

*) Objects are the basic run time entities in an object orient System.

*) They may represent a person a place, a bank account a table of data or any them that the program has to handle.

*) They may also represent user defined data such as vectors, time and lists.

*) programming problem is analysed in terms of objects and the nature of communication between them.

*) Program objects should be chosen such that they match closely with real world objects

*) when a program is executed the objects interact by sending messages to one another.

*) Each object contains data and code to manipulate data.

*) For Example, class name object name.

Classes:-

*) The class is a collection of variable declration and method definitions.

*) The class contains that data and code of an object can be made a user defined data type.

*) Once a class has been defined we can create any number of objects to that class.

*) Each object is associated with the data of type class which they are created.

*) class is a collection of similar objects

*) For eg):

```
class name of the class
{
Variable declarations;
Methods definitions;
}.
```

Data. ~~class~~ obstraction:

*) The abstraction means provide essential things and hide non-essential things

*) classes use the concept of abstraction and are defined as a list abstraction attributtes such as sing. weight, cost and function to operate on those attributes.

*) The encapsulate all the essential properties of the objects that are to be created.

*) The classes use the concept of data abstraction they are known as abstraction data types.

Data encapsulation:-

*) The wrapping up of data and function into single unit is called encapsulation.

*) Data encapsulation is the most striking feature of a class.

*) These functions provide the interface between objects data and the program

*) The insulation of the data from direct access by the program is called data hiding.

*) They encapsulate all the essential properties of the objects that are to be created.

Inheritance:-

*) The inheritance means derived a new class from the existing one.

*) It is the process by which objects of one class acquire the properties of objects of another class.

*) It supports the concepts of hierarchical classification

*) The concept of inheritance provides the idea of reusability

*) The new class will have the combined features. (or) both the classes

Polymorphism:-

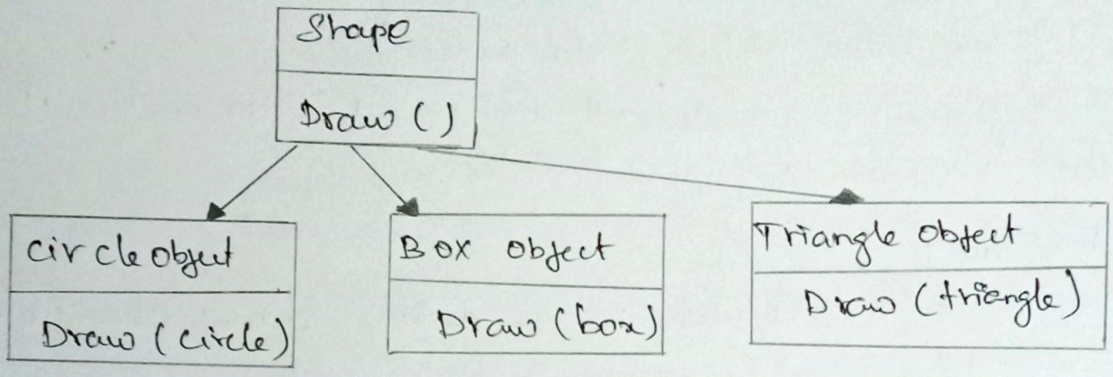
*) The polymorphism means the ability to take more than one form.

*) An operation may exhibit different behaviors instances. The behavior depends upon the types of data used in the operation.

*) The process of making an operator to exhibit different behaviors in different instances is known as operator overloading.

*) A single function name can be used to handle different number and different types of arguments.

*) the polymorphism is extensively used in implementing inheritance.



Dynamic binding!-

*) Dynamic binding means the code associated with a given procedure call is not known until the time of the call at run time.

*) It is associated with polymorphism and inheritance.

*) Binding consist of two types. they are dynamic Binding and static binding.

*) The static binding means compile time process and the dynamic binding means run time process.

Message passing!-

*) As object oriented program consist of a set of objects that communicate was each other.

*) The process of programing in an object oriented language, therefore involves the following basic types.

*) Creating Classes that define objects and their behavior.

*) Creating classes that define objects and their behavior

*) Creating objects from class definitions.

*) Establish communication among objects

*) A message of an object is required for execution of a procedure and therefore will involve a functions in the receiving object that generate the desired result.

The advantages of object oriented methodology:

*) The principle of data hiding helps the programer to build secure program that cannot be invaded by code in other parts of the program.

*) It is possible to have multiple instant of an objects to co-exist without any interfaces.

*) object oriented system can be easily upgraded from small to large system.

*) software complexity can be easily manages

*) It is possible to map objects in the problem domain to those in the program.

*) It is easy to partition the work in a project based on Objects.

*) Through inheritance, we can eliminate redundant code and extended the use of existing classes.

*) It is possible to save development and their higher productivity.

12) a) (i) C++ program using operator overloading to adding two time values :-

```
#include <iostream.h>
#include <conio.h>
class time
{
public:
time()
{
public:
hr=0;
min=0;
sec=0;
}
int hr, min, sec;
void read()
{
cout<<"Hours=";
cin>>hr;
cout<<"\n Minutes=";
cin>>min;
cout<<"\n seconds=";
cin>>sec;
}
time operator +(time t2)
{
time t3;
t3.sec=sec+t2.sec;
if (t3.sec>60)
{
```

```
t2.mint=1;
t3.sec-=60;
}
t3.min=min+t2.min;
If (t3.min>60)
{
t2.hr+=1;
t3.min-60=60;
}
t3.hr=hr+t2.hr;
roturn t3;
}
Void display()
Pf (hr<10)
{
cout<<"0"<<hr;
}
else
  cout<<hr;
if (min<10)
{
cout<<":0"<<min;
}
else
  cout<<":"<<min;
Pf (sec<10)
{
cout<<":0"<<sec;
}.
```

```
else.
 count<<":"<<sec;
};
void main ()
{
clrscr ();
time c1,c2,c3;
cout <<"\n\n Enter the First Time\n\n";
c1.read();
cout<<"\n\n Enter the second Time\n\n";
c2. read();
c3= c1 + c2;
cout <<" \n \n First Time \t\t ";
c1. display();
cout <<" \n \n Second Time \t\t";
c2.display();
cout <<" \n \n After addition, the time is ";
c3. display ();
getch ();
}
```

12)
a)
(ii) Friend function in C++ :-

The concept of encapsulation and data hiding dictate that non-member function should not be allowed to access an objects private and protected members. (Not protected member's will be seen in Inheritance).

Imagine that the user wants a function to operate on objects of two different classes. At such times, it is required to allow functions outside a class to access and manipulate the private members of the class.

In C++ this is achived by using the concept of friends!

A function declaration must be prefixed by the key word friend whereas the function definition must not.

Using friend function (or) friend class a non-member function can able to access private data members.

A Function can be a friend to multiples classes.

A Friend function possesses the following special characteristics.

*) The scope of a friend function is not limited to the class in which it has been declared as a friend.

*) A friend function cannot be called using the object of that class, it is not in the scope of the class. It can be invoked like a normal function without the use of any object.

*) Unlike class members function, it cannot access the class members directly. However it can use the object and the dot operator with each member name to access both the private and public members.

*) It can be declared in the private part (or) the public part of a class without affecting its meaning.

```
//friend function.
#include<iostream.h>
class sample
{
 int a,b;
 public;
 void setdata() {a=10,b=20;}
 friend float mean(Sample s);
 };
 float mean (Sample s)
 {
 return float (s.a+s.b)/2.0;
 }
 int main()
 {
 Sample x;
 x.setdata();
 cout<<"Mean value ="<<mean(x)<<"\n";
 return();
 }
```

Output:-

Mean value = 15.0

Consider a situation of operating on objects of two different classes. Such a situation friend function can be used to bridge the two classes.

// Normal function accessing object's private members

```
#Include<Iostream.h>
class two; // advance declaration like function
                                    prototype.
class one
{
int data1;
Public:
 void setdata (int ,int it)
 {
 data1 = init;
}
friend int add: both (one a, two b);
};
class two
{
int data2;
Public:
void setdata (int init)
{
data2 = init;
}
friend int add: both (one a, twob);
};
Int add-both (one a, twob);
{
return a.data1 + b. data2;
}
```

```
void main()
{
one a;
two b;
a.setdata(5);
b.setdata(10);
cout<<"Sum of one and two"<<add:both(a,b);
```

Output:-

Sum of one and two:15

Friend function are useful in the following situation:-

*) Function operating on objects two different classes. This is the ideal situation where the friend function can be used to bridge two classes.

*) Friend function can be used to increase the versatility of overloaded operators

*) Some time, a friend allow a more obvious syntax for calling a function, rather than what a member function can do.

13) I O Streams used to file operation:-

a)

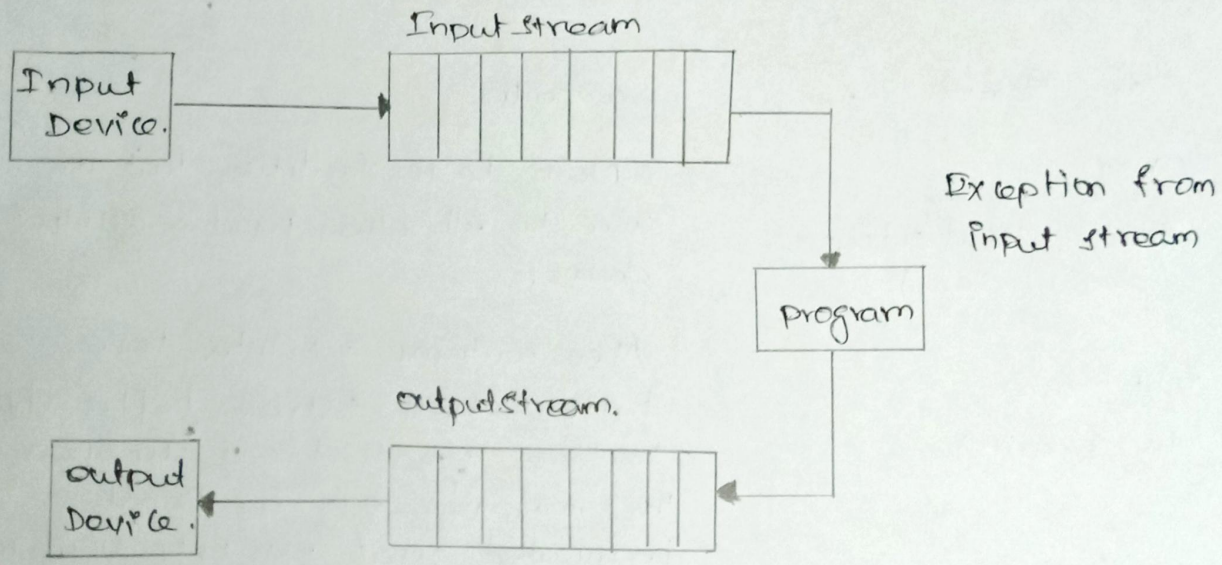
(i) The Input/output system in C++ is designed to work with a wide variety of devices including terminals disk and tape drives.

Although each devices is very different, the Input/output system supplies an interface to the programmer that is independent of the actual device being accessed.

This interface is known as stream.

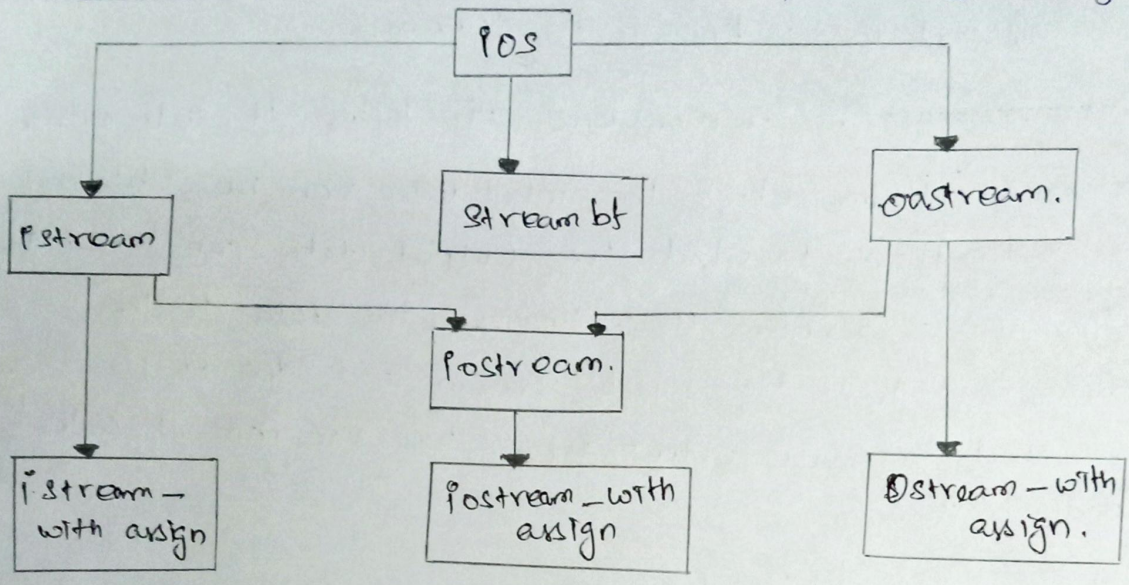
A stream is a sequence of bytes. It acts either as a source from which the input data can be obtained or as destination to which the output data can be sent.

The source stream that provides the data to the program is called the input stream and the destination stream that recives output from the program is called the output stream.



C++ contains several predefined streams that automatically opened when a program begins it's execution (eg. cin, cout)

The classes are declaimed in the header file iostream.



class Name.	contents:
*) ios. (General input/output stream class)	contain basic facilities that are used by all other input and output classes. Also contains a pointer to a buffer object (stream buffer object) Declares constant and functions that are necessary for handling terminated input and output operation Inherits the properties of ios
*) istream (Input Stream)	
iostream.	contains overload insertion operator "<<"; inherits the properties of ios istream and ostream through through multiple inheritance and thus contain all the input and output functions.
streambuf	provides an interface to physical devices through buffers. Acts as a base for filebuf class ios -files

13)
a) C++ Program for create a file with odd numbers and
ii) Create another file with set of even numbers and merge these two files and storing in another file.:

Program:-

```
# include <iostream.h>
# include <conio.h>
# include <fstream.h>
void main ()
{
fstream f1, f2;
int number;
clrscr ();
Printf ("contents of DATA file \n\n");
f1. fopen ("DATA"); /* Create a data file*/.
for (i = j ; <= 30; i++)
{
cin >> number;
if (number ==-1) break;
}
fclose (f1);

f1. fopen ("DATA");
f2. fopen ("ODD");
f3. fopen ("EVEN");
while ((number = getw (f1)) != EOF) /* Read from
                                    Data file*/.
{
```

```
if (number % 2 == 0)
putw (number, f2);
else
putw (number, f2);
}
fclose(f1);
fclose (f2);
fclose (f3);
f2.fopen ("ODD");
f3.fopen ("EVEN");
Cout<<"\n\n Contents of ODD file \n\n");
while (number = getw (f2)) != EOF)
Printf ("%4d", number);
Printf ("\n\n Contents of EVEN file \n\n");
while ((number = getw (f3)) != EOF)
Printf (" %4d ", number);
fclose (f2);
fclose (f3);
getch ();
}.
```

14)
b)
(7)

Packages in JAVA :-

The JAVA Standard library (or API) includes hundreds of classes and method grouped into several functional packages.

Language support package :-

A collection of classes and methods required for implementing basic features of JAVA.

Utilities Packages :-

A collection of classes to provide utility functions such as date and time functions.

Input/output package :-

A collection of classes required for input/output manipulations

Networking Package :-

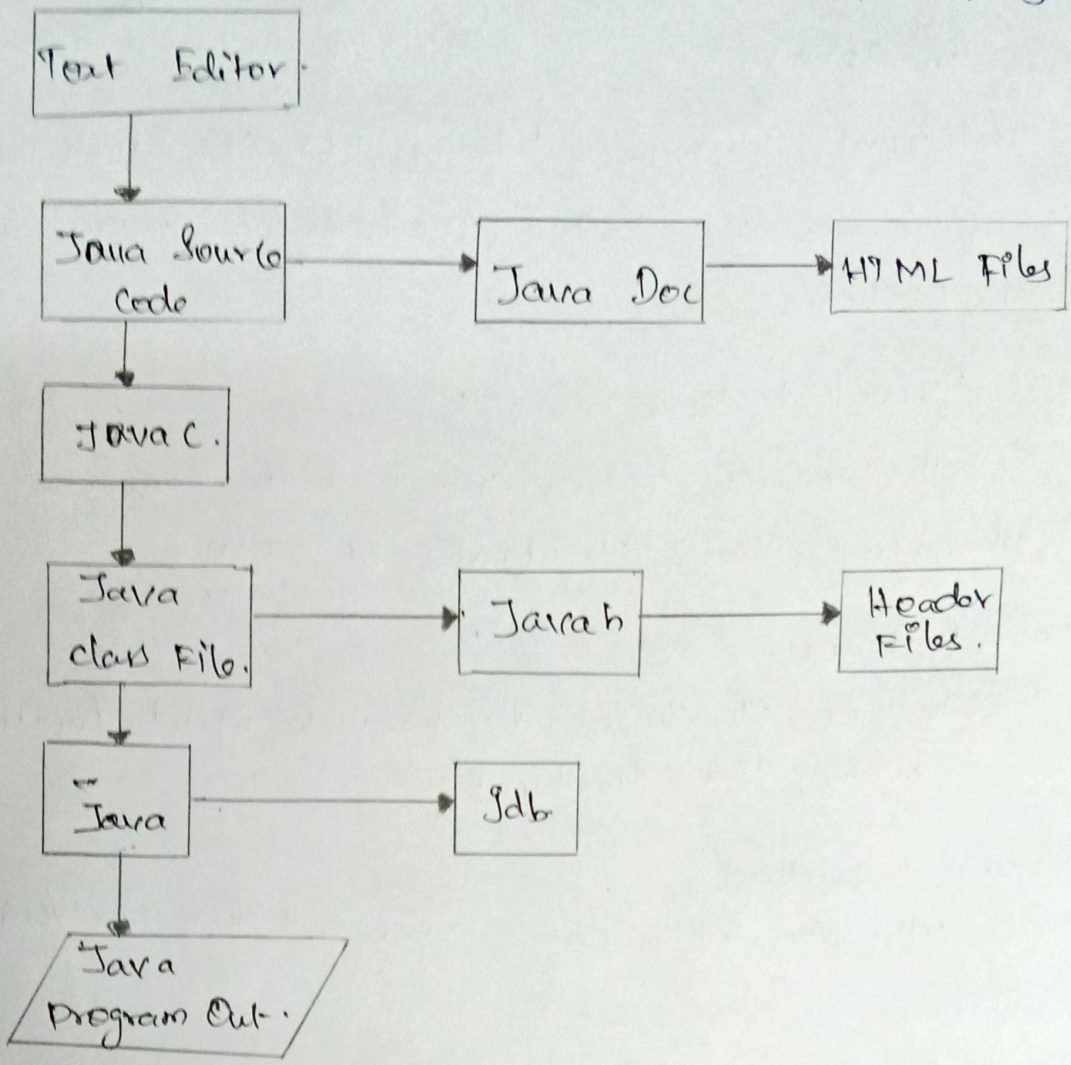
A collection of classes for communicating with other computers via Internet..

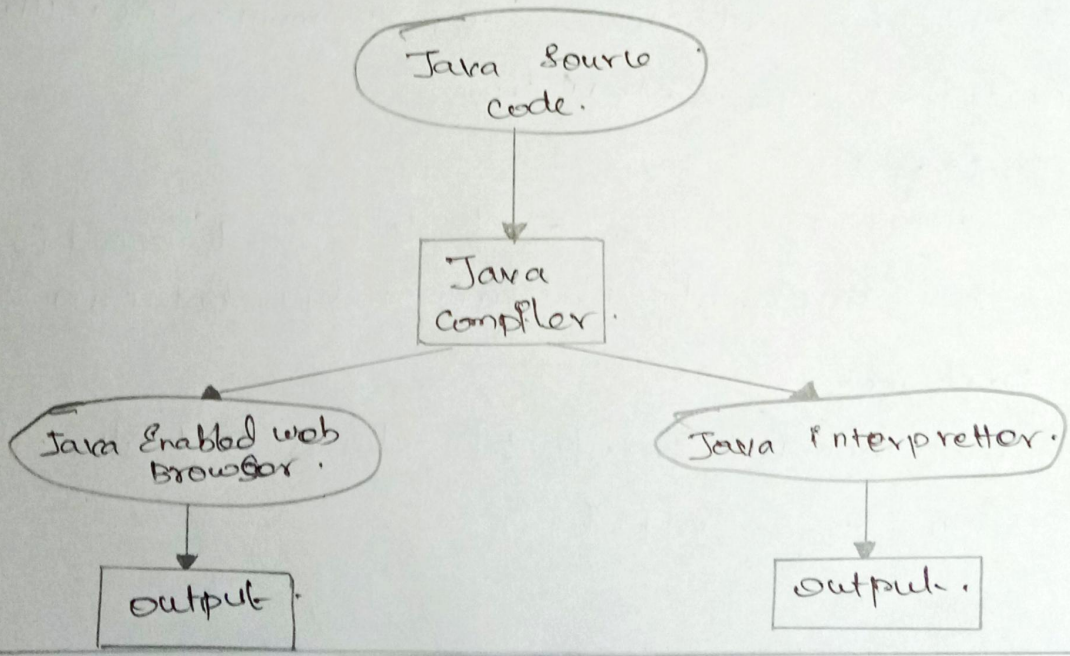
AWT Package :-

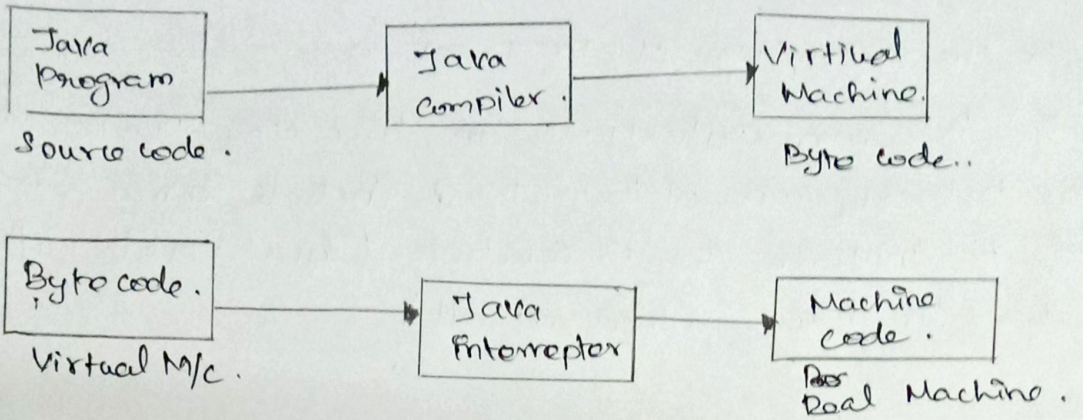
The abstract window toolkit package contains classes that implements platform-independent GUI.

Applet Package :-

This included a set of classes that allows us to create Java applets.


Text Editor
Java Source code
Java Doc
HTML Files
Java C.
Java class File.
Javah
Header Files.
Java
Jdb
Java Program Out.


Java Source code.
Java compiler
Java Enabled web Browser.
Java interpretter.
output
output.



Simple Java Program:-

```
class Sampleone.
{
Public static void main (String args[])
{
System.out.println ("Java is better than C++");
}
}
```

Class Declaration:-

Java is a true object-oriented language and. therefore every thing must be place inside a class.

Class Sampleone.

Class name.

The main line:-

Public static void main (string args [])

define a method named main.

*) Java application program must include the main () method.

*) This is staring point for the Interpreter to begin to the execution of the program

*) A Java application can have any number of classes but only one of them must include a main method to initiate the execution (Java applets will not use the main method at all)

Public :

It is an access specifier that declares the main method as unprotected and therefore making its accesible to all other classes.

Static :-

which declares this methodo as one that belongs to the entire class and not a part of any objects are class.

Void :-

It states that the main method does not return any value.

All Parameters to a method are declared inside a pair of parenthesis Here, strings args [ ]. declares a parameter named args which contains an arry of objects of the class type string.

eg: Pgm 2.

```
Import Java.lang.math;
Class Squareroot
{.
```

```
Public Static Void main (String args[])
{
double x = 5;
double y;
y = math.Sqrt (x);
System.Out.println("y = "+y);
}
}
```

y = math Sqr (x); Invokes the method Sqrt of the math class; computer square root of x and then assign the result to the variable y;

In the output statement + acts as a concatenation operator of two strings. the value of y is converted. In to a string representation before concatenation.

eg: Pgm 3 with multiple classes

```
class Room
{
float length, breadth;
void getdata (float a, float b)
{
length = a;
breath = b;
}
}
class Room Arog
{
public static Void main (String args[])
{
float area;
```

```
Room room = new Room(); // Creates an object room'
room1.getdata(14,10);
area = room1.length * room1.breadth;
System.out.println("Area = "+area);
}
}
```

15) a) Interface and Inner classes.

The Implementation of particular class can be made fully abstract by creating an interface for. It That meaning using interface we can specify what a class must do but not how it does.

Interface are similar to the classes but their methods do not have any defination body. Thus interface are used to hide the implementation details.

Interface can defined using following syntax:-

```
access_modifier interface name_of_interface.
{
return_type method_name1(parameter1, parameter2, ....
                                    Parameterm);
.....
return_type method_name1 (parameter1.parameter2, --parameterd);
type static final variable_name = value;
.....
}
```

The access_modifier specifies the whether the interface is public or not. If the access specifier is not specified for an interface will be accessible to all.

But if the interface is declared as public then it will be accessable to any of the classes the methods declared with in the interface have no bodies. It is excepted to any methods must be defined. with the class definition.

The variable can be assigned with some values with in the interface. they are implicity final and static. Even if you dont not specify the variables as final and static they are assumed to be final and static.

Let us now learn how to write an interface for a class.

Step1 : write following code and save it as my_interface.java.

```
Public Interface my_interface.
{
Void my_method (int val);
}
```

Step:-2 write following code in another file and save it by using interfaceDemo.java

Java Program[InterfaceDemo.java]

```
class A implements my_interface.
{
Public void my_method (int i)
{
System.out.println("\n the value in the class A:" + i);
}
Public void another_method()//defining another method not declared in interface.
```

```
{
System.out.println("\n This is another method in class A");
}
}
class InterfaceDemo
{
Public static void main (String args[])
{
my Interface obj = new A();
// or A obj = new A() is also allowed.
A obj1 = new A();
obj.my_method(100);
obj1.another_method();
}
}
```

Step 3: Compile the program created in step 2 and get the following output.

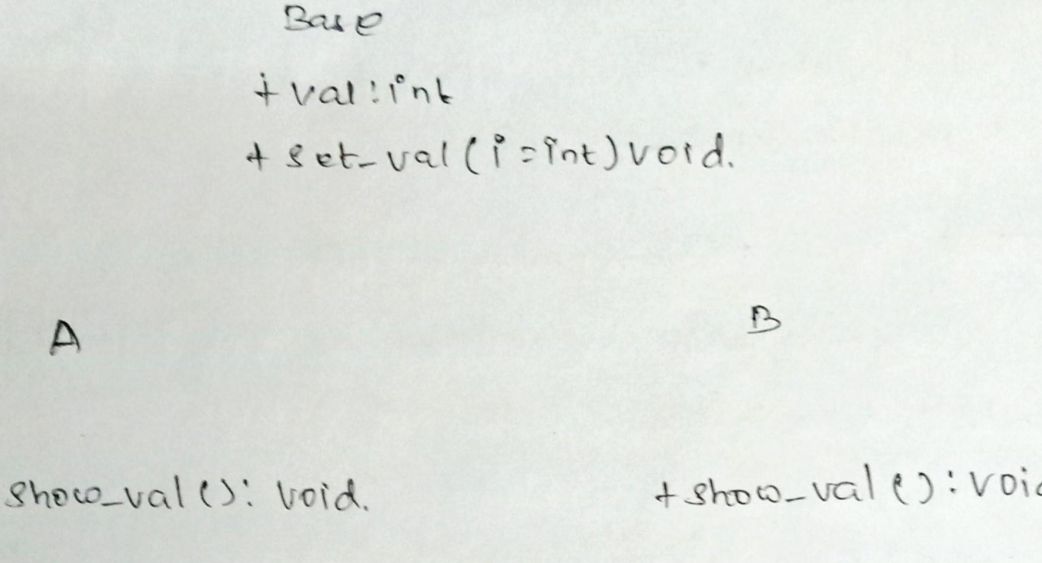
```
F:\test > Java InterfaceDemo.Java.
F:\test > Java InterfaceDemo.
```

The value in the class A: 100
This is another method in class A.

In above program the interface my-interface declares only one method ie. my-method. This method can be defined by class A. There is another method which is defined in class A and that is another_method.

Multiple Inheritance.

* Multiple inheritance is a kind of inheritance in which more than one classes are derived from the same basis.



Mutiple. inheritance is not possible in Java. But still it can be achived with the help of interface.

PART-A

(1) Atribute:-

Atribute define the characteristics of a class

*) The set of values of an attribute of aparticular Object is called it's state.

*) In class program attribute can be a string (or) it can be integer.

(2) Compile time polymorphism in oops:-

Polymorphysism is defined as one interface to controll acess to a general class of action.

There are types of polymorphism.

(i) compail time polymorphism

(ii) run time polymorphism.

(i) compail time polymorphism is function and operatore overloading

when ever an object is found with the functionality at compile time this is known as compile time polymorphism

(3) Data abtraction:-.

abstraction is the concept of object oriented programing that shows only essential atributes, and hide unnecessary deatils.

The main purpose of abstraction is hiding the unnecessary deatils from user.

It sleeting data from a large pad to show only relevent deatils of the object to the user.

407105009 Dinesh T CS2311

(5) Use of Try, Throw and catch statements:

The try statement allows you to define a block of code to be tested for errors while it is being executed.

The throw keyword throw an exception when a problem is detected, which lets us create a custom error.

The catch statement allows yo to define a block of code to be executed. if an error occurs in the try block.

(6)

(7) Function for file handling:-

1.	fopen()	open new or existing file.
2.	fprintf()	write data into the file.
3	fscanf()	reads data from the file.
4	fputc()	writes a character into the file.
5	fgetc()	reads a character from file.
6.	fclose()	close the file.

(7) Java is robust because:-

It uses strong memory mangement

There is ~~memory~~ lack of pointers that avoid Security problem.

There are exception handling and types checking mechanism in Java. All these point make Java robust.

(8) Ex is a file extension for executable file format.

An executable is a file that contain a program that is a particular kind of file that is capable of executed as a program in the computer.

(9) An exception in an event which occurs during the execution of a program, that disrupts the normal flow of the program instructions.

10) Multithreading is a Java feature that allows concurrent of two or more parts of a program for maximum utilization of CPU. Each part of such program is called a thread.

i) Extending the thread class
2) Implementing the Runnable interface.

(11)

```
#include <iostream>
using namespace std;
struct A
{
virtual void f() { cout << "Class A" << endl; }
};
structure B: A {
 virtual void f() { cout << "Class B" << endl; }
};
```