1. An Introduction to Java.

* object oriented programming concepts. (OOP)
That can be use as templates for creating copies of such modules on demand is known as object oriented programming (OOP)

- Features of OOP.
1) OOP emphis on data rather than procedure.
2) OOP follows bottom-up approach in program design.

* Introduction to Java.
Java programming language was initated by James gosling he is called futher of Java Programming.

* Features of Java.
1) simple = Java is simple language.
2) object-oriented = Java is object-oriented language like C++, Python.
3) Platform independent = Java language is platform independent language.
4) Interpreated = Java is an interpreated language programs run directly from the source code.
5) Distributed = Java facilites the building of distributed application.
6) Dynamic = Java language is capble of dynamically linking in new class libraries.
7) Robust = Java is robust language because it provides many safeguards to ensure reliable code.
8) secure = Java is aimed to be used in networked or distributed environments.

Tools, Array Program.

JIT = Just - In - Time.
CGI = Common gateway interface.
GUI = Graphical user interface.

* Java Development tools.

1) Java c = The java compiler that translets javacode to bytecode file which java interpreter can understand.

2) Java = Java is a Java application launcher. it launches a java application by starting Java runtime environment, loading class and invoking that class's main method to run the program.

3) appletviewer = Enables us to run java applets. normally applets are the programs those can be embedded in a web page and run in web browser.

4) javadoc = Creates HTML formatted documentation from java source code files.

5) jdb = Java debugger that heles us in finding errors in Java programs.

6) jar = Java achieve file, creates a zip / jar file.

7) javah = produces header files for use with native methods.

8) Javap = Java disassembler which enables us to convert bytecode files into a program description.

* Java environment:

1) JDK (Java development kit)
The full package needed to develop Java programs.

2) JRE (Java Runtime environment)

3) Jvm (Java virtual machine)

* Structure of java program.

1) Documentation.
2) package statement.
3) Import statement.
4) class definition.
5) main method.
6) variables and statements inside methods.

* Basic concepts in java.

- Comments and data type.

Single-line comment
multi-line comment
Documentation Comment

- Tokens.

A token is the smallest individual unit in a java program that the compiler understand.

- Variables.

A variable is a named location used to store a value that can change during program execution.

* Control flow statement.

* program for score of variables.

```
Public class sample.
{
string name;
int age;
static int count;
void accept (string n, int a)
{
name = n;
age = a;
}
void show()
{
Count ++;
system.out.println ("count=" + count);
}
void display()
{
system.out.println ("Name =" + name);
system.out.println (" Age =" + age);
}
Public static void main (string args[])
{
sample s1 = new sample();
sample s2 = new sample();
s1.accept ("abc", 10);
s1.display();
s1.show();
s2.accept ("xyz", 20);
```

```
s2.display();
s2.show();
}
}
```

* Final variable.

* Display for loop 1 to 10. using Java.

```
public class forDemo
{
public static void main(string[] args)
{
for(int i=1; i<=10; i++)
{
system.out.print("count is :"+i+" ");
}
}
}
```

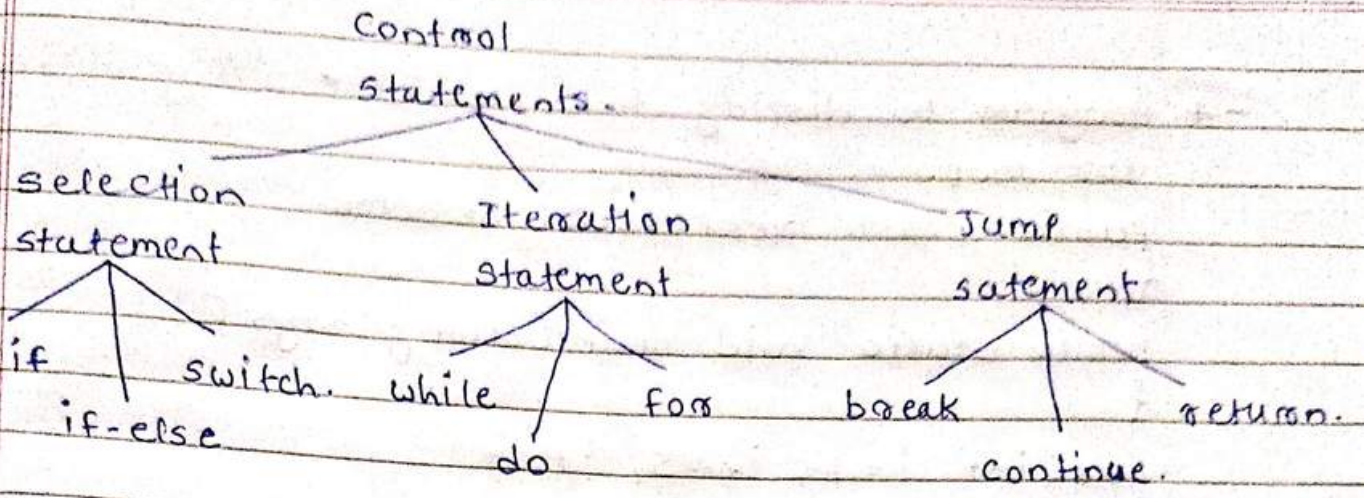
* Final variable.

The final variable ~~as~~ also called constant variable.

This is accomplished using a special type of variable known as final variable.

* Control flow statements.

statements that control the flow of the program are called control flow statement.



* Selection / Decision making statements.
1) If statement
2) If else statement.
3) else if statement.
4) switch statement.

- we can use switch statement inside other switch stament in java. it is known as nested switch statement.

* Loops.
The processes of repeatedly performing tasks is known as looping. there is either entry controlled loop or exit controlled loop.

1) while loop.
2) do-while loop.
3) for loop.
4) for each loop.

* ~~Ne~~ Nested loops.

- Program to display triangle of * using nested for loop.

```
public class NestedForLoopDemo
{
Public static void main(string args[])
{
for(int i=1; i<=3; i++)
{
for(int j=1; j<=i; j++)
{
System.out.print("*");
}
system.out.print(" ");
}
}
}
```

- Nasted do while loop.

* Arrays in Java.

* Program for one Dimensional array.

```
public class arraydemo.
{
Public static void main(string args[])
{
int a[]={2,4,6,3,1};
system.out.println("Number of element in array A:"+a.length);
system.out.println("Elements in array A");
for(int i=0; i<a.length; i++)
```

```
system.out.print (a[i] + "\t");
}
}
```

Two Dimensional (2D) Array.

Syntax.

data-type [][] array-name = new int [row-size] [column-size];

Program of 2D array.

```
public class twodim.
{
final ~~sati~~ static int r = 3;
final static int c = 3;
public static void main (string [] args)
{
int ~~ar~~mult [][] = new int [r] [c];
int row, column,
int i, j;
for (i = 1; i <= r; i++)
{
for (j = 1; j <= c; j++)
{
mult [i] [j] = i * j;
system.out.println (" " + mult [i] [j]);
}
}
system.out.println (" ");
}
}
```

* program sort in array.

```
import java.util.Arrays;
Public class sort Int Array example.
{
public static void main (string [] args)
{
int [] i1 = new int [] {3,9,2,5,4,1};
system.out.print ("original int array:");
for (int index = 0; index < i1.length; index++)
system.out.print (" " + i1 [index]);
Array.sort (i1);

system.out.print ("sorted int array:");
for (int index = 0; index < i1.legth; index++)
system.out.print (" " + i1[index]);
int [] i2 = new int [] {5,2,3,1,4};
Arrays.sort (i2, 1, 4);
system.out.print ("pratially sorted int array:");
for (int index = 0; index < i2.length; index++)
system.out.print (" " + i2 [index]);
}
}
```

* Accerting Input:

1) Using buffered ~~Reder~~ Reader Class.
   - Class Declaration of buffer Reader.

- Program for buffer Reader class.

```
import Java.io.*;
Public class Example.Buffered Reader.
{
public static void main (string args []) throw
Exception.
{
Buffered Reader br=new Buffered Reader
(New InputstreamReader (system.in));
string name = " ";
system.out.println("Enter data:");
name=br.readLine ();
system.out.println ("data is:"+name);
}
}
```

2) Using command line arguments.

- Program to display all command line arguments.

```
Public class CommdLineArg.
{
Public static void main (string args [])
{
System.out.println ("Total arguments passed
are" + args.length);
System.out.println ("Each of them are");
```

```
for (int k=0; k < args.length; k++)
system.out.println ("arg[" + k + "]:" + args[k]);
}
}
```

3) using scanner class.

-- program for Java scanner nextInt()

```
import java.util.scanner;
public class sample {
{
public static void main (string [] args)
{
scanner input = new scanner (system.in);
system.out.println ("Enter an integer:");
int number = input.nextInt();
system.out.println ("number using nextInt():" +
number);
input.close();
}
}
```

* programs.

- program of addition of numbers.

```
import java.io.*;
class Jadd
{
public static void main (string s[])
system.out.println (" {
```

```
int a, b, sum;
a = 88;
b = 77;
sum = a+b;
system.out.println("a = "+a);
system.out.println("b = "+b);
system.out.println("sum = "+sum);
}
} end of class
```

- Program to cheack a leap year on not.

```
import java.util.Scanner;
class Evenodd
{
Public static void main (string[] args)
{
scanner reader = new scanner (system.in);
system.out.print ("Event a number :");
int year = reader.next Int ();
if (((year %. 4 == 0) && (year %. 100 != 0)) ||
(year %. 400 == 0))
system.out.println (year + " is leap yer");
else
system.out.println (year + " is Not Leap year");
}
}
```

- Program to display even number.

```
public class examplewhile
{
public static void main (string [] args)
```

```
{
int i= 2;
while (i<= 10)
{
system.out.println(i);
i+=2;
}
}
}
```

- Program to display the array element in descending order.

```
import java.util.Scanner;
public class Exdessort.
{
public static void main (string [] args)
{
 int n, temp;
scanner sc = new scanner (system.in);
system.out.print ("Enter the limit :");
n = sc.nextInt ();
int [] array = new int [10];
system.out.println ("Enter the elements of the array:
for (int i=0; i<n; i++)
{
array [i] = sc.nexInt ();
}
system.out.println ("Array elements are:");
for (int i=0; i<n; i++)
{
system.out.print (" " + array [i]);
```

```
}
for (int i=0; i<n; i++)
{
for (int j = i+1; j<n; j++)
{
if (array [i] < array [j]) {
temp = array [i];
array [i] = array [j];
array [j] = temp;
}
}
}
system.out.println (" In sorted Array elements
are : ");
for (int i=0; i<n; i++)
{
system.out.print (" " + array [i]);
}
}
}
```

2. object and class.

- program for accessing class members.

```
public class student
{
string name = "abc";
int age = 20;
void display ()
{
system.out.println("my name is " + name);
system.out.println("I am" + age + years old");
}
public static void main(string args[])
{
studen s = new student ();
s.display ();
}
}
```

* object
The state and behavior of the basic program components is known as object.

- oop - object - oriented programming.

- object in Java.
A real - world entity that has state and behaviors is known as object.

- class in Java.
A class can be defined as a templete that describe the behaviors that object of its type support.

- mutator and accessor method.

* Defining your own calsses.

- access modify.
1) default / friendly.
2) Private
3) Public
4) protected.

* Constructors.
Constructors constructs the value i.e. provides data for the object that is why it is known as constructor.

- Program for Constructors.

```
Class rectangle
{
int length;
int breadth;
Rectangle ()
{
length = 5;
breadth = 6;
}
int are ()
{
int rectArea = length * breadth;
return rectArea;
}
}
```

~~Calss~~ class Constructor Example.

```
{
public static void main (string [] args)
{
Rectangle firstRect = new Rectangle ();
system.out.println ("Area of Rectagular:" + firstRec
area ()); }
}
```

- Parameterized Constructor.

A Constructor that is has parameters is known as parameterized constructor.

Syntax: ConstructorName ([Parameter List])

```
{
}
```

* Constructor overloading

Constructors can be overloaded. constructor having the same name with different parameter lists is called as Constructor overloading.

- program for Constructor overloading

```
Class student
{
int id;
string name;
int age;
student (int i, string n)
{
id = i;
```

```
name = n;
}
student (int i, string n, int a)
{
id = i;
name = n;
age = a;
}
void display () {system.out.println (id + " " + name +
age); }
Public static void main (string args []) {
student s1 = new student (1, "Amar", 22);
student s2 = new student (2, "Amol", 20);
s1.display (); | s2.display (); } }
```

* Array of objects.

It is possible to create array of objects of user created class.

syntax:

```
class_name : arr_name [] = new class_name
[size];
```

* Use of this keyword.

1) This keyword can be used to refer current calss instance variable.
2) This keyword can be used to invoke current class constructor.
3) this keyword can be used to invoke current class method. (implicity).
4) This can be passed as an argument in the method call.

- syntax: this.field.

* static block, static fields and methods.

- program to show use of substring.

```
class strToy
{
public static void main (string args [])
{
string st1 = "Java is platform independent";
string sub = st1.substring (9, 16);
system.out.println ("original string is: +st1");
system.out.println ("substring is: + sub");
}
}
```

- Difference between string and stringbuffer

- program for printf()

```
class StringFormat
{
public static void main (string [] args)
{
System.out.printf ("Integer: %d\n", 15);
system.out.printf ("Floating number with 4 decimal
digits: %.4f\n", 1.123123123123);
system.out.printf ("Floating number with 6 decimal
digits: %.6f\n", 1.123123123123);
System.out.printf ("string: %s, integer: %d, float:
%.6f", "Hello World", 55, 9.1234567);
}
}
```

* static block.
A static block is used to initialize static attributes. static blocks are also called static initialization blocks.
e.x. static
{
// whatever code is needed.
}

* static field
This is accomplished by declaring the fields(s) to be static and such fields are known as astatic field(s).

Syntax.
static datatype fieldName;

* static methods.
It is possible to have static method in a class in the way as we have static fields. If you apply static keyword with any method, is known as static method.

Syntax.
classname.staticmethodname (arg-list);

* predefined classes.
There are many predefined classes in Java. the object class in a root of all classes in the Java technology programming language.

- object class
The object class is the parnet class of all the classes in Java by default. in other words, it is the tormost class of java.

- string class.
The strings in Java are treated as object of tyre 'string' class. this class is present in the package java.lang.

- string buffer class.
string buffer class is a mutable class unlike the string class which is immutable.

- Formatting string
Formated string not only display the string content but also it displays the content in the specified sequence.

Java consists of two methods of formating the strings.
1) Format () method in java.lang.string class.
2) printf () method in printstream class.

1) Format () method.
The format method is a static method. it takes three arguments as input and returns a formatting string.

2) printf () method.

This method is similar to printf use in c programming. It's used similar to a replacement of system.out.println().

* Wrapper classes.
- uses of wrapper classes.

1) Converting primitive number to object.
Integer iobj = new Integer (i)
float fobj = new folat (f)
double dobj = new double (d)

2) Converting objects to primitive numbers.
int i = iobj.intvalue ()
float f = fobj.floatvalue ()
double = dobj.doublevalue ()

3) Converting Numbers of string.
String s3 = Integer.tostring (i);
string s4 = Double.tostring (d);
string s4 = Float.tostring (d);

4) Converting string to Numbers.
int i = Integer.parseInt (str)
double d = Double.parseDouble (str).

- Definiation.
The primitive data types are not objects; they do not belong to any class; they are defined in the language itself. Sometimes, it is required to convert data types into object in Java language.

* Packages.

A ~~Java~~ package can be defined as a group of similar types of classes, interface, enumeration and sub-packages.

- we can defined package using package keyword.

- Two types.

i) Built-in package = Existing buit-in Java packages like Java.lang, Java.util, Java.sql etc.

2) User-defined package.

Java package created by user to categorized classes, interface and enumeration.

* Creating packages.

```
package mypackage;
public class class A
{
 int x;
public void accept (int x1)
{
 x = x1;
}
public void display()
{
system.out.println("x = " + x);
}
}
```

* Accessing packages.

package pkg;

- Program shows the usage of built in packages.

```
import java.lang.math;
class mat
{
public static void main (string args[])
{
double k = java.lang.math.sqrt (25);
system.out.println (k);
}
}
```

* Accessing packages (Import a package).

1) Using packagename.

If we use package.*

2) Using packagename.classname.

If we import package.classname then only declared class.

3) Using fully qualified Name.

If we use fully qualified name then only declared calss of this package.

* User defined packages.

The packages credited by user are called as user defined package.

Ans-

* Program to display the player number and name using of default Constructor.

```
class TestConst
{
int Playnumber;
string Playname;
public TestConst()
{
Playnumber = 10;
Playname = "sachin tendulkar";
}
Public static void main (string [] args).
{
TestConst tc = new Testconst ();
System.out.println ("Default value:");
System.out.println ("play number =" + tc.playernumber);
system.out.println ("play Name =" + tc.Playname);
}
}.
```

* package. :- class Tg.