Module-5

Exploring collection Framework

Collection overview:

The collection in java is a framework that provides an architecture to store and manipulate the group of objects.

Java collections can achieve all the operations that you perform on a data such as searching, sorting, insertion, deletion and manipulation.

Eg:

```
package collection.project;
Public class Arraypgm
{
 Public static void main (String[] args)
{
 int arr[]=new int[10]
 for(int i=1;i<=5;i++)
 {
  arr[i]=3;
 }
 for(int i=1;i<=5;i++)
 {
  System.out.println(arr[i]);
 }
```

```
        arr[11] = 5 ;
    }
}
```

O/P:
3
3
3
3
3
error

array cannot
- size (increase)
- size (decrease)

to overcome this we use collections.

The major problem with static array is size cannot be increased during runtime, and size cannot be decreased during runtime.

Static arrays can hold only homogeneous data elements to resolve the above mentioned problem we use dynamic arrays (collection class)

The collection classes

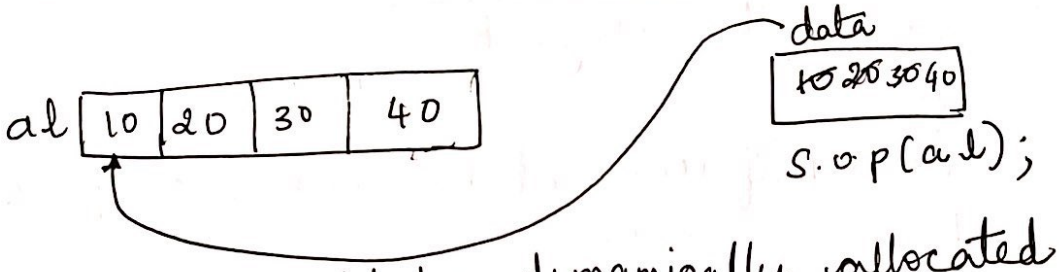
→ The Arrays class / Array list
→ The vector List
→ The Linked List

Array List

They are thread based programs and thread safe.

Array list would grow in size and also can hold heterogeneous elements.

Eg:



Here elements should be dynamically allocated

```
Package collection project;
import java.util.Arraylist;
import java.util.Scanner;

Public class Arrayprogram
{
 Public static void main(String[] args)
 {
  Scanner Scan = new Scanner (System.in);
  int choice;
  Arraylist al = new Arraylist();
  do
  {
   System.out.println("Enter the element");
   int data = Scan.nextInt();
   al.add(data);
   System.out.println("Press 1 if u want to continue
                       or press 2 to stop !!!");
   choice = Scan.nextInt();
  }
   while (choice==1);
   System.out.println(al);
 }
}
```

O/P Enter the element

10

Press 1 if u want to continue or press 2

1

Enter the element

20

Press 1 if u want to continue or press 2

2

10 [10 20]

20

AutoBoxing and UnBoxing

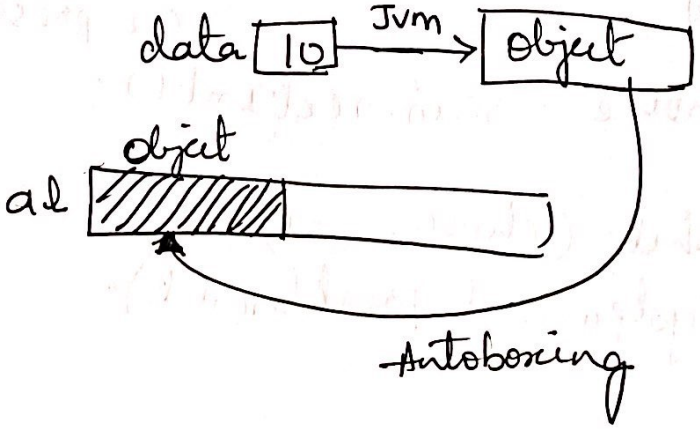
The process of converting the primitive set of data into an object type is called as autoboxing. This would be done by JVM.

The process of converting the object into Primitive type of data is called as unboxing.

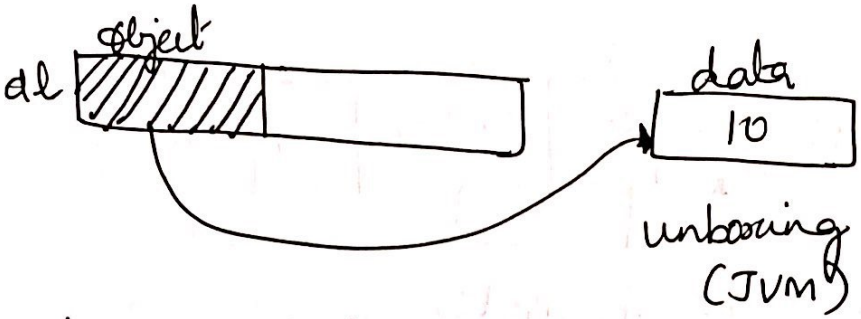
Autoboxing and unboxing would be done by JVM on behalf of user

Array list would grow in size and also can hold heterogeneous elements it is as shown below

al.add(data)



S.o.p(al)

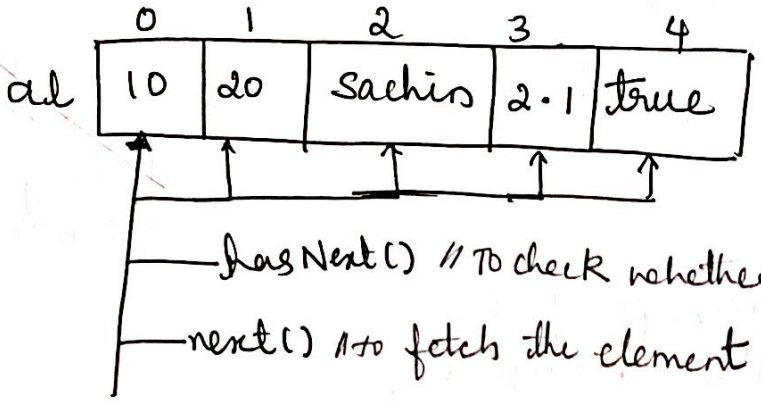


Package collection project
import java.util.Arraylist;
Public class Arraypgmlist
{
Public static void main (String[] args)
{
Arraylist al = new Arraylist();
al.add(10);
al.add("sachin");
al.add(25.5);
}
}

O/P: [10, sachin, 25.5]

The collection interfaces

1. Iterator
2. Iterator list




```
Package collectionsproject
import java.util.*;
Public class Arrayprogiterator
{
Public static void main (String[] args)
{
   arraylist al = new arraylist ();
   al.add (10);
   al.add (20);
   al.add ("sachin");
   al.add (2.1f);
   al.add(true);
System.out.println ("The array elements are:"+al);
System.out.println ();
```

```
Iterator itr = al.iterator();
while (itr.hasNext())
{
 System.out.println(itr.next())
}
System.out.println()
ListIterator litr = al.listIterator();
while (litr.hasNext())
{
 System.out.println(litr.next());
  }
 }
}
```

O/P:

The array elements are: [10, 20, sachin, 2.1, true]

10
20
sachin
2.1
true
} — iterator

10
20
sachin
2.1
true
} — iterator list

Vectorlist

Same example you take and change vectorlist in place of arraylist.

Vector uses a dynamic array to store the data elements. it is similar to the arraylist. However, it is synchronized and contains many methods that are not the part of Collection framework.

Eg: Refer Arraylist

Linked list

Linked list implements the collection interface. it uses a doubly linked list internally to store the elements. it can store the duplicate elements. it maintains the insertion order and is not synchronized. In linked list, the manipulation is fast because no shifting is required.

Eg:
```
import java.util.*;
Public class Linkedlist
{
Public static void main(String [] args)
Linkedlist<String> al = new Linkedlist<String>();
 al.add("Ravi");
 al.add("Vijay");
 al.add("Ravi");
 al.add("Ajay");
 Iterator<String> itr = al.iterator()
 while (itr.hasNext())
 { System.out.println(itr.next());
} }
```

o/p
Ravi
Vijay
Ravi
Ajay

Enumeration, A

→ An enumeration defines a class type, it can have a constructor and also can have instance variable.

→ Enumeration are used to hold the values of any specific type.

→ An enumeration is created using the "enum" keyword.

→ Enum constants are implicitly static and final and cannot be changed once created.

(it is just like an array but it a class type, but not a class it has the features of class)
[Array name & enumeration name should be same]

Eg:
```
public enum WeekDays
{
    Sunday, Monday, Tuesday,
    Wednesday, Thursday,
    Friday, Saturday;
}
```

→ ; is where; Monday, Tuesday are called enumeration constants and each is implicitly declared as a public, static, final member of weekdays.

→ Creating variable of enumeration

enum name → WeekDays day; → variable

→ Assigning Value

```
day = WeekDays.sunday;
```

→ Enumeration constants can be compared

```
if (day == WeekDays.Monday)
```

→ when enumeration constant is displayed it prints the name of the constants

```
System.out.println(weekDays.Monday);
//output Monday.
```

Restrictions on enum

→ Even though enumeration define a class type we cannot instantiate an enum using new.

Eg: weekDays day = new weekDays(); // wrong.
(represent)

The values() and valuesOf() Method

All enumerations automatically have two predefined methods

i) values()
ii) valuesOf()

values() Methods → This method is used to retrive all the values of a function.

Syntax

```
public static enum-type[] values();
```

valuesOf() Methods → This method is used to retrive a particular value of a constant present in a function.

Syntax

```
public static enum-type valueOf(String str);
```

where,
values() and valueOf() methods packaged in the java.lang.Enum.
The str must match exactly an identifier used to declare an enum constant in this type

Eg:

```
enum WeekDays
{
Sunday(1), Monday(2), Tuesday(3), Wednesday(4)
Thursday(5), Friday(6), Saturday(7);
int a; // instance variable
WeekDays(int value,) // constructor type
 {
   a=value;
 }
}
Public class EnumerationApp
{
  Public static void main(String[] args)
  {
    System.out.println("Printing WeekDays using the
                                    numbers");
// WeekDay[] days = WeekDays.values();
for (WeekDay day : days)
{
System.out.println(day)
}
```

taking values from enumeration

Here we r printing

```
System.out.println("The Day using valueOf is "
        +weekDays.valueOf("Tuesday"));
    }
}
```

SSSS → error

↓

Here it checks whether the value is in enumeration or not. (if s it prints) else error

O/P:

printing weekDays using the number
Sunday
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
The Day using valueOf is Tuesday.

Constructor, methods, instancevariable and enumeration

write a java program to create an enumeration Day of week with seven values SUNDAY through SATURDAY. Add a method isWorkday() to the DayOfWeek class that return true if the value on which it called is MONDAY through FRIDAY. For example, the call DayOfWeek.SUNDAY.isWorkDay() return false.

```
import java.util.*;
import java.lang;
import java.io;
enum Week
{
 SUNDAY, MONDAY, TUESDAY , WEDNESDAY, THURSDAY,
  FRIDAY, SATURDAY ;
 }
Class DayOfWeek
{
  Public boolean isWorkDay(Week W)
  {
   if (W. equals(Week. SATURDAY) || W. equals(Week.
                                        SUNDAY))
   {
     return false;
    }else
     { return true;
     }
  }
}
Public class Labprogram8
{
  Public static void main (String[] args)
  {
    Scanner scan = new Scanner(System.in);
    System. out. println ("The days in a week are:");
```

```
Week [] days = Week.values(); // taking values from enumeration
for (Week day : days)
{
  System.out.println(day); // prints
}
String ch;
do
{
System.out.println(" Enter the day to check
      whether working or not");
 String checkDay = scan.next().toUpperCase();
 week day = week.valueOf(checkDay); [if user as entered in smaller case to convert]
 DayOfWeek dweek = new DayOfWeek(); // object creation for class
 if (dweek.isWorkDay(day))
 {
  System.out.println ("it is working day");
 }
 else
 {
  System.out.println (" it is a weed end !!! ");
 }
 System.out.println ("Do u want to continue(Y/N)");
 ch = scan.next();
}while ( !ch.equals ("n"));
 }
}
```

O/P: The days in a week are:

SUNDAY
MONDAY
TUESDAY
WEDNESDAY
THURSDAY
FRIDAY
SATURDAY
Enter the day to check whether working or not
MONDAY
it is working day
Do U want to continue (Y/N)
Y
Enter the day to check whether working or not
Sunday
it is a week end
Do U want to continue (Y/N)
N

Annotations (Metadata)

Definitions : Java Annotations are metadata. Metadata is information about our source code, although they are not a part of the program itself

→ Annotations can be applied to the class, interface, methods, method parameters, local variables,

→ it refers to the extra information given by the compiler to make sure even if the Programmer makes a mistake, it should work with some errors.

:- Annotations are classified into 2 type

1) 3 built-in Annotations used in java code and imported from java.lang:

1. @Override

→ This annotation is used to indicate that the Programmer is overriding a specific method.

→ it assures that the subclass method is overriding the super class method.

→ if the compiler finds that there is no matching method found in superclass that generates a warning

Eg:
```
class Database
{
  Public void connectDB()
  {
   System.out.println("connecting to Database");
  }
}
class Mysql extends Database
{
@Override
  Public void connectDB()
  {
   System.out.println("connecting to Mysql Database");
  }
}
```

```
Public class checkApp()
{
 Public static void main(String[] args)
 {
   Database db = new mysql();
   db.connectDB();
  }
}
```

O/P: connecting to Mysql Database

2- @ Deprecated

This annotation should be used, if the Programmer should tell to the compiler that in the failure process that a particular method should not be used.

Eg:

```
Class Database
{
 Public void connectDB()
  {
  System.out.println("connecting to Database ");
 }
@ Deprecated
 Public void updateDriver()
 {
  System.out.println("updating the Driver");
 }
}
 Class Mysql extends Database
 {
   @ Override
```

```
    Public void connectDB()
    {
    System.out.println("connecting to Mysql Database");
    }
}
Public class checkapp()
{
Public static void main(String[] args)
{
Database db = new Mysql();
db.connectDB();
db.updateDriver();
}
}
```

O/P: connecting to Mysql Database

3. @ Suppress Warnings

This annotations is used to supress the warnings for the variables which are not been used in the program.

Eg:
```
public class checkapp
{
@SuppressWarning("unused")
Public static void main(String[] args)
{
int x=10;
int y;
System.out.println(x);
}
}
```

→ it shows the warning here that "y" is not used. So we are using this annotations.

O/P: 10

Custom Annotations

Custom annotations are such type of annotations where the user can define the values as per his expectations

Eg: @ interface courseDetails → name of the custom annotations

```
{
  String courseName();
  int courseFee();
  int courseDuration();
}
```

it is variable but when it is written inside interface is denoted as methods

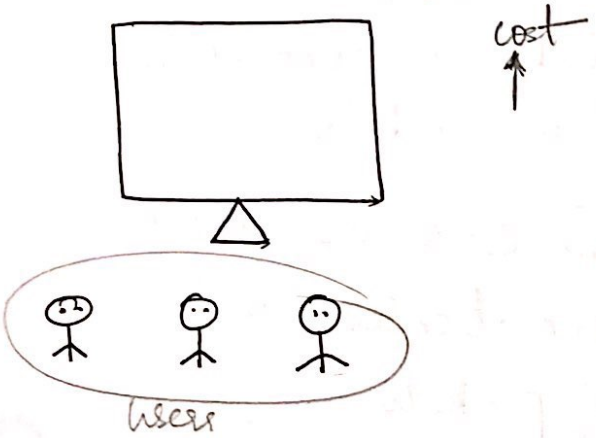
```
@courseDetails(courseName = "Java", courseFee = 3000,
                courseDuration = 90)
class student ()
{
}
Public class studentApp
{
 Public static void main(String[] args)
  {
   }
  }
```

@Target - This is used to indicate where the annotation is applied by default annotation is applied at the class level.

Networking with java.net

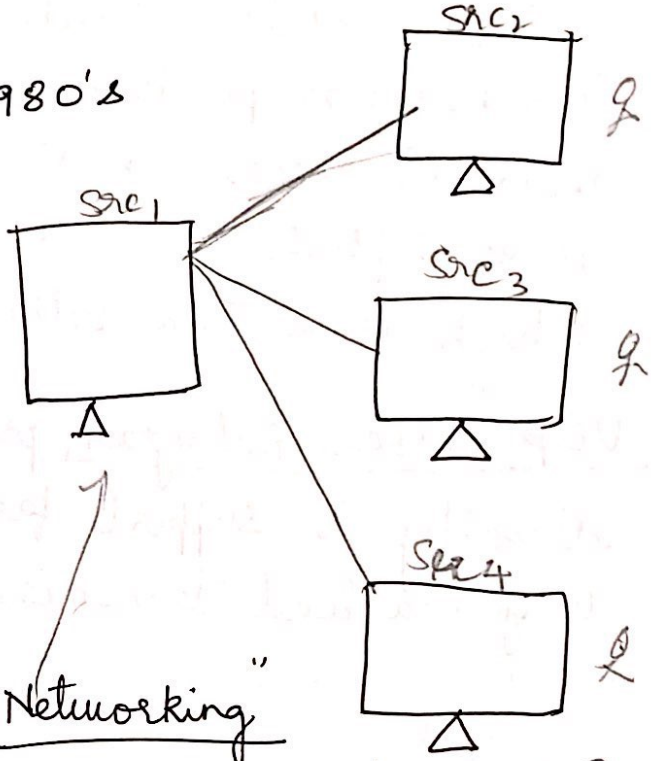
Networking fundamentals

1940's



limited no of computers because cost was high

1980's



"Networking"

[How java supported N/working?]

Networking refers to process of connecting (or more) the computers so that the user can share the resource.

In java, networking is supported through socket programming.

Tcp/Ip and UDP is supported in java for networking. These are the 2 types of communications.

[For communication to happen there is a protocol i.e Tcp/Ip and UDP (hw)

Ip [Internet protocol] - is a low level routing protocol that breaks data into small packets and send them to an address across a n/w.

TCP [Transmission control protocol]

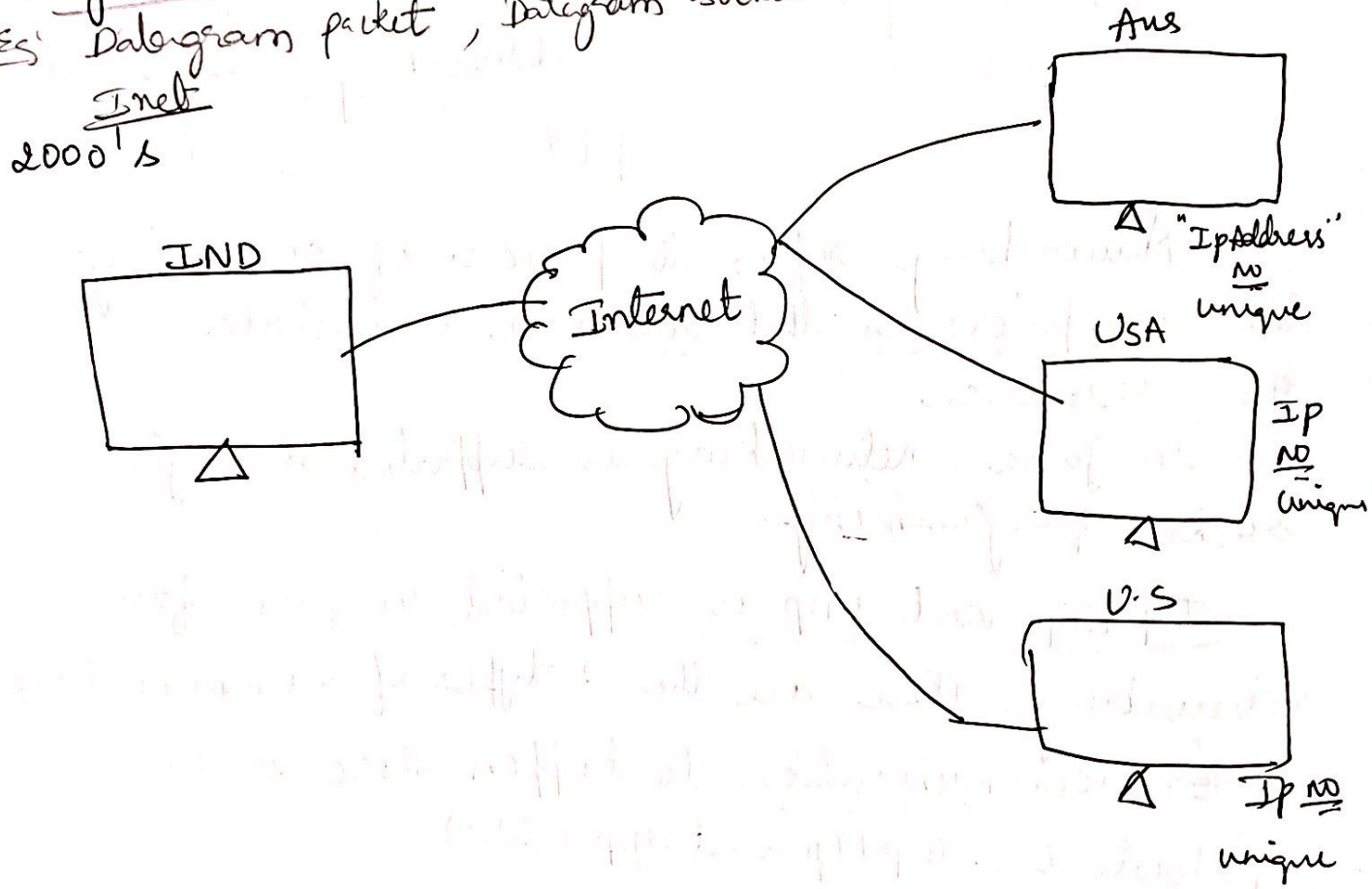
is a higher-level protocol that manages the transmission packets, sorting and retrasmitting them as necessary to reliable trasnmit data. To accomplish this, TCP establishes a connection, which is a link between two sockets.

Eg: URL, URL connection

UDP: [User Datagram protocol] can be used

directly to support fast, connectionless, but unguaranteed transmission of packets.

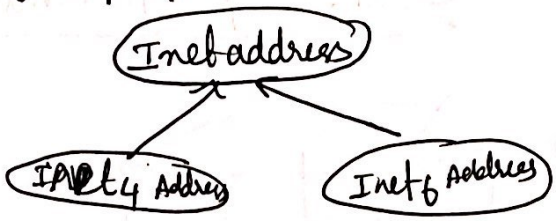
Eg: Datagram packet, Datagram socket



Inet Address class

Inet address encapsulates both numerical Ip address and the domain name for that address.

InetAddress can handle both IPV4 and IPV6 addresses.



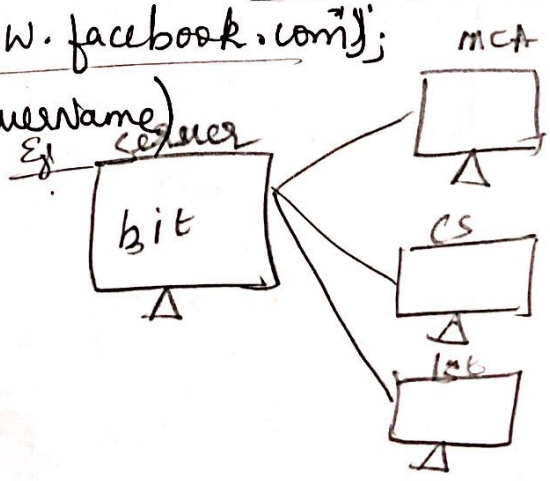
Eg: www.google.com,
www.facebook.com.

```
import java.net.InetAddress
import java.net.UnknownHostException

Public class GoogleApp
{
public static void main (String[] args) throws
{
InetAddress host = InetAddress.getLocalHost();
System.out.println(host);

InetAddress googleserver = InetAddress.getbyName
                           ("www.google.com");
System.out.println(googleserver);

InetAddress serverName = Inet.Address.getAllbyName
                         ("www.facebook.com");
for (InetAddress host1 : servername)
{
System.out.println(host1);
}}
}
```



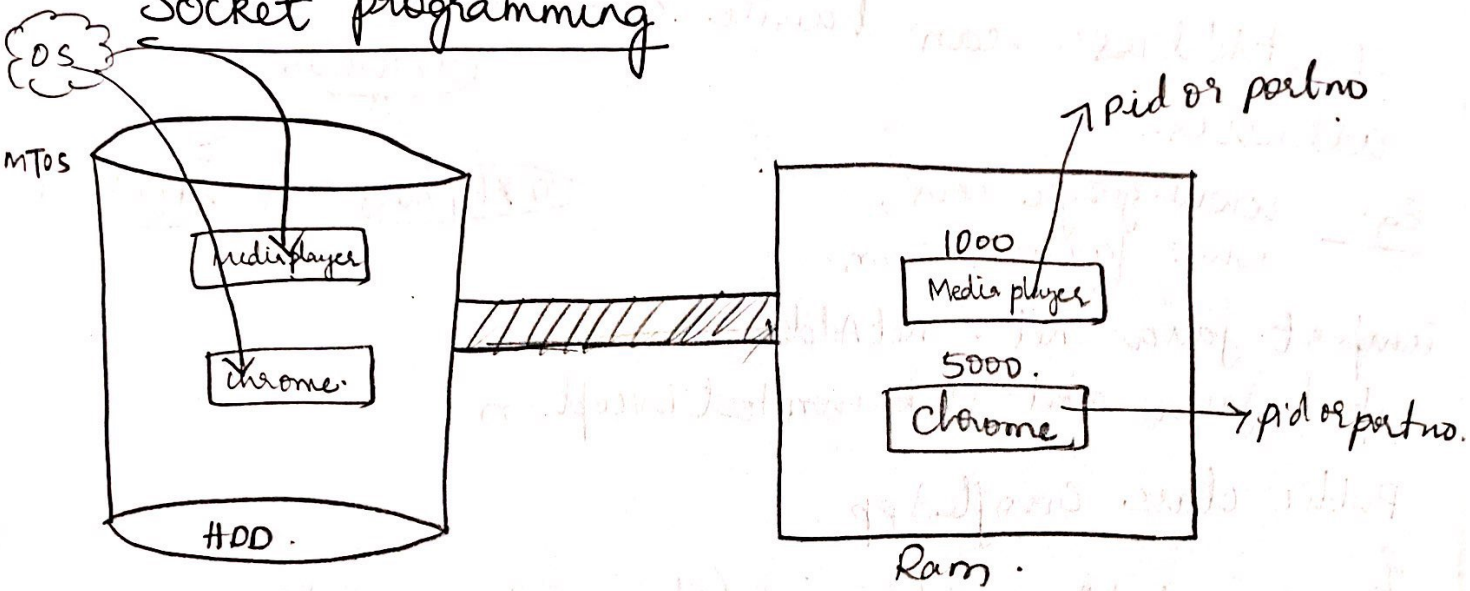
o/p

Saranya.pc/192./68.56.1

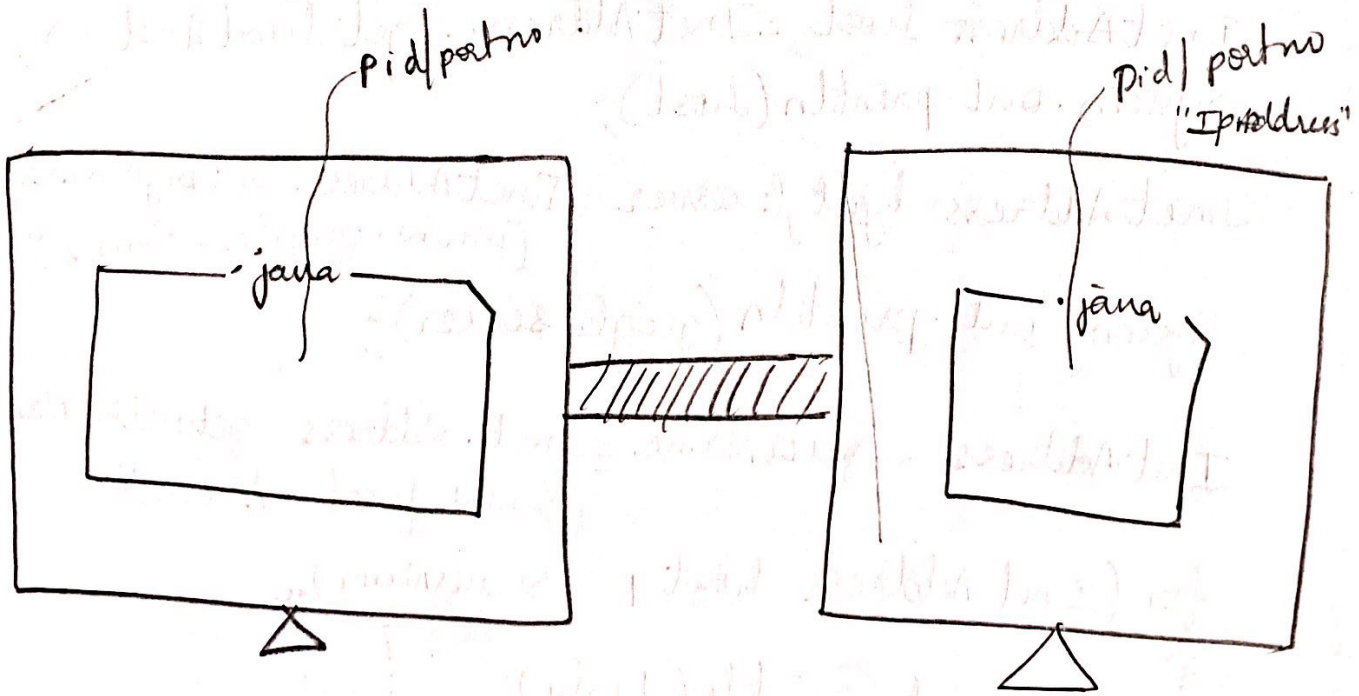
www.google.com/216.58.197.68

www.facebook.com/157.240.192.35

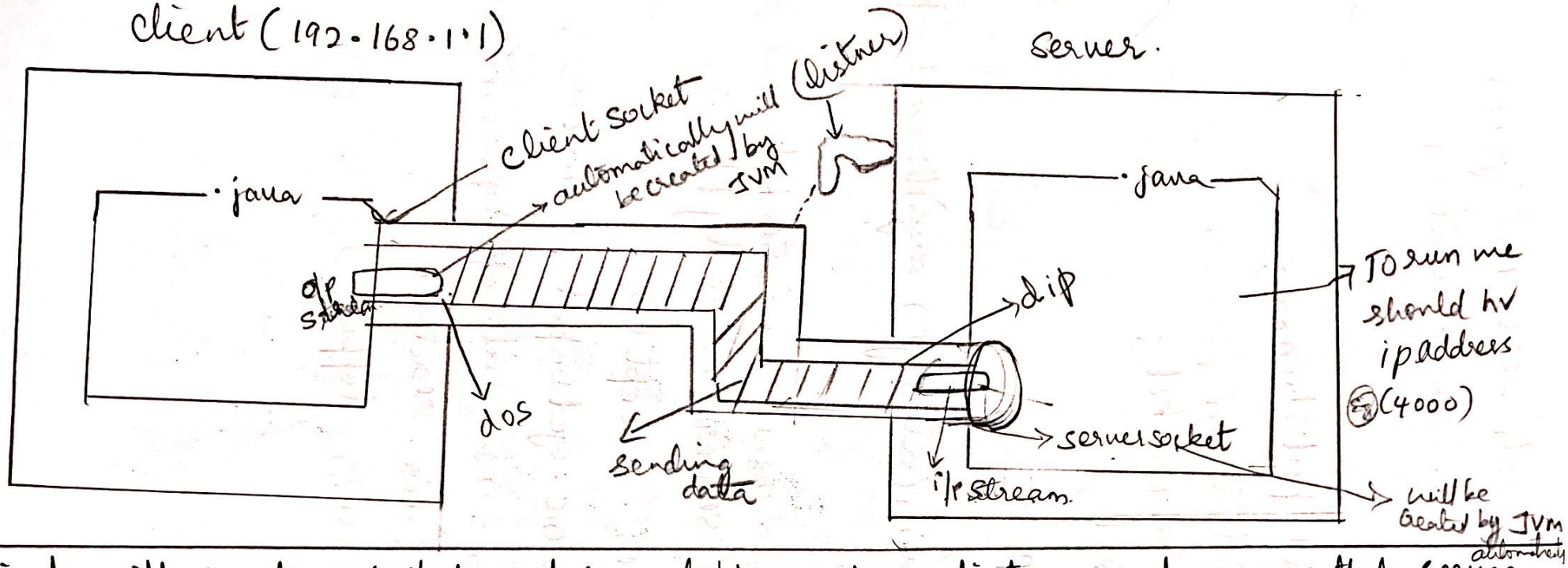
Socket programming



Networking Architecture



connecting 2 computers / phones.



1) Client will create a socket and request will be sent automatically by JVM.

2) get access to o/p streams os and create a new pipe called dos (data o/p stream)

3) send data using write UTF

1) create a listner and annoce that server is ready

2) Listner will accept and create a new socket

3) get access to i/p stream is and create a new pipe called dis. (data i/p stream)

4) Read the data using read UTF

[Unicode translation format]

write a java program which uses Datagram socket for client server communication

Serverapp.java

```
import java.io.DataInputStream;
import java.io.InputStream;
import java.net.ServerSocket;
import java.net.Socket;

Public class ServerApp
{
  Public static void main(String[] args) throws Exception
 {
   ServerSocket listner = new ServerSocket (4000);
   System.out.println ("Server is ready!!!");
   Socket ssoc = listner.accept()
   InputStream is= SSOC.getInputStream( );
   DataInputstream dis = new DataInputStream (is);
   System.out.println (dis.readUTF());
   System.out.println ("server stopped");
  }
}
```

```
import java.io.DataOutputStream;
import java.io.IOException;
import java.io.OutputStream;
import java.net.Socket;
import java.net.UnknownHostException;
import java.util.Scanner;

Public class clientApp
{
  Public static void main(String[] args) throws
                    UnknownHostException, IOException
  {
   System.out.println("client is running");
   Socket CSOC = new Socket("localhost", 4000);
   OutputStream os = CSOC.getOutputStream();
   DataOutputstream dis = new DataOuputstream(os);
   Scanner scan = new Scaner(System.in);
   System.out.println("Enter the message:");
   String data = Scan.next();
   dis.writeUTF(data);
   System.out.println("client stopped");
  }
}
```

O/P Server is ready
Client is ~~ready~~ running
Enter the Message
BIT
Client stopped
Server stopped

URL class

URL represents unified resource locator.
URL is used to get the information of the server computer.

Eg:
```
import java.io.IOException;
import java.net.URL;
Public class URLApp
{
  Public static void main(String[] args) throws
                                    IOException
  {
    URL url = new URL("https://magento.com/
                              home.page");
    System.out.println(url.getcontent());
    System.out.println(url.getfile());
    System.out.println(url.getprotocol());
  }
}
```

O/P:

Sun.net.www.protocol.http.HTTPURLConnection
$HttpInputStream@9f9fb0

content

encrypted info

/home.page → file

https → protocol

URL Connection class

```
import java.io.Exception;
import java.net.URL;
import java.net.URLConnection;
import java.util.Date;

Public class URLconnectionApp
{
 Public static void main(String[] args)throws
                                    IOException
 {
   URL url = new URL("https://magento.com/home-page");

   URLConnection connection = url.Openconnection()
                                  V open is start taking
   longdate = connection.getdate();
   if (date ==0)
     {
     System.out.println("Date not displayed");
   }
   else
   {
     Date d = new Date(date);
     System.out.println(d);
   }
}
```

date which is not in human readable format

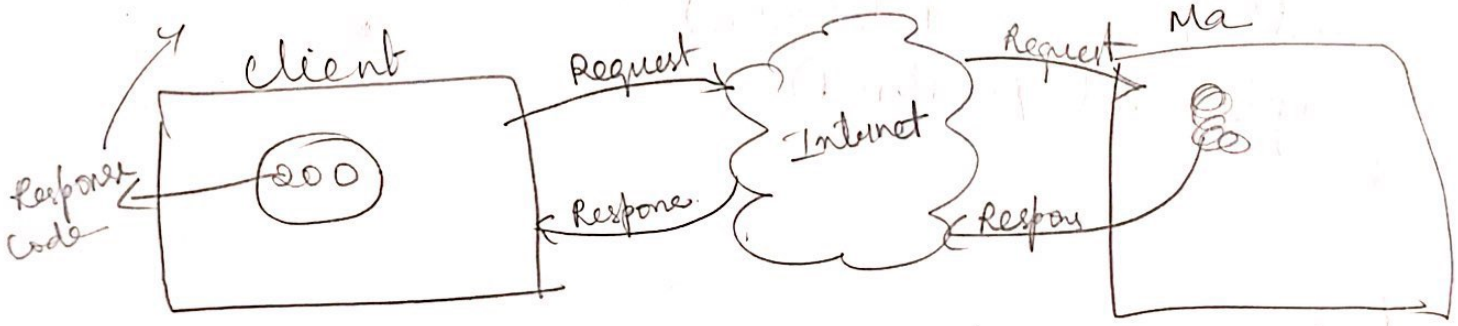
o/p:

Sun Jun 09 18:52:52 IST 2019

HTTP URL Connection

```
import java.io.IOException;
import java.net.URL;
import javax.net.ssl.HttpsURLConnection;
public class URLAppHTTP
{
  public static void main(String[] args) throws
                                        IOException
  {
    URL url = new URL("httpURL =(HTTPURL
                Connection) URL.OpenConnection();
    System.out.println("Request Message is :" +httpURL1.
                                    getRequestMethod());
    System.out.println("Response message is :" +
                        httpURL1.getResponseMessage());
    System.out.println("Response code is :" +
                        httpURL1.getResponseCode());
  }
}
```

op: Request Message is : get



Once response is sent then the response code will be generated and that will be mainted by browser (cachememory)

404 → Error → if the file is not there