```
class H extend B
{
    p s v m( )
    {
        new Honda().run();
    }
}
```

* Abstract Class:-

Eg. abstract class CalculateOp
{
    public abstract void add(); // abstract method
    " " " sub();
};

* Abstract method is a do-nothing method, It does not have a body.
* We cannot make object of abstract class.

→ Abstract method is always made in abstract class, not normal class.

→ Abstract class may/not include include abs. method.

OR

→ A class can have abstract & non-abstract method
→ Abs. class can have data member, abs. method, method body (non-abs. method → static) & constructor

```
abstract class calOp
{
    public abstract void add();
        "       "        sub();

    void disp()
    {
    }   sop("Hii");

class cal extends calOp
{
    public void add()
    {  }

    public void sub()
    {  }
}
```

→ Abs. class means a class which is declared with "abstract" key.
→ It may / not include abs. method.
→ Abs. method means a method which are only declared & not defined. (no body).
→ It needs to be extended & its method implemented, means abs. method which are there in ~~the~~ super class, must be overrided in subclass.
→ Else, it gives error msg.
→ We cannot create an instance of abs. class as it is an incomplete class.

abstract → diff. b/w GUI & their implementation.
Provide data to outside users & hide their info.

Date
Page

☐ Syntax to create abstract method – (no body)

⇒ Interface provides pure extraction.

Q. Create an abstract class CalculatorOperation with fol. abs. method → add(), sub(), mult(), & div();
Create a menu driven prog. in Calculator class such as 1 for Add, 2 Sub, 3 ~~div~~ mult, & 4 div, 5 exit.
A class Calculator that implements the abstract class CalculatorOperation.

Q. WAP which has-
(i) an abstract class for stack operations.
(ii) a class that implements the abstract stack class & creates a fixed length stack.

```
abstract class StackOp
{
    public abstract void push();
                        pop();
                        peek();
                        disp();
}

class Stack extends StackOp
{
```

same prog. for interface.

Q. We have to cal. area of rect, sq. & circle. Create an abs. class 'Shape' with 3 abstract methods namely 'RectArea', 'SqArea' & 'CircleArea'. Now create another class 'Area' inheriting class Shape. Class 'Area' containing all the 3 method 'RectArea', 'SqArea' ; 'CircleArea' for printing the area of rect, sq & circle resp. Create an obj. of class 'Area' & call the func^n. ().

Q. Create abs. class 'Bank' with Abs. Method getRateOfInterest 7, 7.5, 8 are interest rate in banks SBI, PNB & UCO Bank resp. SBI, PNB & UCO are subclasses of class 'Bank' each having a method named getRateOfInterest. Create reference variable of class 'Bank' & call the func^n. () for all 3 methods.

Q. Define a class Employee having members — id, name, dept, salary. Create a subclass called "Manager" with private member bonus. Define methods accept and display in both the classes. Create n no. of objects of Manager class, & disp. details of Manager having max total salary (sal + bonus).

24/8/24

Q. WAJP to create a super class Vehicle having members Company & price. Derive 2 different classes LightMotorVehicle (members - mileage) and HeavyMotorVehicle (memb- capacity in tons). Accept info of n vehicles & disp info in approp. form. While taking data, ask user about type of vehicle first.

⇒ Java has automatic memory mang.

V. Imp. (exam)

* Java Garbage Collection :-

→ Process by which Java program perform automatic memory mang.
→ Java prog. compile to bytecode that can be run on a Java Virtual Machine (JVM).
→ When Java prog. runs on JVM, obj. are created on the heap, which is a portion of memory dedicated to the prog.
→ So, some obj. will no longer be needed. The garbage collector finds these unused obj. & deletes them to free up memory.

⇒ "Garbage means unreferenced object."

* Advantage of Garbage Collec. :-

• Makes Java memory efficient because garbage collector removes unref. obj. from heap memory.

- It is automatically done by the garb. coll. (a part of JVM), so we don't need to make extra efforts.

* How can an object be unreferenced?

→ By nulling the reference
→ By assigning a reference to another
→ By anonymous obj. etc.

□ By nulling a reference:

```
Employee e = new Employee();
e = null;
```

□ By assig. reference to another:

```
Employee e1 = new Employee();
Employee e2 = new Employee();
e1 = e2; // first obj referenced by e1 is available for garb. collection
```

□ By anonymous obj.:

```
new Employee();
```

* finalize() method :-

It is invoked each time before the obj. is garbage collected. This method can be used to perform cleanup processing. This is

defined in obj. class.

eg. protected void finalize()
{ }

* gc() method :-

→ The gc() method is used to invoke the garbage collector to perform cleanup processing.

→ It is found in System & Runtime classes.

Syntax → public static void gc() { }

eg. (Explicit garb. colle.)

```
public class TestGarbage
{
  public void finalize()
  {
    S.o.p("obj. is garb. coll.");
  }
  public s v m( )
  {
    TestGarbage s1 = new TestGarbage();
    s1 = null;
    System.gc();
  }
}
```

O/P → obj. is garb. collected.

* Why Java has better memory mang?

We use free() in C & delete() in C++.
But, Java has automatic mem. mang.
So, Java provides better mem. mang.

* Garbage Collection (in other words):

→ Process of reclaiming the runtime unused memory automatically.
→ It is a way to destroy unused obj.

⇒ Garb. coll. is automatic done implicitly.

⇒ To do it explicitly, we need to override the finalize() func.

* Abstract class eg

```
public class Cal extends CalOp
{
    @Override
    public void add()
    {
    }
    @Override
    public void sub()
    {
    }
```

Annotation (just as comment)

Q. Define an abstract class "Staff" with abstract method accept & display. Define 2 subclasses of this class i.e., Employee & Manager. Employee class have members — id, name, dept, salary; and Manager class with private member bonus and total salary. Create n no of objects of manager class & display detail of manager having the maximum total salary i.e. (salary + bonus)

* Interface :-

→ defines what a class must do without saying anything about its implementation.

→ It is a mechanism to achieve abstraction.

→ Used to support functionality of multiple inheritance.

→ Can have abstract method & variables. It cannot have a method body.

Eg. interface StackOp
{
    public abstract void push();
    " int pop();
    " void peek();
    " void display();
}

```
class Stack implements StackOp
{
    public void push()
    {
    }
}
```

* Why use Java interface?

→ Used to achieve abstraction.
→ Used to achieve loose coupling.
→ " support functionality of multiple inheritance.

* Tight Coupling:-

→ Means dependent to each other.

NOTE:-

Types of phones → CDMA (dependent) Jio phone
→ GSM (not-dependent) Android

- Interface fields are public, static & final by default.
- Interface methods are public & abstract by default

eg. interface Printable
{
int MIN=5;
void print;
}
Printable.java

→ compiler →

interface Printable
{
public static final int MIN=5;
public abstract void print();
}
Printable class.

Mouse Listener
6 methods (imp. to learn).

⇒ A class uses "implement" keyword to implement an interface.

Q. WAJP which has –
(i) an interface for stack oper.ⁿ
(ii) a class that implements stack interface & creates a fixed length stack.

Q. Same as abs. class (area of rect., circle,...).

eg. ~~interface~~

* Multiple Inheritance by interface :-

If a class implements multiple interfaces
OR
one class & one interface
OR
An interface extends multiple interfaces.

is known as multiple inheritance.

eg1.

```
interface A
{
    void m1();
    void m3();
}

interface B
{
    void m2();
    void m4();
}

class C implements A,B
{
    public void m1()
    { }
    public void m2()
    { }
};
```

eg2.

```
interface Tax
{
    int trade = 5;
    void calTax();
}

class Publication
{
}
```

class Book extends Publication implements Tax
{
};

* Extending an Interface (ppt)

25/8/24

* Package :-

→ A Java package is a group of similar types of classes, interfaces & sub-packages.

→ It removes naming collision. (advantage)

→ Reuse the classes contained in the packages of other pgms.

→ Two forms, • Built-in
• User-defined (created by developer)

→ Built-in → java, lang, awt, javax, io, util, etc. are eg.

→ Reduces project development time. (adv.)

→ Any class inside package is always public (compulsory).

→ "package" keyword used to create package in Java.

→ A package is always public.

eg. package loop;

```
public class Factorial
{
    int i, f=1;
    public void findFact(int no)
    {
        for(i=1; i<=no; i++)
            f = f*i;
        S.O.P("Factorial =" +f);
    }
};
```

Source file

d:\ Factorial.java

Compile

d:\ javac -d. Factorial.java ↵

This statement automatically creates folder i.e., loop & put class file under this folder.

Use of package → import loop.Factorial;

```
class Test
{
    P S V m( )
```

=> Study Namespace concept

```
{
    Factorial f = new Factorial();
    f.findFact(5);
}
}
```

7/9/24.

* Adv. of Packages :-

→ Make easy searching or locating of classes & interfaces.
→ ~~Ad~~ Avoid name collision.

Ways to make a package: ① Manual
② Automatic

Syntax - (Hierarchy) of Pkge.
package. pkage1. pkage2. package3;

Exam:
* How to access package from another package?

NOTE:-
import p1.*; ← may sometimes not work in Notepad with *, so write ABC (any name).
OR
import p1.Peerson;

□ Can we use package in inheritance?
Yes.

Q. Create a class calculator with add(), sub(), mult() & div() inside pkage name lotnic calculato

Q. ~~Test~~ Create a menu driven prog. in SimpleCalculator class for foll. oper?? such as: Add, Sub, Mult, Div, Exit SimpleCalculator use methods of calculator.

⇒ A set of classes can be created in a package.

9/9/24

* What is JAR?

→ JAR stands for Java ARchive. It is a file format based on the popular ZIP file format and is used for aggregating many files into one.

→ Although JAR can be used as a general archiving tool, the primary motivation for its develop. was so that Java applets and their requisite components (.class files, images & sounds) can be downloaded to a browser in a single HTTP transaction, rather than a new connection for each piece.

→ This greatly improves the speed with which an applet can be loaded onto a webpage & begin functioning.

→ JAR format also supports .compression, which reduces the size of the file & improves download time still further.

→ Additionally, individual entries in a JAR file maybe digitally signed by the applet author to authenticate their origin.

□ JAR is –

- the only archive format that is cross platform
- the only img format that handles audio & image files as well as class files.
- backward-compatible with existing applet code
- an open standard, fully extendable, & written in java.

9/9/24

* Access Specifiers in Java :-

→ Regulate/control access to classes, fields & methods.
→ Determine whether a field or method in a class can be used.

Access Modifiers	default	private	protected	public
(1) accessible inside the class	Yes	Yes	Yes	Yes
(2) Accessible within the same subclass inside the same package	Y	No	Y	Y
(3) Accessible outside the package	N	N	N	Y
(4) Accessible within the subclass, outside the package	N	N	Y	Y

□ Public specifier

→ highest level of accessibility
→ classes, methods, fields declared as public can be accessed from any class.

□ Private

→ lowest level of accessibility
→ private methods & fields can only be accessed within same class to which methods & fields belong.
→ class can't be made private

localhost - loopback protocol

Date Page

□ Protected

→ Methods & fields declared as protected can be accessed by the subclass in the same package or another subclass in other package.

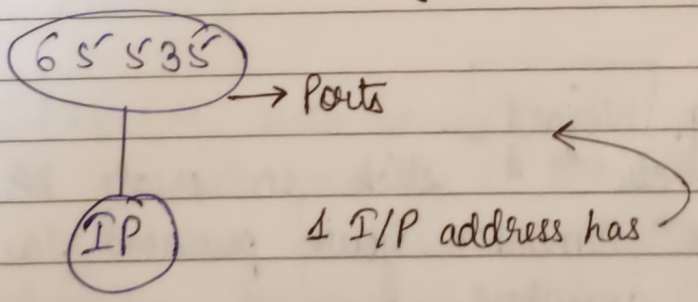
□ Default

* Java N/W ing: (ppt)

13/9/24

Same ppt (N/W ing)

UDP → Usergram Data Protocol
Socket → Send/receive info
Packet → is nothing but information



* Imp. Qus:-

Q1. Java is a robust lang. Describe. (4 mm)

Q2. Why do we create inner class (or nested class)? What are its restrictions? Explain with eg. (3)

Q3. Can the rows in a 2-dimensional array have diff. length? If yes, then describe with eg.

Q4. Distinguish b/w the following -

(a) String & String buffer
(b) length & capacity func^n.
(c) length & length func^n.
(d) ~~set charAt~~ setCharAt & Insert func^n.
(e) Equals & ==.
(f) Typecasting
(g) String builder

Q5. What is boxing & unboxing?

14/9/24

* Database :-

Derbi DB.

- DB using Wizard
→ Right click & click on create DB.
→ Always remember your password, as it is not saved anywhere.
→ Always add DB after your name of DB.
eg. classDB, lmsDB, etc.

→ To see Data, right click on Service & then view Data.

Command.DB.

14/9/24

* Wrapper Class in Java :-

Primitive DT (8 wrapper class in total)	Wrapper class
boolean	boolean
char	character
byte	byte
short	short
int	integer
long	long
float	float
double	double

→ This, in java provides the mechanism to convert primitive DT into object is called boxing and object into primitive DT is called unboxing.

→ Since J2SE 5.0, auto boxing and unboxing features converts primitive DT into object and obj. into primitive DT autom. The automatic conversion of primitive DT into obj. is known as auto-boxing and vice-versa auto-unboxing.

→ One of the eight classes of java lang. package are known as wrapper class in Java.

→ This list of 8 wrapper class are given above.

Wrapper class eg.; primitive → wrapper

eg. public class Wrapper example 1
{
p s v m ( )
p{
int a = 20;
Integer x = Integer.valueof (a);// boxing
Integer y = a; //auto boxing,
now compiler will write
Integer.valueof (a) internally.
S.O.p (a+" "+i+" "j);
}

O/P → 20 20 20

⇒ NO func? in java is ordinary but member func?.

String b = Integer.toBinaryString (a)

eg. OR
p s v m ( )
p{
BR br = new BR (System.in);
Integer no;
S.O.p ("enter number");

can be used in

- public static string toBinaryString (int i)

→ Returns a string represented. (ppt)

⇒ Character too has no of methods → isdigit(), islowercase(), isuppercase()

Q. WAJP to accept one char from the command line user. Check whether the given character is upper case, lower case, digit or special character. Solve this problem by using the concept of wrapper class.

```
// Converting integer to int
Integer i = new Integer(10);
```

OR

```
Integer i=10;
int x = i.intValue();   // unboxing
int j = a;              // auto unboxing
S.O.P(a+" "+i+" "+j);

P S V M() throws Exception
{
    Integer a = new Integer(10);
    int i = a.intValue();
    int j = a;          // auto unboxing
    S.O.P(a+" "+i+" "+j);
}
```

Anonym. class → only used to override
15/9/24 Overriding can be done in any method.

* Java Anonymous Inner class :-

→ A class that has no name is known as anonymous inner class in java.
→ It should be used if you have to override method of class or interface.

ways to create -
① class (maybe abstract or concrete)
② Interface

eg.
```
abstract class Animal
{
    public abstract void eat();
}

class Dog
{
    p s v m ( )
    {
        Animal a = new Animal(){   → ref. variable
        {
            public void eat()
            {
                s.o.p("  ");
            }
        };
        a.eat();
    }
}
```

if no obj. / ref. variable ⟹ eat();

* Local Inner Class :- (LIC)

→ A class that is created inside a method is called local inner class.
→ We need to create obj. of that class inside method only.
→ LIC cannot be invoked from outside.

Exam.

* Static Nested Class :-

→ A static class that is created inside a class is called static nested class in java.

→ A static class can have static method.

* Exception Handling :-

□ Exception
→ It is an event that disrupts the normal flow of prog.
→ Exception means any abnormal event, unexpected event or extra-ordinary cond.ⁿ/event that may occur at runtime.
→ When exception occurs in prog., then Java environment prints some error mesg. on the screen & prog. gets terminated there only.

→ Lets take a scenerio;

```
Statement 1;
"     2;
"     3;
"     4;
"     5; // Exception occurs
"     6;
"     .
"     .
"     10;
```

Suppose there are 10 statements in our prog. & there occurs an exception at statement 5, the rest of the code will not be executed, i.e, stat. 6 - 10 will not be executed. If we perform exception handling, the rest of the statement will be executed. That is why we use EH in Java.

Types :-

- Built - in
- User - defined

* Exception Handling :-

→ means if exception occurs in your prog, handle that situation & continue the prog. execution by taking some necessary action.

Actions can be -

→ try
→ catch
→ throw
→ throws

16/9/24

Default Exception Situation :-

→ If you were trying to divide a ~~num~~ number by 0.

→ If you declare the obj. & memory is not allocated for that obj.

→ If you were trying to access the element of an array which exceed the array limit.

→ If you are not accessing the file which is not on hard disk.

→ If you are converting invalid string to a number.

* Some Java built-in exceptions: (Runtime Exc.)
(Unchecked Excep.)

(a) Arithmetic exception
It is caused by math error such as /0.

(b) Null-pointer exception
If obj. is declared but memory is not allocated.

(c) ArrayIndexOutOfBoundException
It is caused by bad array index.

(d) FileNotFound Exception
It is caused by an attempt to access a non-existing file.

(e) IOException
It is caused by general input/output failure.

(f) NumberFormatException
It is caused when a conversion b/w String & number fails.

(g) InputMismatch Exception

⇒ Exceptions
It

① Built-in
- Checked Excep.
- Unchecked Excep.

② User-defined or custom Excep.

* Exception Handling :-

→ It means if exception occur in ur prog., handle that situation & continue execution by taking some action.

→ Exception is a pre-defined class available in lang. package.

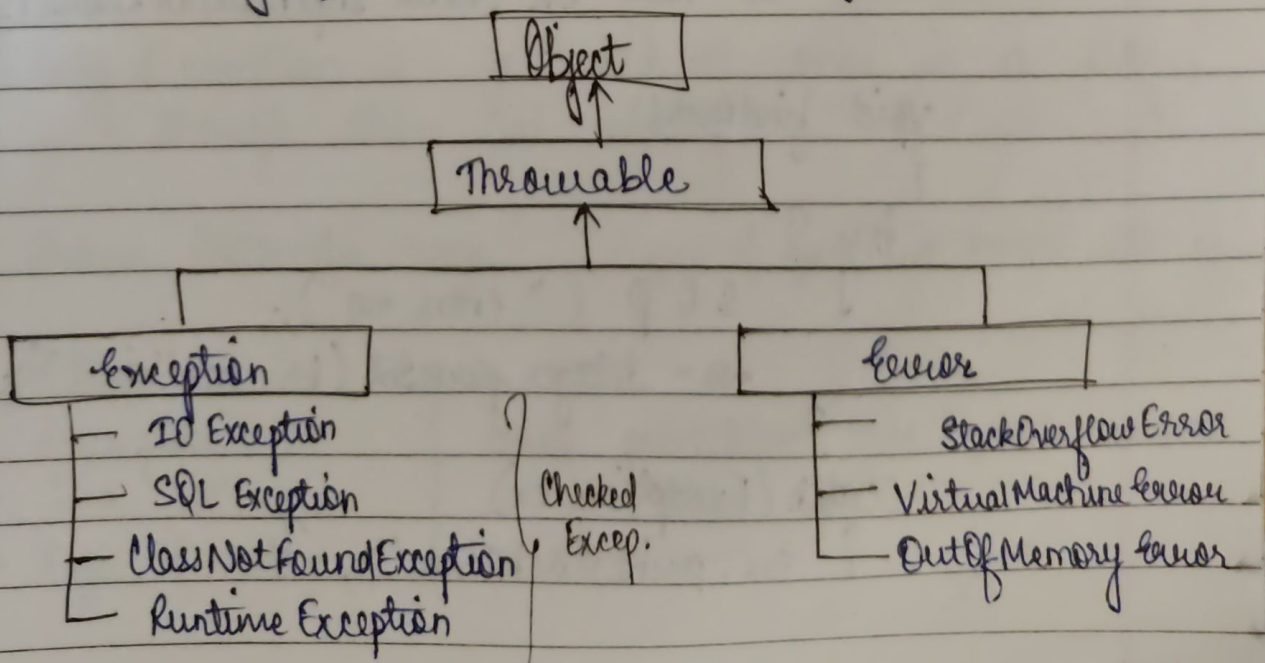
* Hierarchy of Java-Exception classes :-

→ Obj. class is parent class of all the classes in java.

→ In other words, it is the topmost class in java.

→ Throwable class is the root class of Java-Exception Hierarchy which is inherited by 2 sub-classes, i.e., 'exception' & 'errors' class.

• A hierarchy of J-E classes are given below-



* Exceptions :

(1) Built - in
which are available in Java library. They are suitable to explain certain error situations.

Acc. to Oracle (Types) - (3 else only 2)

(a) Checked
- excep. that is checked by the compiler at the compilation time.
- They cannot be simply ignored, the progammer should handle these excep.
- It forces prog. to deal with the excep. that maybe 'throws' or with the help of 'try-catch' block.

Eg. class Factorial
{
BR br = new BR (new InputStreamReader (System.in));

void findFact()
{
try {
SOP ("enter no");
no = Integer.parseInt (br.readLine());
}
catch (Exception ex)
{ ex.printStackTrace(); }

=> OR

⇒ s.o.p("Exception name");

OR,

⇒ s.o.p("Exception = " + ex.getMessage());

(b) Unchecked.

→ that occurs at the time of execution is called unchecked excep.
→ also called runtime excep.
→ are ignored at the time of compilation.
→ It is in java.lang.RuntimeException class, which is subclass of Exception class.

Keywords for handling excep.-

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

Only 1 exception is generated at once in a prog., though there are many.

Q. Discuss Arithmetic excep. (Type of ques. in exam) with eg.

NOTE :-
→ description
→ line no.
→ type of excep.

} Only printStackTrace() tells about these in O/P.

□ Multiple catch exception
One try can have multiple catch statements.

Syntax –

```
try {
    statm.;
}
catch (Exception-Type 1 ex)
{
    stat;
}
catch (Excep. Type 2 ex)
{
    stat;
}
```

20/9/24 [Excep. prog.] → V. Imp. (10 MM in exam)

Q: WAJP to raise and handle the arithmetic exception

eg.
```
P S V m ()
{
    try
    {
        S O P (3/0);
    }
    catch (Arith. Excep. ex)
    {
        S O P ("Excep : " + ex.getMessage());
    }
```

eg p s v m()
{
int b, rs = 0;
Scan. - - -
try
{
SOp("enter 1 no");
a = sc.nextInt();
SOp("enter 2 no");
b = sc.nextInt();
rs = a/b;
}
catch (AE ex){
{ ex.printstackTrace(); }
}
SOp("Result" + rs);
SO.p("Thank u");
}
}

• Multiple catch exception -

eg. Same as above

catch (AE ex)
{ ex. pST();
}
catch (InputMismatchExcep. ex)
{ ex. pST();
}
catch (ArrayIndexOutOfBoundsExcep. ex)

```
{ ex.pST();
}
catch(Excep. ex)
{
    ex.pST();
}
S.O.p("Result"+ns);
}
}
```

⇒ NOTE:-

- ek baar mei humesha ek hi exception handle hota hai.
- we are writing diff. itimes catch(), catch() to make the process fast.

Q. WAJP to raise & handle ArrayIndexOutOfBoundsExcep.

* InputMismatch → in util package

Q. WAJP ~~to~~ to raise & handle the NullPointerExcep.

□ NullPointerExcep.

If obj. is declared but memory is not allocated.

eg. Employee emp[] = new Employee[10];

for(i=0; i<10; i++)

emp[i] = new Employee() ; // Null Pointer Exc.
emp[i] = ~~read.Employee~~ ;
emp[i].readData() ;

Q. WAJP to raise & handle the NumberFormat Excep.

Caused when a conversion b/w string & number fails.

eg. import.....
class Test
{
psvm ( )
{
Int num;
try
{
num = Integer.parseInt ("Manjeet");
}
catch (NFE ex)
{
S O p (ex);
}
S.O.p ("Thank you");
}
}

⇒ NOTE:-

Manjeet cannot be converted, but S O p("10") can be converted.

* Throw :- (in java)

→ In java, 'throw' keyword is used to explicitly throw an exception.

→ We can throw "unchecked" exception in java by 'throw' keyword.

imp

→ It is mainly used to throw "custom" or user-defined exception.

imp

Syntax -

throw exception;
↳ obj.

eg.

throw new ArithmeticException ("divide by zero");

OR.

with obj.

AE obj = new AE("divide by zero");
throw obj;

eg.

```
class Test
{
    p s v m ( )
    {
        try
        {
            int a, b, as;
            Scan. . . .
            s o p ("enter 1 no");
            a = sc.nextInt();
            s o p ("2nd");
```

```
b = sc.nextInt();

if (b==0)
        throw new AE("/by zero");
        rs = a/b;
        S O P ("result:" + rs);
    S.O.P ("thank u");
}
catch (AE ex)
  {
      S O P ("Exception: " + ex.getMessage());
  }
}
}
```

⇒ Throw does not work without try & catch.

22/9/24

* ~~Custom~~

Q. WAJP to create age exception class, to raise the excep. when user input is <18 and >40.

Q. WAJP to create employee class with age as an attribute. If age <18 or >40, then throw age exception.

* NOTE:- Custom excep. (user-defined excep) typically extend exception class.

eg.

```
class AgeException extends Exception
{
    public void msg()
    {
        S.O.p("Invalid age");
    }
};

class Employee
{
    int age;
    void read(int a)
    {
        age = a;
    }

    void display()
    {
        try
        {
            if (age<18 || age>40)
                throw new AgeException();
            S.O.p("Age = "+age);
        }
        catch (AgeException ex)
        { ex.msg(); }
    }

    p s v m ( )
    {
```

```
Employee e = new Employee();
        e.readData(25);   OR.   e.readData(15);
        e.display();
    }
}
```

OR.

// to print the age even if wrong.

```
try
{ same
} catch ( ) { }
S.O.p. ("Age = " + age);
```

Q. WAJP to accept the record of employee as id, name, dept, desg, age & sal. If age < 18, or > 40, then throw age exception.

Exam.

Q. WAJP for user-defined excep. that checks the internal & external marks. If internal marks > 40, it raise the excep. "internal marks is exceeded. If external marks > 60, it raises the excep. & display the msg "external marks is exceeded". Create above excep. & use it in ur prog.

```
try
{
    if (im > 40)
        throw new InternalExcep.();
```

```
package udep;
import udep.*;
```

```
if (em > 60)
    throw new ExternalExcep();
}
catch(Excep ex) { }
}
```

NOTE:- Data / Month / Year etc. are not built-in exception. So, u need to make these exceptions.

```
Eg. class DateException extends Exception
{
    public void msg()
    {
        sop("Invalid date");
    }
};

class Employee
{
    int eid, salary;
    String name, desg, dept;
    String doj;
    String ss[];
    #
void readData ( )
{ }

void display()
{
    ss = doj.split("/");
```

# ss = doj.split("/");
ss[0] = 32
ss[1] = 01
ss[2] = 2024.

UDEP → User-defined Exception Package.

```
if (Integer.parseInt(ss[0]) < 1 || Integer.parseInt(ss[0]) > 31)
    throw DateException();
sop(id + "\t" + name + "\t" + desg + "\t");
}

psvm( )
{
    Employee e = new Employee();
    e.readData(1001, "Amit", "CS", "AP", 20000, "32/01/2024");
}
```

Q. WAJP that generate custom exception (date, year, month), if entered date is incorrect.

23/9/24

* Throws :-

→ It is used to declare exception. It does not throw an exception.
→ It specifies that there may occur an exception in the method. It is always used with method signature.
→ If any method is able to raise exception but, if it not handled there & if u want to handle that exception in calling method, then in front of method name mention

the possible exception list by using throws keyword.

V.V Imp. * Diff b/w throw & throws

	Throw	Throws
(1)	used to explicitly throw an exception.	used to declare an exception.
(2)	Checked excep. can't be propagated using throw.	Checked excep. can be propagated using throws.
(3)	It is followed by an instance (obj.)	It is followed by a class.
(4)	Used within a method.	Used with a method signature.
(5)	Can't throw multiple exceptions.	It can declare multiple exceptions.

* Finally keyword :-

→ In finally block, we can write those statements which u want to execute in both situations, exceptions generated or exception not generated.

→ Finally block should be immediately after the try block or catch block.

eg.

```
class Test
{
    p s v m ( )
    {
        int a,b;
        try
        {
            a = sc.nextInt();
            b = "
            rs = a/b;
        }
        catch (Excep. ex)
        {
            ex.pST();
        }
        finally
        {
            S.O.p ("result" + rs);
        }
    }
}
```

* Nested try blocks:-

Using try block inside another try block.

eg. Inner try block can be used to handle ArrayIndexOutOfBoundsExcep. while outer try block can handle ArithmeticExcep.

eg.
```
try
{
    try
    {
        try { }
        catch() { }
    }
    catch () { }
}
catch () { }
}
```

* Diff. b/w error & exception.

□ Exception

→ Most of the cases, exceptions are caused by our prog. & excep. are recoverable.

eg NullPointer exc. — we allocate the memory.
ArrayIndexOutOfBound — we correct the index range.

→ If our prog. req. is to read data from a file. If the file is not available, then we will get "FileNotFoundExcep.".

→ If FileNotFoundExcep. occurs, then we can provide the file & rest of the prog. can be continued normally.

Error

→ Most of the times, errors are not caused by our prog. They are due to lack of system resources.

→ Errors are not recoverable.
eg. If OutOfMemoryError occurs being a programmer, we can't do anything & the prog. will be terminated abnormally.

→ System admin or Server admin is responsible to increase the heap memory.

27/9/24

* Multithreading :-

→ Process of executing multiple threads simultaneously.

→ Thread is basically a light weight sub-process, a smallest unit of processing.

→ Multithreading & multiprocessing, both are used to achieve multitasking.

→ Multithreading is mostly used in games, animation etc.

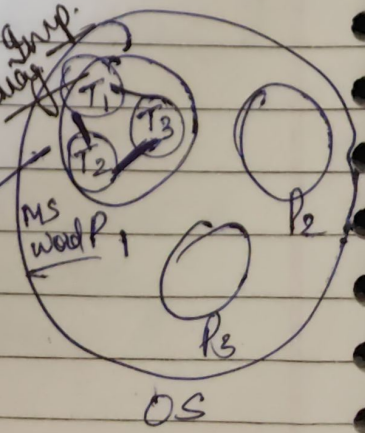
□ Advantage :

(1) Doesn't block the user because threads are independent and you can perform multiple operation at same time.

(2) We can perform many operations together, so it saves time.

(3) Threads are independent, so it doesn't affect other threads if exception occur in a single thread.

(4) Thread does not have a deadlock.



Context - switching (b/w threads)

* Multitasking:

→ Process of executing multiple tasks simult. We use it to utilize the CPU.

Two ways - (to achieve MT)

(a) Process-based MT [Used in OS Level] (Multiprocessing)
(b) Thread-based MT [" " programmed level] (Multithreading)

(a) Process-based MT
- Each process has its own ~~memory~~ address in memory

(b) Thread-based MT
- Threads share same address space.
- Thread is lightweight.
- Cost of communication b/w thread is low.

* NOTE:- At least one process is req. for each thread.
- Thread is a separate path of execution.

* How to make thread?

→ Extending thread class
→ Implementing runnable interface

- run() → heart & soul of any thread

Date
Page

Extending

→ one thread performs only one task.
→ declaring the class extending the "Thread" class.
→ Implementing the "run()" method.
→ creating a thread obj. & call the start() method to initiate the thread execution.

eg.

```
class MyThread1 extends Thread
{
    public void run()
    {
        for (int i =1; i <=10; i++)
        {
            S.O.p (2*i);
            try
            {
                Thread.sleep (1000);
            }
            catch (Exception ex)
            {
                S.O.p (ex);
            }
        }
    }
    p s v m ( )
    {
        MyThread1 t1 = new MyThread1 ();
        "  t2 = "  "
        "  t3 = "  "
```

→ it invokes run() method.

```
t1.start();
t2.start();
t3.start();
```

⇒ Every thread has a priority attached to it.

Normal priority
Minimum "
Maximum "

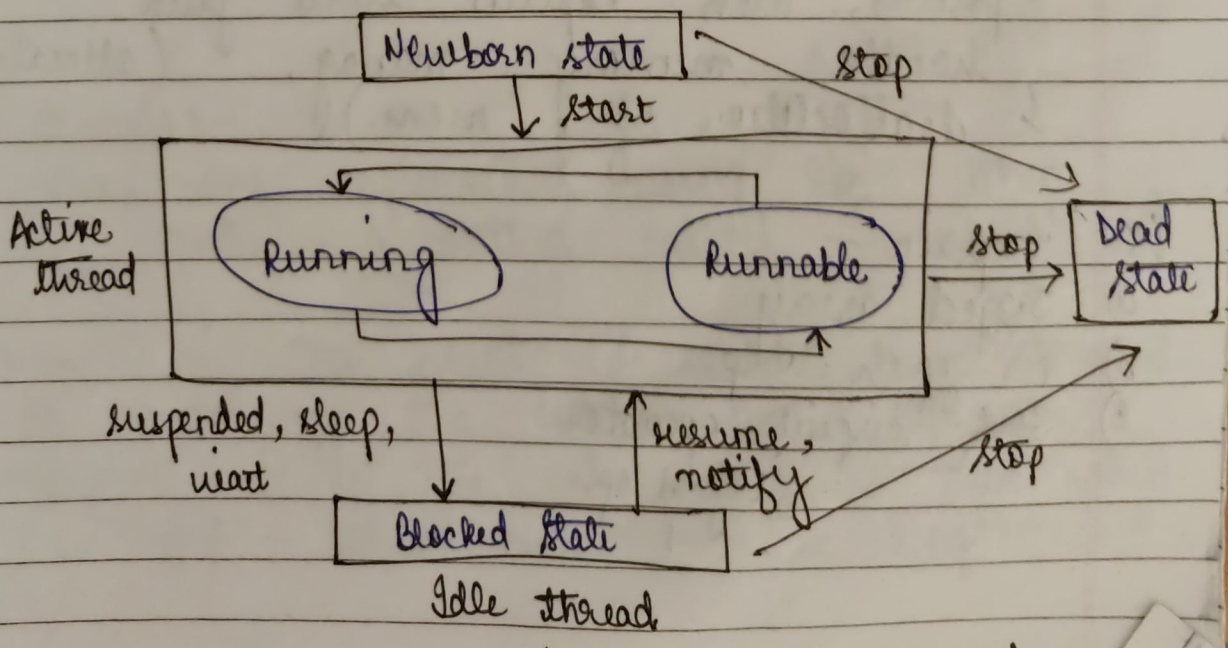
Types of priority Range (1 . . 5 . 10)
↙ ↓ ↓
Norm. Min. Max.

→ Programmer can change the priority of a thread.
Priority is given by JVM - having 5 by default.

V.V.Imp.

* Life Cycle of a Thread (Thread States) :-

It has 5 states. Acc. to Sun Micro., 4 states.



State transition diag. of a thread

- Thread schedular → used to tell / control which thread will provide to the processor.

- Time slicing

→ one thread can run only once.
→ When processor gives you time, then thread will run.

28/9/24

* Final exam questions :-

Q1. What is diff. b/w JDK & JRE.
Q2. ~~(a)~~ Explain the diff. JDK tools available in Java.
Q3. ~~(b)~~ Explain diff. types of classes available in Java, with eg. of each.
Q4. (a) Why is string in java immutable?
(b) If java does not support concept of pointer, then explain how java handle memory manag. (allocation & deallocation of mem.).

Q5. Explain -
a) Jagged array
b) For-each loop
c) ~~Sort~~ short-circuit operator
↳ or logical op.

Q6. Can u make constructor as private or protected? If yes, explain with proper eg. & if no, explain why?

Q7. What are abstract methods? You can use interfaces & abstract class for creating them. Explain both the ways & their implementation using proper eg. Also explain the scenario where u would choose one over the other.

Q8. How do u throw an exception? Can u throw multiple exception in one statement?

Q9. Why is finally block used? Give eg. for signifi.

Q10. When a catch block throws an excep., what happens?

* Threads :-

→ They are implemented using obj. that contain a method called run().

→ run() is the heart & soul of any thread. It makes up the entire body of the thread & is the only method in which thread's behaviour can be implemented.

→ A typical run() method could appear as follows:

```
public void run()
{
}
```

The run() method should be invoked by an object of the ~~concerned~~ thread. This can be allowed by creating the thread & initiating it with the help of another thread method called start() method.
There are 2 ways to create a

28/9/24
(TS)

* Thread Scheduler in Java :-

Q. WAP to create 2 threads. One will print odd no's & second thread will print even no's b/w 1 and 20.

(1) TS in java is the part of JVM that decides which thread should run.

(2) There is no guarantee that which runnable thread will be chosen to run by TS.

③ Only one thread at a time can run in a single process.

④ T.S. mainly uses preemptive or time slicing scheduling to schedule the thread.

⑤ TS assigns processor to a thread based on priority of a thread.

* Methods of Thread Class (ppt):

→ User-defined thread
→ ~~Daemon~~ Daemon thread

Q. Can we start a thread twice?
NO,
→ If you do so, illegal thread state exception is thrown.
→ In this case, thread will run once but for second time, it will throw exception.

Q. If we call run() method instead of start() method?

→ Each thread starts in a separate call stack.
→ Invoking the run() from main thread, run() goes into the current call stack.

- join() method
  waits for thread to die. Causes the currently running thread to stop executing until the thread it joins with complete its task.

29/9/24

eg. (Thread) [There is context switching in thread]

```
class DemoThread extends Thread
{
    public void run()
    {
        for (int i=1; i<=100; i++)
        {
            S.O.p(i);
            try
            {
                Thread.sleep(1000);
            }
            catch (Excep. ex)
            {
                S.O.p(ex);
            }
        }
    }
    p s v m ( )
    {
        DemoThread t1 = new DemoThread();
    }
```

□ currentThread() method :
returns reference of the currently executing thread. It is static.

eg.
```
class Demo extends Thread
{
    public void run()
    {
        sop (Thread.currentThread().getName());
    }
}
psvm()
{
    Demo t1 = new Demo();
    "  t2 = "

    t1.start();
    t2.start();
}
```

⇒ sop (Thread.currentThread().getPriority());

Tells the currently executing thread.

* Java Thread Priority in Multithreading (MT) :-

→ In a MT environment, thread schedular assigns processor to a thread based on priority of thread.

→ When we create a thread in Java, it always has some priority assigned to it.

→ It is assigned by JVM at the time of obj. creation.

→ Priority range is b/w 1 to 10.

1 – 5 – 10
(MIN) (NORM) (MAX) → Priorities (value)

→ There are 3 static variables defined in Thread class for priority as shown above.

* Priority Methods :-

① getPriority() → returns priority of given thread.

② setPriority() → to change or update the priority of a thread.

* Illegal Argument Exception → when you give thread value 0 ~~and~~ or (max. as 10 & min as 1).

* ways to write how to set priority:

→ t1.setPriority(2); or t1.setPriority(Thread.MIN_PRIORITY);

→ " (5); or " " (NORM_PRIORITY);

→ " (8); " ";

- If you write any other value, other than default, so u need to specify the number.

eg. t1.setPriority(4);

* NOTE:-

→ Default prior. of main thread is 5, but it can be changed later.

→ Def. prior. for all other threads depends on the priority of the parent thread.

(aur)
→ Some OS don't provide proper support for thread priority.

→ If 2 threads have same prior., we can't expect which will execute first. It depends on thread scheduler's algo (Round-Robin, FCFS etc.)

→ If we are using thread pr. for thread sched. then we should always keep in mind the underlying platform cond.", as it may or may not run on your system. (reason)

Q. What is synchronization? } exam ques.
Q. What is race cond.? }

30/9/24

Q. WAP to create threads. One thread will print odd no's and second thread will print even no's b/w 1-20.

Imp:

Q. Anonymous class. (that does not have name)
Purpose - to avoid the method.

* Race Condition in Java:-

→ When more than one thread try to access same resource without synchronization causes race condition.

eg Train Ticket Booking

→ Explana.→ If only 1 ticket avail., & 2 passengers are trying to book at same time without synchronization.

It might happen that both might end up booking it, though only 1 ticket was avail. which creates a prblm.

⇒ But if synchronization was there, only one of them would have been able to book.

* How to overcome race cond"?
Synchronization
↓
means to apply lock to an object.

* Synchronization :-

→ Capability to control the access of multiple threads to any shared resource.

→ It is better when we want to allow only one thread to access the shared resource.

→ Synchronization keyword allows one thread at a time to access a method variable or an obj. ~~such that making~~

* Why Synchronization?

- To prevent thread interference
- " " inconsistencies in the data

* Thread Synchr :-

→ Mutual Exclusive ← Synchronized method / " block.
→ Co-operation (Inter-thread commun.)

* Concept of lock in java :-

- Synch. is built around an internal entity known as lock or monitor.
- Every obj. has a lock associated with it.

⇒ This also raises Deadlock.

eg. class Table
{
synchronized void printTable (int no)
{
for (i=1; i<=10; i++)
{
s.o.p(no*i);
try
{
Thread.sleep(500);
}
catch (Exception ex)
{
s.o.p (ex);
}
}
}
};
class MyThread1 extends Thread
{
Table t;
public MyThread1 (Table tb1)
{
t = tb1;
}
public void run()
{
t.printTable (2);
}
};

```
class MyThread2 extends Thread
{
    Table t;
    public MyThread2 (Table tb1)
    {
        t = tb1;
    }

    void run()
    {
        t.printTable (7);
    }
};

class Test
{
    p s v m( )
    {
        Table obj = new Table ();
        MyThread1 t1 = new MyThread1 (obj);
        MyThread2 t2 = " " 2 (obj);

        t1.start();
        t2.start ();
    }
}
```

* Synchronized method :-

→ If you declare any method as synchronized, it is known as synchronized method.

→ It is used to apply lock to an object, to prevent thread interference & avoid race cond".

Q. Elaborate on adv. of using multi-threading in a prog. WATP to demo 2 threads from the same class to perform 2 different tasks.

Q. Demo a class having 2 threads : one thread to print 'A to Z' & other from 'Z to A'.

4/10/24

eg. synchronized method using anonymous class.

```
class Table
{
  synchronized void printTable (int no)
  {
    for (int i=1; i<=10; i++)
    {
      S.O.P(no*i);
      try
      {
        Thread.sleep (100);
      }
```

Q. Diff. b/w join & synchronization.

```
        catch(Exception ex)
        {
            SOP(ex);
        }
    }
}
}

class Test
{
    p s v m( )
    {
        Table obj = new Table();
        Thread t1= new Thread()
        {
            public void run()
            {
                obj.printTable(2);
            }
        };
        Thread t2 = new Thread()
        {
            public void run()
            {
                obj.printTable(7);
            }
        };
            t1.start();
            t2.start();
    }
}
```

* Synchronized block in java :-

→ Can be used to perform synchronization on any specific resource of the method.

→ In a 50 line code, if you want to lock only 5 lines, use synch. block.

→ If u put all the codes of the method in syn. block, it will work same as syn. method.

□ NOTE :-

- used to lock an obj. for any shared resource.
- Scope of syn. block is similar to that of the method

eg. Syn. block

```
class Table
{
    void printTable (int n)
    {
        synchronized (this)
        {
            for( int i=1 ; i<=5 ; i++)
            {
                sop (i), (n*i);
            }
            try
            {
                Thread.sleep(400);
            }
```

```
catch (Excep. ex)
{
    ex.pST( );
}
:
:
```

* Inter-thread communication :- (or Co-operation)

→ It is all about allowing synchronized threads to communicate with each other.

→ Cooperation is a mechanism in which a thread is paused running in its critical section & another thread is allowed to enter in same critical section to be executed.

→ It is implemented by fol. methods of obj. class -

- wait ( )
- notify ( )
- notifyAll ( )

Q. Why are they part of obj. & not thread class?

Ans. They are related to locks not threads.

eg.

```
class Customer
{
    int amount = 10000;
    synchronized void withdraw (int amt)
    {
        S.O.p ("Going to withdraw.....");
```

Q. Diff. b/w Thread & Runnable.

```
if (amt > amount)
{
sop("Less amount; waiting for deposit");
try
{
    wait();
}
catch (Excep. ex)
{
  sop(ex);
}
amount = amount - amt;
sop("Avail. balance = " + amount);
sop("withdraw completed successfully");
}

synchronized void deposit (int amt)
{
    s.o.p("Going to deposit .....");
    amount = amount + amt;
  s.o.p("Available balance after deposit:" +amount);
  s.o.p("Deposit amount successfully");

    notify();
}
}.

class Test
{
    p s v m()
    {
```

Q. Diff. b/w class & interface → provides full abstraction

```
Customer cust = new Customer();
    Thread t1 = new Thread();
    {
        public void run()
        {
            cust.withdraw(5000);
        }
    };
    Thread t2 = new Thread().
    {
        public void run()
        {
            cust.deposit(10000);
        }
    };
        t1.start();
        t2.start();
    }
}
```

(Imp)

* Implementing Runnable Interface :-

It only has run() method, which is heart & soul of thread.

http://b.org

eg.

5/10/24

```
class Table
{
  synchronized void printtable (int no)
  {
    for (int i=1; i<=10; i++)
    {
      System.out.println (no*i);
      try
      {
        Thread.sleep(1000);
      }
      catch (Exception ex)
      {
        System.out.println (ex);
      }
    }
  }
}
class MyThread1 extends Thread
{
  Table t;
  public MyThread1 (Table tb1)
  {
    t = tb1;
  }
  public void run()
  {
    t.printtable (2);
  }
}
```

```
class MyThread2 extends Thread
{
    Table t;
    public MyThread2 (Table tb2)
    {
        t = tb2;
    }
    public void run()
    {
        t.printtable (7);
    }
}

public class test
{
    public static void main (String args[])
    {
        Table s = new Table ();
        MyThread1 t1 = new MyThread1(s);
        MyThread2 t2 = new MyThread2(s);

        t1.setPriority (Thread.MAX_PRIORITY);
        t2.setPriority (Thread.MIN_PRIORITY);

        t1.start ();
        t2.start ();
    }
}
```

Manjunath
5/10/24

5/10/24

* Implementing the runnable interface :-

Runnable interface declares run() method that is required for implementing thread in our prog. To do this, we must perform the steps listed below -

(i) Declare class as implementing the runnable interface.
(ii) Implement the run() method.
(iii) Create a thread by defining an obj., i.e., instantiated from this runnable class as the target of the thread.
(iv) Call the thread start method to run the thread.

Imp: Thread & Runnable diff. (ppt)

eg. by runnable interface

```
class Odd implements Runnable
{
    public void run()
    {
        for(i=1; i<=20; i=i+2)
        {
            S.O.p(i);
        }
    }
}
```

```
class Test
{
    p s v m ( )
    {
        Add obj = new Add ();
        Thread t1 = new Thread (obj);
        Thread t2 = new Thread (obj);
        t1.start();
        t2.start();
    }
}
```

Q. Create a prog. using runnable interface.

Thread Class	Runnable Interface
① Thread is a class. It is used to create a thread.	Runnable is a functional interface used to create a thread.
② Thread has multiple methods including start, stop, getPriority, setPriority etc.	It has only abstract method, i.e, run() method.
③ Each thread create a unique obj. & get associated with it.	Multiple threads share the same obj.
④ Multiple inheritance is not allowed in java. Hence, after a class extends thread class,	If a class implementing Runnable interface, then your class can

it cannot extend another class.	extend another class.

* Daemon Thread (ppt):-

- It has lower priority & runs in bg.
  eg. garbage - collection.

  boolean < true / false

- This thread is always set before starting.

Exam ques.-

Q. How many types of thread are there in java?
Q. What is Deadlock? (in Java) Explain with eg.
Q. How to prevent DL in java?
   Lock Time - out → Good to apply. (better than others)

join method	synchronized method
→ puts implicit lock in thread obj.	used by putting lock on obj. they work upon.
→ used for sequencing of threads. eg. if we call t1 join,	not used for sequencing. Rather, they are used for synchronous access to

the next thread will join after t1 completes.	an Obj.
→ Running a prog. that uses join method will make sure threads join the order they are joined.	They ensure that threads run in a synchronous manner. They don't ensure sequence of threads.

6/10/24

* Database → Collection of inter-related data for any organization.

OLTP → DBMS
OLAP → Big data
} Connectivity

SQL → Interactive SQL , PL SQL

Driver → (mediator) converter [It is an intermediate S/W]

import java.sql.Connection;
import java.sql.*;
} APIs (at application layer)

- JDBC is an API, used to store data permanently into DB.

→ Whenever ClassNotFoundException is displayed, it means no driver is present.

→ One page can be connected with multiple DB.

→ Check driver before writing DB connectivity code for your project.

→ Driver should be connected to your project.

• Steps for DB connectivity – 5 steps (V. Imp.)

* Code for DB Connectivity –

□ Create table Employee –

eid	number(4)
name	varchar
desg	varchar
salary	number(7)

number (7,2) → double
number (7) → integer
} forms.

```
import java.sql.*;

eg. class Emp
{
    void view ()
    {
    try {
Step 1 ← Class.forName ("driver class");   // url of driver class.
Class ← Connection conn = DriverManager.getConnection
                ("db url", "uid", "pwd");

    Statement st = conn.createStatement ();
    ... → continued
```

[DataSet is table having unique records.]

(Select)

Executing Queries → executeQuery();
→ executeUpdate();
(remaining other queries)

→ continued

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

next method → moves cursor to next record.

eg while(rs.next())

Till the time it gets records, it keeps on printing them, & asap the cond.n is false, it moves out.

```
eg. while (rs.next())
    {
        sop(rs.getInt(1) + "\t" + rs.getString(2));
    }
    conn.commit();
    conn.close();
}
catch (Excep. ex)
    { sop(ex); }
}

psvm( )
{
    Emp e = new Emp();
    e.view();
}
}
```

```
public class NewClass
{
    p s v m (String args[])
    {
        try {
            Class.forName("org.apache.derby.jdbc.ClientDriver");
            Connection con = DriverManager.getConnection
("jdbc:derby://localhost:1527/empDB", "ab", "ab90");
            Statement st = con.createStatement();
            ResultSet rs = st.executeQuery("select * from EMP");
            while (rs.next())
            {
                s.o.p (rs.getInt("ID") + "\t" + rs.getString
                                                ("Name"));
            }
            con.close();
        }
        catch (Excep ex)
        {
            ex.pST();
        }
    }
```

* NOTE:-

for integer → "+ +"

for string → "' + +' "

next() → reads record one by one.

7/10/24 [Only select Query]

* login

```
import java.sql.*;
import java.io.*;

class login
{
    String userid, upwd;
    BR br = new BR ( );

    void login()
    {
      try{
          String query;
      sop("enter user id");
      userid = br.readline();
      sop("enter password");
      upwd = br.readline();
      query = "select * from emp where
          uid = '"+userid+"' and pwd = '"+upwd+"'";

      class.forName("driver class");
    Connection con = DriverManager.getConnection("dBURL", "uid",
                                                 "pwd");

    Statement st = conn.createStatement();
    ResultSet rs = st.executeQuery(query);
        boolean status = rs.next();

          if(status)
          {
          Employee emp = new Employee();    emp.view();
```

Both are same → rs.getInt(1) or rs.getInt("id");
⇒ attribute name & indexing both can be written.

```
else{
   sop("Invalid uid & pwd");
}
   catch(Excep. ex)
   {  sop(ex.pST);
   }
}
   p s v m( )
   {
      login.login();
   }
}
```

11/10/24

Q. WAJP for inserting a record into the table with the fol. attrib. -
emp → empid, ename, desg, dept, salary.

```
import java.io.*;
import java.sql.*;

class Emp
{
   int id, sal;
   String ename, desg, dept;
   BR br = new BR( );

   void insert ()
   {
    try
    {
```

```
sop("enter emp id");
id = Integer.parseInt(br.readline());
sop("enter name");
ename = br.readline();
sop("enter desig.");
desg = br.readline();
sop("enter dept");
dept = br.readline();
// storing data in db.
```

(#) query = "insert into emp values ("+id+",'"+name+"','"+desg+"','"+dept+"',"+sal+")";

query += ;

// for nextline (good Method)

```
class.forName("driver class");
Connection con = DriverManager.getConnection("db url",
                                              "uid","pwd");
Statement st = con.createStatement();
// will return value >=0
int x = st.executeUpdate(query)
```

(#) can directly write here, but avoid for neat code.

```
if (x>0)
{
  sop("Record inserted successfully"); // for message
  viewEmpDetails(); // for viewing
}
else
{ sop("Error in Inserting Record");
}
```

```
con.commit();
con.close();
} catch (Excep. exe)
{ sop (exe.printStackTrace());
}
}

// main method
p s v m ( )
{
  Emp e = new Emp();
  e.insert();
}
}
```

Q. WAJP to update the emp. sal/desg/name on the basis of empid.
(on basis of Primary key).

```
class Emp.
{
  int id, sal;
  BR br = new BR...( );

  void updatenamebyID()
  {
  try{
    SOP("enter id");
    id = Integer.parseInt(br.readline());
    SOP("enter name to be updated");
    name = br.readline();
    query = "Update emp setName" + Name + " where
```

```
eid = " + id + " ";

void dBConnectivity ()
{
  class.forName = ("driver class");
  Connection con = DriverManager.getConnection("dB url",
                                    "uid", "pwd");
  Statement st = con.createStatement ();
  int x = ....

  if (x>0)
{ sop("Record updated successfully");
   view EmpRecord();
}
 else
 { sop("error in updating record");
 }
  con.commit();
  con.close();
} catch (Excep. ex)
  { ex.pST();
  }

p s v m ()
{
    Emp e = new Emp();
    e.updateEmpRecordbyID();
```

~~JA~~

JAVA (continued...)

11|10|24

Q. WAJP to delete the empRecord on basis of id.

(##) Same.

```
void delete()
{
  try{
     sop("enter emp id");
     id = Integer.parseInt(br.readLine());
  }

query= "delete from emp where eid = "+id+";

dB connectivity()
 Statement st = ...
    int x = ....

  if (x>0)
  {
    sop("Deleted successfully");
       viewEmpRecord();
  }
  else
  { sop("error in updating record");
  }
  con.commit();
  con.close();
}
```

```
catch (Excep ex)
{   sop (ex.pST());
}

p  s  v  m()
{
    Emp e = new Emp();
    e.delete();
}
```

1. WAJP to create Table in db using Java code.

(#) 1 [Same]

```
// Making in Derby db
   ∴ using its syntax.

void createTable()
{
 try {
 query = " create Table Student (rno numeric(4),
  Primary Key, sname varchar(100),
 course varchar(100), fee numeric(7));

 class.forName("driver class");
 Connection con = DriverManager.getConnection
          ("db url", "id", "pwd");

 Statement st = . . .
    int x= . . .
```

```
if (x==0)
    sop("Table created successfully");
else
    sop("Table already created");
}
con.commit();
con.close();
}
catch(Exeop. ex)
{
  ex.pST();
}
}
psvm()
{
}
```

Q. Create table emp with fol. attribute eid, ename, age, gender, sal, address, city & state.
Write a menu driven prog. to perform various operations such as insert emp. record, update & delete record from emp table on the basis of eid, & also display the entire emp. record from the emp. table.

18/10/24

Starting version 9 ; SQL → internal

[Parameterized]

```
class Emp
{
  int eid, sal;
  String name, desg, dept;
  BR br = new BR(new ISR(System.in));

  void insert()
  {
    try{
      sop("enter id");
      eid = Integer.parseInt(br.readline());
      sop("enter name");
      name = br.readline();
query = "insert into emp values(?,?,?,?,?)";

  class.forName("driver class");
Connection con = DriverManager.getConnection("dburl",
                                  "uid", "pwd");
PreparedStatement pst = con.prepareStatement(query);
      pst.setInt(1, eid);
      pst.setString(2, name);
      pst.setString(3, desg);
      pst.setString(4, dept);
      pst.setInt(5, sal);

  int x = pst.executeUpdate();

    if(x>0)
      sop("insert record successfully");
    else
      sop("error in inserting record");
```

```
con.commit();
con.close();
}
catch (Excep. ex)
{ ex.pST();
}
}

p s v m ( )
{
    Emp e = new Emp();
    e.insert();
}
}
```

* NOTE (imp.) :-

→ The basic createStatement method returns statement obj. that can be used to send SQL (query) to the db for execution.

→ Every SQL u send will get validated, compiled & executed.

→ With prepareStatement method, call u pass in SQL once where it can be validated & compiled.

→ It will throw an excep. if the SQL is syntactically invalid or reference table or columns that does not exist or if no of parameter that does not match with no of values specified.

PreparedStatement pst = con.prepareStatement
("insert into int values ?, ?);

→ With this prepared statement, u can do only 1 thing → insert record into emp table.

→ U have now told the db what u want to do, but not what value to insert.

→ This comes with the 2nd set of statement

```
pst.setInt(1, eid);
pst.setString(2, name);
pst.update( );
```

OR

```
pst.setString
(2, "Anu");
```

↓

This is static so should not be used.

Q. WAJP to update the name of emp on the basis of ID.

```
class Emp
{
    int eid, sal;
    String name, desg, dept, query;
    BR bf = . . .

    void updateempNamebyID()
    {
```

try{
sop("id");
id .....
sop
name = ...
} same.

query = "Update emp set name = ? where eid = ?";

class.forName("driver class");

Connection con = ... same.

PreparedStatement pst = con.prepareStatement(query);

pst.setString(1, name);
pst.setInt(2, eid);

:
same

main → same.

Q. To delete emp record by ID.

same.

void deleteempbyID().
{
same
query = "delete from emp where eid = "+eid+";
pst.setInt(1,eid); ..... same

* NOTE:-
It is not compulsory that Prepared Statement only has ?. It can also have values.

* SQL Injection :- (ppt)
V.V. Imp.

* Diff. b/w Statement () & PreparedStatement () [ppt.].

* Why u need driver?
We cannot talk to db directly.

* Types of drivers → 4 types of JDBC drivers

Imp. Today used driver → Type - 4 [Thin driver]

V. Imp. exam. Discuss diff. type of drivers.

14/10/24

Course	Student
cid - P.K.	rno - P.K.
cname	name
fee	dept
duration	cid - F.K.

String query = "Select rno, name, dept, cname, fee, duration from course, Student where Student.cid = course.cid";

for plug in → setBounds() method is used.

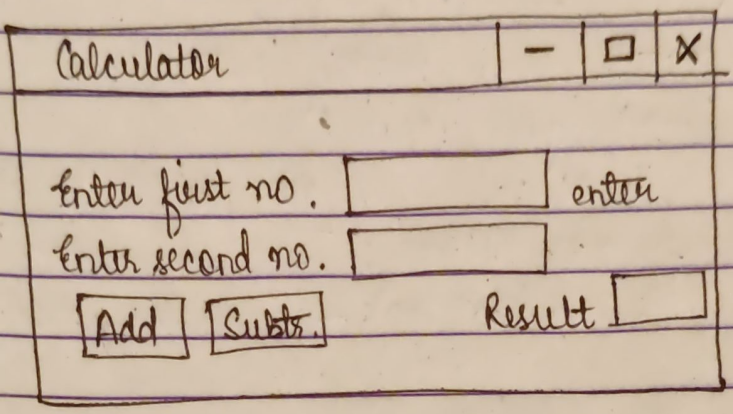
14/10/24

* GUI or Window-based applic.n :- (ppt) [AWT]

Radio button → Select one item
List → Select multiple item

* Panel : It is a class.

eg.



Panel → does not have title bar & menu bar
Frame → has " " "

AWT is an API.
↳ has buttons, checkboxes, etc.

* Prog. for Calculator :- (by extending frame class)

```
import java.awt.*;

class Calculator extends Frame
{
    Label lbl.fno, lbl.sno, lblres;   // fno → first no, sno → 2nd, res → result
    TextField txtfno, txtsno, txtres;
    Button btnAdd;
    Panel p1;
```

public Calculator()

or

public void Calculator()

```
{
    lblfno = new label ("enter first no.");
    lblsno = new label ("enter second no.");
    lblres = new label ("Result:");
    txtfno = new Textfield (20);
    txtsno = new Textfield (20);
    txtres = new Textfield (20);
    btnAdd = new Button ("ADD");
      p1 = new Panel ();
    add (p1);          // add panel to frame
    p1. add (lblfno);  ] // add control to
    p1. add (txtfno);  ]    the panel

    p1. add (lblsno);
    p1. add (txtsno);

    p1. add (lblres);
    p1. add (txtres);

    p1. add (btnAdd);
}

p s v m ( )
{
    Calculator calc = new Calculator ();
Imp.←  calc. setSize (500, 600);
    calc. setTitle ("Calculator"); // for frame title
    calc. setLocation (100, 100);
Imp.←  calc. setVisible (true);
}
}
```

To convert → enter 1st no [10]

int a = Integer.parseInt (txtfno.
getText ());

↙
To retrieve data

→ Put null along with setBounds().

18/10/24

* To set label or caption :-

It cannot be changed except for at run-time.

eg label lblfno;

lblfno = new label ();

* Button class :-

setlabel → to set label or caption to button.

eg Button b1;
b1 = new Button ();
b1.setlabel ("OK");
→ to set label

[OK]

String s = b1.getlabel (); → to get label.
b1.getlabel ("OK");
↙
prints [OK]

Static → cannot be changed at runtime

* Drop-down list :-

eg. Course : [MCA ▽]
MCA
MA
B.Tech.
→ Static

Button b1, b2, b3;

After choice (Syntax in ppt) Pg 10
c1. add (C1);

String cname = dropcourse.getSelectedItem();
↓
very imp.

We can insert at run time as well using insert().

* Checkbox :- Red Green Blue

Checkbox cR, cG, cB;

cB = new Checkbox("Blue");

Insert.

setState() → for true or false.

eg.
```
if{ g.setState(false);
    b.   "    "
}
else if
    { b.   "
      b1.  "
    }
  else
    { g.
      b.
    }
```

* Radio Button :-

radio Gender

Gender: | Male O Female O Others O |

→ Checkbox
→ Checkbox group

eg. CheckboxGroup cg; ← name of checkbox group
Checkbox cMale = new Checkbox ("Male", cg, true);

* Login Page :-

enter user id : [ ] ← txtuid

[ ] ← txtpwd

[Sign in]

select * from login where
uid = '" + txtuid.getText() + "'

OR

String uid = txtuid.getText();
" pwd = txtpwd. "

select * from login where uid = '" + uid + "'

19/10/24

* To change font -

Font f = new Font( );

* For bg or fg color -

lbl.setBackground(Color.red);
" Foreground(" .blue);

* For font style -

Font f = new Font("Arial", Font.BOLD, 20);

Date
Page

* Create this form:-

Registeration Form in Windows Form: