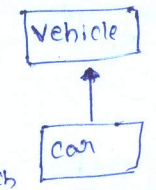
- Upcasting automatic conversion occurs.
  byte → char → int → long → float → double
  (upcasting) from left → Right
- downcasting explicit casting is reqd.
  (downcasting from Right → left
  eg int i = 5
  long j = i (upcast)
  byte b1 = (byte) i (downcaste)

```
Veh v1 = new Car(); //up
Veh v2 = new Veh();
Car c0 = v1  // wrong type mismatch
Car c1 = (Car) v1;
Car c2 = (Car)v2;  // class cast Excep
```



[Concentrate prev]

v2 no knowledge of Car

The classCastException thrown means indicate that code has attempted to cast an obj to a subclass of which it is not an instance (ie bad obj)

To check (or) eliminate these exceptions use
- ClassCastException
- instance of

eg catch (CCE){ }
if (v2 instance of Car)
  Car c2 = (Car)v2;

- converting Premitive datatype to an obj is called "Boxing".
  eg Char ch = 'A'
  Character obj = new Character(ch);
  - Char x = obj.charValue(); // unBoxing.
- Singleton:
  Is a class that can be instantiated only one time in JVM per class loader.
  eg Public class onlyone
  {
  Private static onlyone obj = new Onlyone();
  Private onlyone() { = } // private construct
  Public static onlyone getInstance()
  { return obj; }
  }
  TO use: onlyone m = onlyone.getInstance(); (static)
- diff b/w Singleton class & static class.
  Static class is one approach to make a class singleton by declaring the class as "final" so that it can't be extended & declaring all the methods as static so that you can't create any instance of class and can call the static methods directly.

- can abstract class have constructors?
  Yes. eg inet sol
- abstract class can contain instance vars & concrete methods in addition to abstract classes.
- All methods in interface are Public & abstract by default
- Interface can contain vars which are contain Public, static & final. constants only
- An interface can't implement another interface.
- An interface can extend another interface.
- we can write a class with in interface
- why Char arr is better than String to store Pwd?
  String are immutable. It saved in String Pool, there is a high chance that it will be retain in memory long time. Any one who can access memoryDump can find Pwd in clear text. If u want reset Pwd again a new String obj would be create in Pool. Using char[] you can still set all his elements as blank (or) zero.
  - Java team bon Pwd field defined return Char[] & deprecated getText().
  eg String Pwd = "TestPwd"; Sop(pwd); o/p TestPwd
  Char[] Pwd = new Char[]{'a','b','c'}; Sop(Pwd); o/p Addresss
- String literal.intern() method allows to Put an String Obj in String Pool.
  eg String str = new String("test String");
  ~~Sop(str.intern());~~
  Sop(str.intern());

- Shallow cloning :- any modifications to original obj will also effect the cloned obj. and viceversa.
- Deep cloning :- modifications to the original obj will not effect cloned obj.
  eg class Emp implements cloneable
  {
  Emp(id, name) { id=id; name=name; }
  void disp() { Sop(id + " " + name); }
  Object myclone() throws CloneNotSupportedException
  { return super.clone(); }
  }
  class clonedemo
  {
  P S V m(S a[]) throws CNSE
  {
  Emp e1 = new Emp(10, "A"); e1.disp();
  Emp e2 = (Emp) e1.myclone(); e2.disp();
  }
  }

Static :- Static method can't read instance vars.
Static method can't be overriden in java.
Static vars not Serializable.

- How you call a web server from a stand alone java?

  Using the ~~javax.xxx~~, java.net.URLConnection and its subclasses like HttpURLConnection and JarURLConnection.

- objs saved in heap
  primitive vars saved in stack, if they are local method vars, saved in heap if they are class member vars.

- Public class Test
```
{
    P S V m (S ar[])
    }
        Final class Cons
        { Public static String name = "Hi"; }
    }
}
```
Q) Is it will compile?

Ans No, you can only use static vars in static or top level classes. In the above example we have got a static var in a class that is not top level.

Core Java

1. Types of class loader?

Ans Static class Loader
Dynamic class loader.

Static	Dynamic
classes are statically loaded with java's new operator.	Is a technique for programmer invoking the fns of class loader at run time.
eg Car c = new Car();	eg Class c = Class.forName("clName"); Car obj = (Car) c.newInstance();

- Private constructor is used if you don't want other classes to instantiate the Obj. The instantiation is done by a public static method of same class.
  - Used in Singleton pattern
  - Factory method pattern
  - Utility classes eg "StringUtils" etc.

- StringBuffer vs StringBuilder (1.5 onwards)
  methods are Syn — not Synchronized.

- why String is final in java?
  Security, optimization, to maintain String pool.

- C string is a null terminated char array. But java Str is obj

- why there is no global vars in java?
  These are globally accessable & hence create a collision in namespace.

- == operator, to compare Premitive values only.
  equals(), to compare Objects.

Widening:- mean adding of Sub class Obj to Superclass ref.

A ref;
ref = (A) new B();
OR
ref = new B();

ref.show2() //error

- In widening all Superclass members only accessible you can't access Subclass methods unless they override Superclass methods.

Narrowing: Using Subclass method, access all members & methods of both Super & Sub classes.

One O; O = new Two();
Two t = (Two) O; t.show1(); t.show2();

- How to know Str args[] are empty or Null?
  Print arg.length(); it prints 0 → Empty not Null

Core Java

- why multiple inheritance not supported in java.
  - Ambiguity around diamond problem
  - It does complicate the design & creates problem during casting, constr chaining etc.

- Diff b/w String s = "hello";
  String s = new String("hello");
  - String with new() is created in heap and not added into String pool. while String created using literal are created in String pool itself which exists in perm area of heap.

- can abstract classes contains main()? Ans Yes, Static

- Is it necessary for abstract class to have abstract method()? Ans No.

- Can abstract class have static methods?
  Yes.

- can abstract class implements interface? does they require to implement all methods?
  - Yes, No - not reqd to implement all methods.

- when a con is created, it is in auto-commit mode. means each sql stmt treated as a tr & will be automa commited at after it is executed.

```
con.setAutoCommit(false);
```

JDBC

1. Driver drv = new oracle.jdbc.driver.OracleDriver();
   DriverManager.registerDriver(drv);
2. Connection con = Dr.getConnection
   ("jdbc:oracle:thin:@locho:1521:xe",u,p);
3. Statement stmt = con.createStatement();
4. ResultSet rs = stmt.executeQuery();
5. while(rs.next())
   { ≡ }

- Scrollable RS:

```
con.createStatement
( ResultSet.TYPE_SCROLL_INSENSITIVE,
  ResultSet.CONCUR_UPDATABLE);
```

- Prepared Stmt:-

3. PreparedStatement pstmt =
   con.preparedStatement(vSQL);

Checked

Exception, IOExcep
FileNOTFound
Class not found
Clone not supported
Interrupted Excep

Unchecked

ArrayIndexOutOfBounds
NullPointer,
ClassCast,
StackOverflowError.
NOClassDefFoundError
NumberFormatExcep.

Exception

- Exception handling means prevents the IoM even when exception are occur. It doesn't mean removing the exception.
- Throwable is a class that represents all exceptions and errors.
- The exceptions detected by java compiler are called checked Exceptions.
- The exceptions detected only at run time by JVM are called "unchecked" Exceptions.
- Exception is an error that can be hadled by the programmer. Error is an error but not handled. And error can't be guessed but exception can
- throws: is useful to throw an exception out of a method without handling it.
- throw: to throw an exception obj & also handle it. It generally used in try block.

Callable Stmt :-

To call procedure

3. CallableStatement cstmt = con.prepareCall ("{call procedure}");
4. cstmt.execute();

eg
```
create or replace procedure proceg
(pone IN Number, ptwo IN Number)
BEGIN
    insert into table values (pone, ptwo);
END;
```

- con.prepareCall ("{call proceg(?,?)}");
  cstmt.setInt(1,100);
  cstmt.setInt(2,200);
  cstmt.execute();

OutParameter
cstmt.registerOutParameter(3, Types.INTEGER);
cstmt.execute();

To call function:
- return value as bn

3. con.prepareCall ("{? = callbunName(?,?)}");
4. cstmt.registerOutparameter(1, Types.INTEGER);
   cstmt.execute();
   ≡ SOP (cstmt.getInt(1));

- ResultSet holds the data and we can traverse records from top to down or viceversa even after loss of db connection.
- Batch updates: stmt objs can be grouped into a batch & executed at once.
  2. stmt = con.createStatement();
  3. stmt.addBatch ("insert into a values(1)");
     stmt.addBatch ("insert into a values(2)");
     ....
     int[] countInserts = stmt.executeBatch();
- TYPE_SCROLL_SENSITIVE : can scroll frwd & backword and sensitive to the changes made in the db.
- TYPE_SCROLL_INSENSITIVE: can scroll frwd & backw but insensitive to the db changes; It is actually snapshot of db so once data fetched not changed.
- TYPE_FORWORD_ONLY: can move only in frwd dir
- ResultSet treat like a cursor to a db. if the conn is alive then only we can retrive the info thru rs obj
- Rowset we can retrive data thru rowset obj without depending upon the conn is alive or NOT.

eg Throws:

NO TRY BLOCK
NOT HANDLED EXCE

```
class Sample
{
    void accept() throws IOException
    {
        String name = br.readLine();
    }
}
class Ext
{
    psvm(S arr[]) throws IOException
    {
        new Sample().accept();
    }
}
```

eg Throw:

we can keep method name as class name.

```
Class Demo
{
    static void Demo()
    { try {
            throw new NullPointerException("my");
      }
      catch (NullPointerException e)
        { SOP(e); }
    }
    psvm(S arr[])
    { Demo.Demo(); }
}
```

- User defined Exception :-
  1. class myException extends Exception.
  2. public myException() { super(); }
  3. public myException(String s) { super(s); }
  4. throw new myException("its my error");
- To declare an exception:
  public myMethod() throws myException
  { = }
- Is the program runs successfully?

```
main()
{ try { throw new NullPointerException(); }
  catch (Exception e)
    { e.printStackTrace(); }
  catch (NullPointerException e)
    { e.printStackTrace(); }
}
```

O/P Compilation Error. says NullPointerExcep already caught.

If you interchange catch positions it will work/runs no errors.

- If Superclass method doesn't declare Excep.
  Subclass overriden method can't declare the checked excep. but it can declare unchecked excep.
- If Superclass method declares an excep.
  Subclass overriden method can declare same, subclass excep (or) no excep but can't be declare parent excep. this rule applicable for only checked excps.

String tokenizer:-

```
StringTokenizer = new ST (String, delemeter);
String str = "This is a test msg";   (str, " ");
while (St.hasMoreTokens())
  {  String s = St.nextToken();
     Sop(s);
  }
```

- ✓ Fail fast - Fail Save iterator related to concurrent HM is Sync related. Fail fast iterator fail as soon as they realized that structure of collection has been changed since iteration has been begun.
  eg Map country = new HashMap(); ⇒ Fail fast
  = new ConcurrentHashMap() → Fail Safe
- ✓ Iterator <String> it = country.keySet().iterator();
- ✓ Generics provide stronger type safety. Allow types to be passed as parameter to class, interface & method.
  List <Employee> emps = new AL <Employee>();
  benifit is need not to cast obj we get from collection.
  Employee e = emps.get(0);

Collection

Collection - Interface ①
Collections - class

ArrayList / HashMap	Vector / HashTable
1. NOT Synchronized	1. Synchronised.

- ✓ How to Synch AL (or) HashMap.
  Map mymap = Collections.SynchronizedMap(mymap)
  List ml = Collections.SynchronizedList(ml);
- ✓ Set - unique elements, prevents duplicate, unordered data — eg HashSet, TreeSet
- List - Ordered data, - allow duplicates — eg AL, Vector, LL
- ✓ Map - allow duplicate values, Keys must be unique.

Serialization

- ✓ Is a process of reading & writing an obj
- ✓ Saving obj state to a seq of bytes as well as a process of rebuilding those bytes to live obj. - have to implement Serializable inter
- ✓ Transient var can't be serialized.
  eg file ptr & db connection obj e should be tran
- ✓ Usage: ✓ To send obj over the n/w
  ✓ if the state of obj needs to be persisted to a flat file or db
  ✓ doing deep cloning or copy obj
  ✓ objs stored in HTTP should be Serializable.
  ✓ Objs passed in RMI across n/w using Seriali
- ✓ Static vars also not Serializable. bcos these are not part of any state.
- ✓ Base class fields only handled, if base class itself is Serializ.

- Iterator — all are interfaces
  ListIterator (bi directional)
  Enumeration
  enhance for loop

Eg
```
Iterator it = al.iterator();
while (it.hasNext())
    SOP (it.next());
```

```
ListIterator lit = v.listIterator();
while (lit.hasNext())
    SOP (lit.next());
while (lit.hasPrevious())
    SOP (lit.previous());
```

```
Enumeration e = ht.keys();
while (e.hasMoreElements())
    SOP (e.nextElement());
```

```
for(Map.Entry<Integer, Object> ent : ht.entrySet())
    SOP (ent.getKey() + ... + ent.getValue());
```

- `ArrayList<E> al = new ArrayList<E>();`
  `= new ArrayList<E>(int);`
- `Vector<E> v = new Vector<E>();`
- `Hashtable<K,V> ht = new Hashtable<K,V>();`
- In order to use any Obj as key in hashmap, it must implements equal & hashmap method.
- Array of objs : Arrays.sort();
  List of objs : Collections.sort(); in this case first List convert into Arrays later do sort. Performance wise both are same. but collections.sort takes much time bcoz as it convert to array first.
- Collection classes provide random access are:
  ArrayList, HashMap, TreeMap, HashTable.
- Enumeration VS Iteration:
  Enu is twice as fast as Iterator; uses less memory. But Iterator much safer, bcoz always it's denies other threads to modify the coll obj which is being iterated by it. using iterator we can remove element but with enum not possible.
- How to Sort List in reverse Order?
```
List l = new ArrayList();
Comparator comp = Collections.reverseOrder();
Collections.sort(l, comp);
```

Eg Reading & writing obj to file.

```
Public class Emp implements java.io.Serializable
{                                   ⇒ employee class
  Public static vm cs arc()
  {
    Emp e = new Emp();                          serialization
    e.name = "abc";
    e.addr = "xyz";
    try
      {
        FileOutputStream fout = new FOS("a.ser");
        ObjectOutputStream oout = new OOS(fout);
        oout.writeObject(e);
        oout.close();
        fout.close();
      }
    Catch (IOE i) { i.printStackTrace(); }
  }
}
```

```
Emp e = null;                                   deserialization
try {
    FIS fin = new FIS("a.ser");
    OIS oin = new OIS(fin);
    e = (Emp) oin.readObject();
    oin.close();
    fin.close();
    } catch (Exception e) { e.PST(); }
```

- Difference b/w Serialization & Externalizable.
  - Extern provides us writeExternal(), readExtern() method which gives us flexibility to control java seri. machanism instead of rely on java's default seri. Correct implementation of Extern Ser can improve appl. performance drastically.
- Suppose Superclass of a new class implement Seri, how can you avoid new class to being Serialized.
  - To avoid you need to implement writeObj() and readObject() methods in your class and throw NotSerializableException from those methods. this is another benifit of customizing java Seri process.

One-To-One - Bidirectional:-

mapped by attr should be same var which defined in Person class.

```
class Pdetails
{
  @OneToOne(mappedby = "pdetails", cascade = CasType.ALL)
  Private int zip;
}
```

One-To-Many:- & Many-To-One

```
class College
{
  Private int collegeid; collegeName;
  @OneToMany(targetEntity = "Student.class", mappedby: "college",
  Private List<Student> studs;          cascade = , fetch = )
}
class Student
{
  Studentid; StudentName;
  @ManyToOne
  @ JoinColumn(name = "college-id")  => This att must
  Private College college;
}
```

Many-To-Many:- new table

```
class Event
{ erenid; eventname;
  @ManyToMany
  @ JoinTable(name = "join_delegete_erent",
        @JoinColumns = {@JoinColumn(name="eventid")},
        @InverseJoinColumns = {@JoinColum(name="delegeteid")})
  Private List<Delegete> delegates = new AL<Delegete>();
}
class Delegete
{
  delegeteid; delegateName;
  @ M-M @ JoinTable(name = "same as above",
        @JoinColumns = {@JC(name="delegeteid")},
        @InverseJoinCols = {@JC(name="eventid")})
  Private List<Event> events = new AL<Event>();
}
```

Hibernate

```
cg main {
  AnnotationConfiguration con = new AC();
  Con.addAnnotatedclass(Emp.class);
  con.configure("hib-cfg.xml");
  SessionFactory fac = conf.buildSessionFactory();
  Session ses = fac.getCurrentSession();
  Ses.beginTransaction();
      Emp e = new Emp(); e.setId(); ....
  Ses.save(e);
  Ses.getTransaction.commit();
}
```

Annotations:-

```
@Table(name = "Table Name");
@ column(name = "Column Name")
@ Transient  - Don't save Particular col as colname in table
@ Column(nullable = false);  -making col as not null.
@ Temporal(TemporalType.DATE)  .TIMESTAMP.

cg @ Id
@ SequenceGenerator(name = "", seqname = "", allocationSize = 1);
@ GeneratedValue(generator = "empid", strategy = GenerationType.SEQ)
Private int empid;
```

cg Password2

```
// DDL Stmt
Public void CreateTable()
{
   Sql = "Create Table tab(id integer, name varchar2)";
   jdbcTemplate.execute(sql);
}
```

execute only for DDL Stmts

NamedParameterJdbcTemplate:- JdbcTemplate won't support named Param's. It supports only "?" place holders.

```
Public void insertCircle(Circle c)
{
  Sql = "insert into circle(id,name) values(:id,:name)";
  SqlParameterSource namedparameters =
     new MapSqlparameterSource("id", c.getId()).
              addvalue("name", c.getName());
  njdbctemp.update(sql, namedparameters);
}
```

// Two ways to initialize JdbcTemplate object:

1. Creation of JdbcTemplate object in setter method of DS (above eg)
2. making new bean entry in xml for JdbcTemplate.

// JdbcTemplate - For "?" placeholders
// NamedparameterJdbcTemplate - For "named parameters"
// SimpleJdbcTemplate - For both.
↳ won't have much fns / features compare to others.

Spring JDBC Template

Spring & DataSource config :-

Spring.xml

```
<beans xmlns: .... >
   <context: annotation-config />
   <context: component-Scan base-Package = " " />
   <bean id="dataSource" name="org.apache.com. ">
      <property name="driveclassName" value=" " />
      <property url = " " value = " " />
      <property name = "initialyze" value = "2" />
      <property name = "maxActive" value = "5" />
   </bean>
</beans>
```

JdbcdaoImpl.java

```
@component
class JdbcdaoImpl
{ @ Autowired
  Private DataSource dataSource;
    // getters & setters
    // code for retrive data
}
```

main class

```
PSVM()
  ApplicationContext con = newclassPathXmlAC("SP.xml");
  JdbcdaoImpl dicon.getBean("jdbcdaoImpl", "JdbcdaoImpl.class")
  Circle C = di.getCircle();
    SOP(C.getName());
```

One class Two tables :-

```
@Entity
@Table(Name = "cust")
@SecondaryTable (name = "cust details")
class Customer
{
    private int custId;
    private String custName;
    @Column (Table = "cust details")
    private String address;
}
```

Two classes one table :-

```
@Entity
class School
{
    private int Sid;
    private String SName;
    @Embedded
    private Scdetails scd;
}
```

```
@Embeddable
class Scdetails
{
    private String Scaddr;
    private boolean isPublic;
    private int StdCount;
}
```

Compound key:-

```
@Entity
class Accounts
{ @Id
    CompoundKey ckey;
    private int accbal;
}
```

```
@Embeddable
class CompoundKey implements Serializable
{
    private int acid;
    private int userid;
}
```

Inheritance

```
@Entity
@Inheritance (Strategy = InheritanceType. SINGLE_TABLE)
class Project
{ @Id
  @Generated
  private int proj_id;
  projName;
}
```

```
@Entity
class Module extends Project
{
    moduleName;
}
```

```
@Entity
class Task extends module
{ TaskName; }
```

- SINGLE_TABLE = default, [Pid, PName, mName, Tname] (1)
- JOINED = [Pid, PName], [Pid, mName], [Pid, TName] (3)
- TABLE-PER-CLASS = [Pid, Pname], [Pid, Pname, mname], [Pid, Pname, mname, Tname] (3)

One To One :-

```
@Entity
class Person
{ @Id @Gener
  Personid;
  Personname;
  @OneToOne (cascade = CascadeType. ALL, fetch = FetchType.EAGER)
  @JoinColumn (name = "PDetail_FK")
  private PersonDetail Pdetails
}
class PersonDetail
{ @Id @GeneratedValue @Column (name = "detailId_Pk")
    private int personDetail id;
    Zipcode; Job;
}
```

CascadeType. ALL, REMOVE, REFRESH, MERGE, PERSIST

- EAGER (1-1)
- LAZY (1-*)

If u won't define cascade to insert two rec in two tables u have to write 2 save fnt in main like save(P); save(PersonDetail);

JDBCTemplate: It a class would be used to minimise the code. It concentrates on piece of code before exec of Sql Query and after execut piece of query.

Initialization of JdbcTemplate:-

```
class JdbcdaoImpl
{
    private DataSource dataSource;
    private JdbcTemplate jdbcTemplate;
    // getter & Setter
    @Autowired
    public void setDataSource (DataSource ds)
    { this.jdbcTemplate = new JdbcTemple (ds); }

    Public int getNumb of Circles ()
    {
        String sql = "select count(*) from circle";
        return jdbcTemplate.queryForInt (sql);
    }

    Public String getCircleName (~~create~~ int cid)
    { String Sql = "select circlename from cid = ?";
      return jdbcTem.queryForObject (sql, new Object[] {cid}, String.class)
        // 1. Sql
        2. Runtime Parameters
        3. Returntype which obj was return after execution of Sql.
    }
```

```
// Return model Object.
Public Circle getCircleforId (int cid)
{ Sql = "Seled * from Circle where cid = ?";
  return jdbcTemp.queryForObject
      (Sql, new Object[] {cid}, new CircleMapper ());
}
private Static final class CircleMapper implements
{                    RowMapper <Circle>
    Public Circle mapRow (ResultSet rs, int rowNum)
                                  throws SQLException
    {
        Circle c = new Circle ();
        c.setId (rs.getInt ("id");
        c.setName (rs.getString ("name");
        return c;
    }
}
// Return List
Public List <Circle> getallCircles ()
{     Sql = "Select * from circle";
      return JdbcTemp.queryForObj (Sql, new CircleMapper);
}
// write data into db
Public void insertCircle (Circle circle)
{ Sql = "insert into circle values (?, ?)";
  JdbcTemp.update (Sql, new Object[] { c.getId(), c.getName()
}
    => di.insertCircle (new Circle (2, "second cir");
```

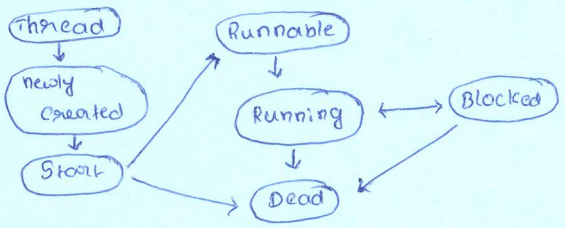
Thread

- process is an execution of a prog. But thread is a single execution seq with in the process.

- class conter extends Thread
```
{
  public void run() { ≡}
  psvm (S ar[])
  {
     Thread t1 = new counter();
      t1.start();
  }
}
```

- class conter extends base implements Runnable
```
{
  public void run() { ≡}
  psvm (S ar[])
  {
     Thread t1 = new Thread(new counter());
      t1.start();
  }
}
```

- Thread States: [Inmobile explanation]



File Handling

byte Stream:- data in form of bits & bytes eg text, audio video
Char Stream:- data in form of char eg only Text files.

byte	Char
FileInputStream	Filereader
FileOutputStream	Filewriter
DataInputStream	Buffered Reader
Buffered InputStream	Buffered writer.

① Reading from keyboard writing into file.
```
{
  DataInputStream dis = new DIS(System.in);
  FileOutputStream fout = new FOS("my.txt");
  char ch;
      while ( ch != (char) dis.read() != "@")
      {  fout.write(ch);
      }  fout.close();
}
```
→ BufferedOutputStream bos = new BOS(fout, 1025); (Size of Buff)

② Reading from File & write on console.
```
FileInputStream fis = new FIS("my.txt");
int ch;
while ((ch = fis.read()) != -1)
      SOP(char(ch));
      fis.close();
```
→ BufferedInputStream bis = new BIS(bin);

- Synchronized Threads:

Syn block:

```
Synchronized (obj)
{ ≡ }
```

Syn method:

```
Synchronized void method()
{ ≡ }
```

- Thread t = new Thread();
  = new Thread (obj);
  = new thread (obj, thredd name);
- t.join(); To wait till a thread dies.
- t. isAlive(); return true/false.
- Thread Scheduler is a prog in JVM. That loads threads into memory & executes them according to their priority.
- yield(): It changes from Running to Runnable.
  Sleep(): Running to wait/sleep state.
- A Thread can acquire the lock for an obj by using Synchronized keyword.
- Block level is more efficient because it doesn't lock the whole method.
- disadvantage of Syn is that it can cause deadlock when two threads are waiting on each other to do something. Also Syn code has the overhead of acquire lock, which can adversely the performance.
- Thread can comm each other using wait(), notify(), notifyall().

Char Stream

① create text file.

```
String Str = "This is an intl I am Stu";
File writer fw = new FW ("my.txt");
for( i=0; i< Str.length(); i++)
    fw.write (Str.charAt(i));
    fw.close();
```

→ BufferedWriter bw = new BW (fw, 1024);

② Reading from file write on console.

```
FileReader fr = new FR ("my.txt");
int ch;
    while ( (ch = fr.read()) != -1 )
        SOP ((char) ch);
    fr.close();
```

→ BufferedReader br = new BR(fr, 1025);

PL/SQL

DECLARATION
BEGIN
EXCEPTION
END;

open c1;
fetch c1 into num;
close c1;

✓ Cursor: Should be in declared block

- Cursor c1 is
  Select id from emp where empname = "a";
- Cursor c2 (subid in varchar2) is
  Select custnum from emp where subid = sub_id;
- Cursor c1 return emp%ROWTYPE is
  Select * from emp where empid = 1;

✓ Procedure: IN, OUT, INOUT

```
Create or Replace procedure updatecourse (nam IN varchar)
is
    cnumber number;
    Cursor c1 is
      Select curse_number from C_Tab where cname = nam;
BEGIN
    open c1;
    fetch c1 into cnumber;
    insert into Std_cource values (nam, cnumber);
    commit;
    close c1;
EXCEPTION WHEN OTHERS THEN
    raise_application_error (-20001, 'An err encoun' ||
                                     SQLERRM);
END;
```

JSP

✓ JSP contains HTML & JSP tags

✓ JSP is all features of servlet & inaddition we can use implicit objs, predefined tags, expression lang & custum tags. If JSP modified, we don't need to redeploy the proj.

✓ JSP Life cycle:

- Translation of JSP Page
- Compilation "
- Class loading (class file is loaded by C.loader)
- Instantiation (obj of generated servlet is created)
- Initialization (jspinit() invoked by wc)
- Request processing (JSP service)
- destroy.

✓ There is no need of dir structure if you don't have class files or tld files.

✓ JSP API contains two pkgs

javax.servlet.jsp — javax.servlet.jsp.tagext

- JSPPage (interface)
  HttpJSPPage

Scripting elements :-

- Scriptlet tag eg <% out.print ("Hi"); %>
- expression tag eg <%= java.util.calander.getInstance().getTime(); %>
- decloration tag eg <%! int data=50 %>

This tag is used to declare fields & methods The code written in declarationtag is placed outside the service(). So it doesn't get memory at each req.

- Scriptlet tag will support only for declaring vars. but declaration for both.

- JSP implicit objects (only 9)

  Out → JspWriter (Type)

  request, response → HttpServletRequest, Response.

  config → ServletConfig

  application → ServletContext

  Session → HttpSession

  PageContext → pageContext

  page → object

  exception → Throwable.

out <% out.print (java.util.getTime()); %>

inServlet PrintWriter out = response.getWriter();

eg Request :-

<% String name = request.getParameter ("uname");
out.print (name); %>

Response :-

<% response.SendRedirect ("http://google.com"); %>

config :-

<web.xml>
<init-Param>
<Param-name> dname </P-n>
<Param-value> mahesh </P-v>
</i-P> </w.x>

<% out.print (config.getInitParameter ("dname")); %>

Session :-

<% session.SetAttribute ("user", name); %> (1 JSP)

<% session.getAttribute ("user"); %> (2 JSP)

page context scopes :-

- Page
- request
- Session
- application.