we are under CCTV servulance of ~~SUPRIYM~~ continues

1. OOPS

→ Procedure oriented programming: Almari

- it is basically consist of writing a list of instruction for the computer to follow and organizing these instruction in to groups (func)

- procedure - oriented programming.

characteristics:

① algorithm
② large prog. divided into small.
③ most of func share global data.
④ Data move around the system.
⑤ transform data one from another.
⑥ top-down approach program design.

→ features of oops:

- object
- class
- inheritance
- polymorphism
- Abstraction
- encapsulation.
- message passing.

→ what is oops?

- oops is an approach that provide a way of modularizing programs by creating partitioned memory area for both data and function can be used as template for creating copies of modules.

→ objects:

- objects are run time entities.
- they may represent a person, a place anything. (real life entities).
- which as you know, interact by invoking methods.

- objects consist of:

① Attribute: (Properties).
it is represented by attributes of an object
② behaviour: (method / function):
it is represented by method of an object.
③ identity:
it gives a unique name to an obj.
and enables to interact with others.

Ex dog.

identity.	Attributes.	Behaviours
name of dog	Breed	Bark
	Age	Sleep
	color	Eat

Ex

dog tuffy = new dog();

→ classes:

X - classes just maintained that objects contain data.
- class is a collection of similar objects.
- class defines all properties of an object.
- class is just a template or blue print from which object created.
- class does not occupy memory.
- class is group of variables or methods.

- Sy

Access modifier class classname {
// data member
// method.
// constructor.
}

- Ex
Animal, student

- Ex

```
class student {
    int id;
    String name;

    main() {
    Student S1 = new Student();
    }
```

→ Data Abstraction and Encapsulation.

- Abstraction:

- Data Abstraction is the property by virtue of which only the essential details are displayed to the user.
- in relevent details are not show to the user.

ex ATM machine

- Encapsulation: (data hiding).

- it is wrapping up data under the single unit.
- in encapsulation the data in a class is hidden from other classes.
- and can be access only through any members function of the class.

ex department of the company.

→ inheritance:

- it's allow classes to derived properties from another class.

- sub class / derived / base class: inherit
- parent / base / super class: / inherited by subclass.

ex parent class → Animals
child class → dog, cat, horse.

- Reusablity

→ polymorphism:

- polymorphism means exist in many forms. to be displayed in more than one form.

- operator overloading.
- function overloading.

ex at a same time can have different characteristic like father, husband, employee etc--

→ Dynamic Binding:

- compiler doesn't decide the method to be called.
- overriding is the exe. of dynamic binding.

→ message passing:

- objects communicate with one other by sending and receiving info. to each other.
- A msg for an obj is req for execution of the procedure.

→ Application of OOP:

- Real time system.
- simulation and modeling
- object oriented dutubase
- hypertext, hypermedia.
- AI and expert systems.
- neural networks and parallel programming.

2. intro to Java

- Java is high level, robust object-oriented and secure programming language.
- developed by sun microsystem in 1995.

→ features of Java:

1. object-oriented
2. plateform independent
3. simple
4. secure
5. Architecture-neutral
6. portable
7. Robust: Eh, Oc.
8. interpreted
9. high performance.

→ types of Java Applications:

1. Standalone Application
2. web Application
3. Enterprise Application.
4. mobile Application.

var type
local
instance
class

- Java SE (Java Standard edition)
- Java EE (Java Enterprise Edition)
- Java ME (Java micro Edition).

→ Array:

- Array is a group of similer types of data.

eg

```
int arr[];
arr = new int[];
```

Ex

int monthDays[] = {31, 28, 31, 31, 30};

→ operators in Java:

- Arithmetic operators: +, -, *, /, %.
- increment / decrement operators: ++, --
- Relational operators: ==, !=, <=, >=, <, >
- Bitwise operators: &, |, ^, <<, >>
- logical operators: &&, ||, !
- Assignment operators: =, +=, -=, *=, /=,
- misc operators: ?: (ternary)

→ Java Decision making:

- if, if..else, if..else if..else
- switch

→ Java loops:

① for:
② while
③ Do..while
④ for each

- break
- continue

Diff:

- literal - constant pool
  obj - dynamic, heap state.

→ String:

→ String constructors:

- String()
- String(String str)
- String(char str[])
- String(char str[], int num, int num2)

→ Some string methods:

- charAt(int): Returns characters at the index
- compareTo(String):
- concat(String)
- copyValueOf(char[], int, int).

→ Java String:

- char[] ch = {'m', 'c', 'a', 'd', 'd', 'u'};
  String s = new String(ch);
  String s = "mcA ddu";

- String is an object that represents sequence of characters.
- Java.lang package.
- String object are immutable. Can't be change

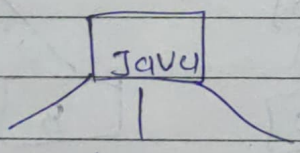
packages:

* types of import
  1. Explicit import:
     import. Java. ~~lang~~ util. Scanner
  2. implicit import:
     import Java. util. *

* package:
  a group of related types (classes, interface, annotesion, subpackages) provide to access protection and name space mang.

- two classes not present with same name.
- two pkg will have to two class with same.

- saparated with .(dot)
  Ad:
  - categorized
  - naming collision.
  - access protection.
- Pkg hirarchy.



Util lang awt
Arraylist string Button.
map System

* inheritence:

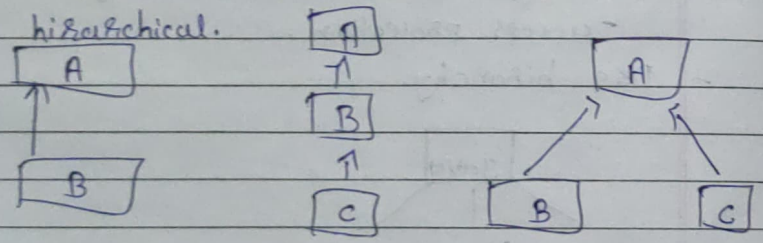
a mechanism in which one object acquires all the properties and behaviours of a parent object.

* why?
  1. method overriding
  2. code reusablity.

- child class
- parent class.
- extend keyword.
- super keyword.

* types of inheritence.

1. single
2. ~~multiple~~ multilevel
3. hirarchical.



* method overriding:

when you extend a class so you can change behaviour of class is changed is called.

- name is same but list of argument is not same.

* ~~method overriding.~~
* type casting:

parent to child - down casting
child to parent - up casting.

1. upcasting: (generalization and widening)

don't access to variables and methods etc...

example

```
Parent obj = (Parent) new child();
```

2. Downcasting:
   - parent ref to child class.
   - we can't assign directly. it is not throw error but throws class cast exception.

Example

```
Parent p = new child();
child c = child p;
```

* Abstract class.

Abstract keyword
non-Abstract + Abstract method.
data hiding & essential info show by user.

- you can't create object of Abstract class.
- final keyword.

abstract void print();

* interface:

- blue print of class.
- 100% achive abstraction.
- is-A Relation ship

interface keyword

interface interfacename {

}

- Abstract class vs interface.
- overloading vs overriding.
* ~~wrapper~~ object class:

- object class is universal superclass.
- parent class of all java classes.

employee e = (Employee) obj;

- hashcode() - hashcode number
- equals (Object obj) - compare
- clone() :- copy of object
- to string - String of object.

java.lang.object

equals() : match the object address.
= = : match type of object.

- to string :
  by default compiler returns student@1234.

- wrapper classes:

- convert premitive into Object and object to premitive.
- Require more memory.

boolean	Boolean
char	Character
byte	Byte
short	short
int	Integer
long	Long
float	Float
double	Double.

Ex:

```
int a=20;
Integer i=Integer.valueof(a);

Integer a=new Integer(3);
int i=a.intvalue();
```

* object cloning:

```
person p = new person();
Person p1 = p;  // share memory location.

//clone obj.

person p = new person();
Person p1 = (person) p.clone();
```

- we require cloneable interface for clone.

Shallow copy	Deep
only ref.	actual copy
no new memory assign	assign.
changes in one so reflect in both.	not reflect.
default version of clone() is.	override clone().
less expensive	highly expensive.
not disjoin	disjoin

Java.lang.clonable interface.

* Java reflection:

- all about class: access modifiers, superclass, field, constructors, methods.

Java.lang.class.

- metadata of class.

- methods:
  1. getname() - classname (type of class).
  2. forname() - ref of class. (name of class known)
  3. newInstance - obj of class.()
  4. isinterface
  5. isArray.

getdeclaredfields
getdeclaredmethods
getdeclaredconstructors.
~~getdeclaredmethod~~

* nested classes:

types of nested class.

1. non-static nested class (inner)
   - (within class outside method) member inner class
   - (within method) Anonymous inner class
   - local inner class.
2. static nested class. (staticclass).

-> object inner.

```
Person p = new person();  // outer.
person.Employee in = p.new Employee();//inn
```

* Exception handling:

- handle runtime error so that maintain normal flow of execution.

Java.lang.Throwable root class.

→ types of Exceptions:

① Checked - except runtime, error. (IO, SQL, JVM exception)
② unchecked. - check runtime (Runtime).

① try
② catch
③ finally
④ throw (custom msg).
⑤ throws : (used method)

* multithreading:

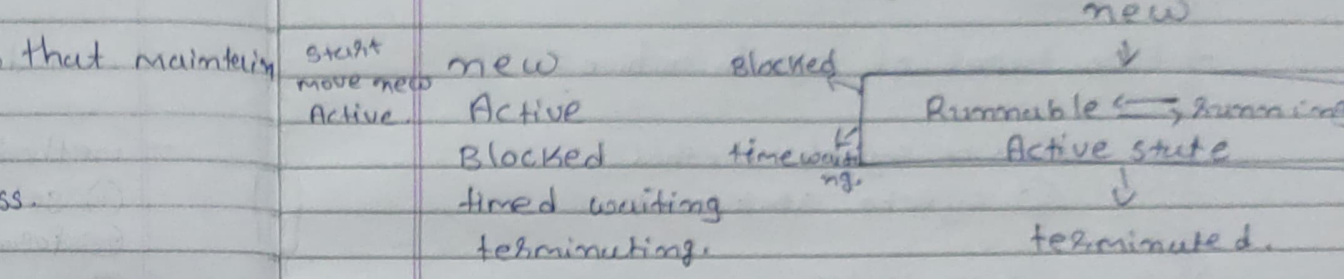
- thread is light wight sub-process.

multitasking = multi process + multithreading.

thread class to used for thread prog.

methods:
start()
run()
sleep()
Join()
getPriority
setPriority
getname
setname.
get id.

thread state.



* two ways to create thread.

① extending thread class. (methods).
② implement Runnable interface. (run())

* thread class.

```
class multi extends thread {

    public void run(){
        s.out.
    }

    main
    multi m1 = new multi();
    m1.start();
```

* interface Runnable.

```
class multi implements Runnable {
    public void run(){
        s.out }
    main(){ multi m1 = new multi();
    thread t1 = new thread(m1);
    t1.start();
```

* thread synchronization:

- two types of synch. achive.

① use synch. method
② syncho. statement.

→ Java collections:

list ->
- Array list
- linked list
- vector
- stack

Queue ->
- dequeue
- priority queue.

set ->
hashset
linked hashset.

-> collection interface. methods.

- add
- addall
- remove
- removeall
- isempty
- clear
- iterator
- to Array.
- hashcode.

-> list:
- order collection
- duplicate allow
- null allow.

- methods:
  - add
  - addall
  - remove
  - removeall
  - isEmpty
  - to Array
  - get
  - equals
  - set
  - size

- Arraydeque:
  - null not Allow.
  - faster than linkedlist, stack.

- hash set:
  - unique element.
  - allow null.

- tree set:
  - unique element
  - doe's not allow null.
  - Asc & desc. order.

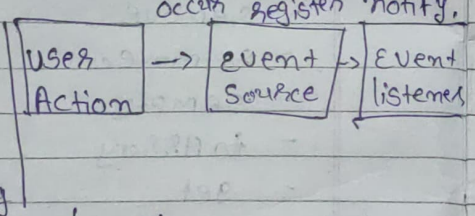
hash map:
- key value pair
-

* Applet:

life cycle:

1. init()
2. start()
3. paint()
4. stop()
5. destroy()

* Event handling:



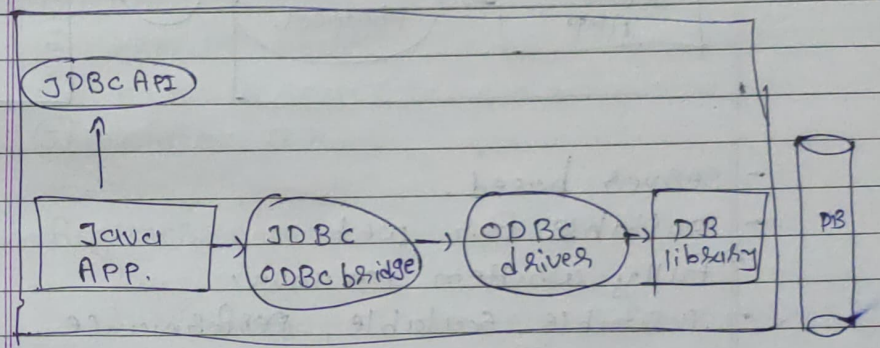
-> Event
-> Event handling
-> Delegation event handling model.
-> Components of DEH Model.

1. Event
   -> foreground event
   -> Background event.
2. Event source
   - Register event.
   - change internal state.
   keyboard -> add keyListener()
   mouse -> add mouse motion listener
   - multicasting | unicasting.
3. Event listener:
   - notify user
   - interaction with user & GUI
   - mouse motion listener.

* JDBC:

* JDBC drivers:

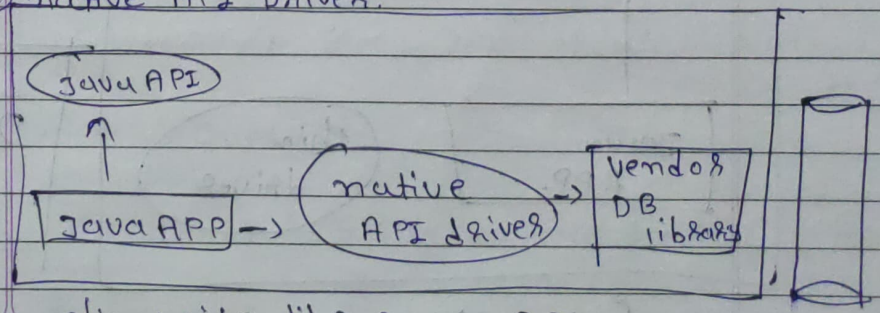
1. JDBC-ODBC bridge driver.



-used for deveopment & testing.

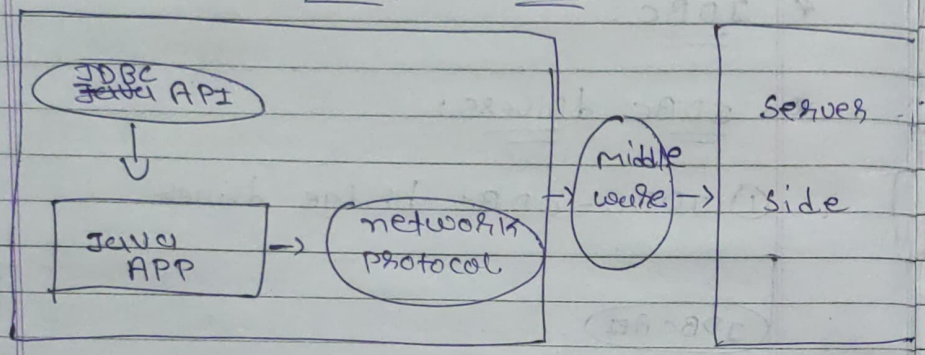
Dis:
- not portable
- fully in Java.
- slowest.

2. native API Driver.



-client side library on DB.
- AD: better performance.
- Dis: portability issue.

③ network protocol driver.

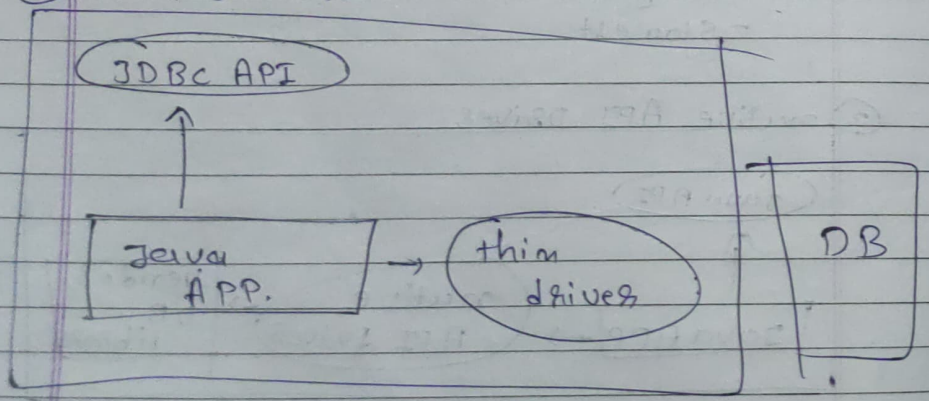


- server based.
- suitable for web.
- fully written in Java.
- portable, scalable, performance.
- most efficient from others.

Dis:

- install maintance.

④ thin driver.



- fully written Java.

Dis:
different driver for each dbs.

* Connectivity JDBC:

① Register classes.
② create con.
③ create state.
④ execute state.
⑤ close con.

① Register classes:

```
class.forname("com.mysql.JDBC.drivers");
```

② create conn.

```
Connection con = DriverManger.get
    Connection("JDBC:mysql://localhost
    :3306/Employee","root","pass");
```

③ statement creations.

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

④ executeQuery():

```
Resultset Rs = stm.executeQuery("select
    * from emp");
```

while(rs.next()){

rs.getInt(1);

⑤ connection Close.

con.close();

* networking programming:

- TCP: transmission control protocol.
- UDP: User Datagram Protocol.
- MAC = media Access control.

server.java.

ServerSocket ss = new ServerSocket(6666);

Socket s=ss.accept();

DataInputStream DI = DataInputStream(s.getOutputStream());

String s = DI.readUTF();

Print(s);

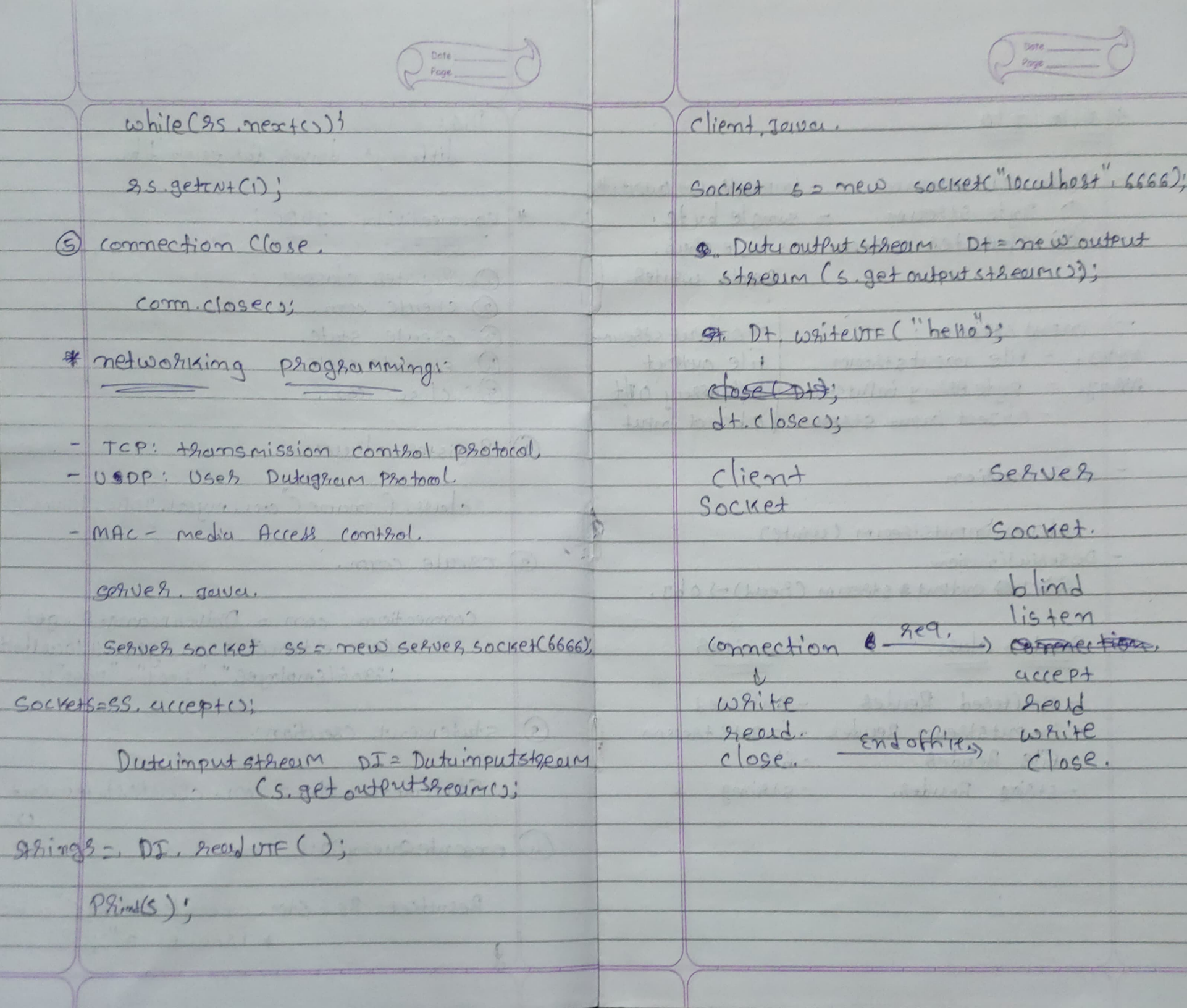
client.java.

Socket s = new Socket("localhost", 6666);

DataOutputStream Dt = new outputstream(s.getoutputstream());

Dt.writeUTF("hello");

~~close(Dt);~~
dt.close();



File IO

- types of stream: inputstream, outputstream
  1. Byte stream - single byte.
  2. Character stream - single char.
     Reader, writer.

→ input stream :	→ outputstream
bytes. - file inputstream	file output
Array. - Byte Array inputstream	Byte Array o/t
- object input stream.	object input

- serialization.

  obj -> inputstream (write)
- Deserialization
  obj output stream (Read) -> obj.

- Reader class: Writer

- Buffered Reader	- Buffered
- inputstream Reader	- inputstream
- file Reader	- file
- string Reader.	- string.