1. WS has provided as the following component,
2. WSDL
3. Skeleton
4. Stub
5. SOAP
6. HTTP
7. UDDI🡺 Universal Description Discovery and Integration
8. WS is for applications interaction
9. A Simple flow Explanation🡺 see the notes🡺 provider has Skeleton and Consumer has Stub where Stub and Skeleton(IS A SERVLET CLASS) has Same Method Declaration as SEI but Logic is to covert Java req and Res to SOAP req and res
10. We have 4 implementers of WS namely,(again this 4 are specifications only we need implementation)
11. JAX-RPC
12. JAX-M🡺 is for async
13. JAX-WS
14. JAX-RS
15. Eclipse will support only 3 implementations it seems for remaining we need some plugins,
16. Axis1
17. Axis2
18. Apache CXF
19. AXIS-1 example🡺 is the implementer of JAX\_RPC
20. Service endpoint Url syntax🡺http://localhost:8080/webappName/skeletonUrl/unique Name of the Service
21. AXIS-1 uses server-config.wsdd(will be present in web-inf folder and this file also contains class ,method and parameters details of a service) file in order to map unique name to the Service

Syntax🡺

<ns:service name=”Unique name of Service”-------- >

<ns1:parameter name=”**allowedMethods**” value=”add sub multiply”/>

1. WSDL GENERATION TOOL is 🡺org.apache.axis.wsdl.java2wsdl 🡺 this is a class that is acting as a WSDL generator
2. java2wsdl WSDL generator needs following information to generate WSDL
3. Service Class
4. endpoint url
5. wsdl file name
6. targetnamespace
7. web.xml file will have skeleton information
8. Steps to create WS provider and Consumer in STS see the notes
9. Final url to be hit in the Browser🡺 <http://localhost:8080/webappName/skeletonUrl/unique?wsdl>
10. In WS we have 2 types of Clients(a client app can be Standalone or web app)
11. Proxy based Client🡺 see the notes
12. DII client🡺 Dynamic Invocation Interface🡺 here we will not generate stub , ws implementors provide few set of classes that acts as a stub to call ws methods
13. Ms to invoke Service class methods we need its reference type, for getting the ref type we will use Service class of Java and URL class of Java🡺URL will have a wsdl location and Service class will have some sort of reference to SEI.
14. Axis2 Example🡺
15. Here Skeleton used is 🡺org.apache.**axis2**.transport.http.AxisServlet
16. Stub and WSDL generation see the notes

SOAP

1. SOAP🡺 Simple Object access protocol
2. IS a Messaging protocol
3. IN WS errors/exceptions are called as Faults
4. Soap contains 3 parts
5. Soap-envelope
6. Soap-header
7. Soap-Body
8. If req is failed then Soap Message will contain 4th Section called🡺 Soap-fault
9. Syntax🡺

<?xml version=”1.0”?>

<soap: Envelope xmlns: soap=”http:------------“

Soap:encodingstyle=”http:-------“>

<soap:header>

------

</soap:header>

<soap:body>

-------

<soap:fault>

------

</soap:fault>

</soap:body>

</soap: Envelope>

1. Here Servlet class used is (I don’t know the Exactly which implementation is this)

<servlet-class>Com.sun.xml.ws.transport.http.servlet.WSServlet</servlet-class>

Note🡺 **for this servlet some Listener class is also used**

Rest Service

1. Rest🡺Representational state transfer
2. REST is an architecture
3. REST service don’t have such WSDL specification file thus URL or URI itself is a specification for rest services to access the web service
4. In case of rest client will directly use the uri to get response
5. SOAP is a Protocol and REST is a architecture
6. Soap permits XML data format only but REST allows different formats like HTML, XML, JSON and etc...
7. SOAP supports different types of protocols like http protocol, RMI protocols. But REST supports only Http and https
8. REST do not need client generation, wsdl generation and etc..
9. Jersey is the implementation of JAX-RS
10. Apache CXF also provides implementation for JAX-RS
11. HTTP Methods,
12. Get🡺 to get information from Server
13. Post🡺 to send or save information in the server
14. Put🡺 is use to update data of that URL
15. Delete🡺 to delete data of that URL
16. Metadata🡺 along with req and res we also have header information which is Meta data
17. HTTP Status Code,
18. 2xx🡺 success
19. 1xx🡺informational
20. 3xx🡺redirection
21. 4xx🡺Client error
22. 5xx🡺Server error
23. We can pass the input parameters to operation In 2 ways for a rest service
24. Request params
25. Path parameter
26. Syntax for Path parameter🡺

Baseurl/100🡺 where 100 is employee id this is sent as method input

1. Syntax for Request parameter or Query Parameter,

baseURL**?id**=1

Note🡺 Query parameter is not like a path parameter its additional parameter that gets added to http header

1. As discussed early along with req and res we have header information🡺 note we have req header and response header
2. List of annotations used
3. @Path🡺 both at class level and method level
4. @Produces🡺is used to, what we want to return from WS 🡺 ms is method level @
5. @Consumes🡺 is used to, what we want to send to WS🡺 ms is method level @
6. @PathParam
7. @Get
8. @Post
9. @Put
10. @Delete
11. JSON will store the value in the form of key and value pair
12. JSONLint is used to validate JSON.
13. @Produces(MediaType.APPLICATION\_JSON)
14. Example
15. web.xml🡺 root tag for web.xml is <web-app>

<servlet>

------

<servlet-class>org.glassfish.jersey.servlet.ServletContainer</ servlet-class>

<initiparam>

<param-name>jersey.config.server.provider.packages</param-name>

<param-value>com.capgemini.services</param-value>

</initiparam>

<load-on-startup>1</ load-on-startup>

</servlet>

<init-param>🡺 in our app we might have 10 packages among this 10 packages only one package will have all service classes thus we need to tell JAX\_RS from where or which package it should start searching for WS classes

1. Java file

@Path(“/Capgemini”)

Public class NotifyNondlineRestService

{

@GET

@Path(“{a}/{b}”)

Public int add (@PathParam(“a”) int a, @PathParam(“b”) int b)

{

Return a+b;

}

}

1. Output🡺

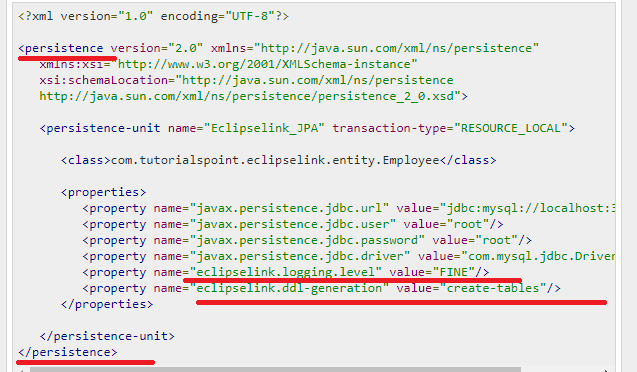
http:..loclahost:8080/applicationContext/url in web.xml file/url as in @path annotation**/10/20**

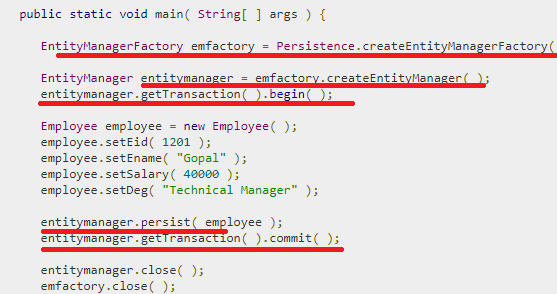
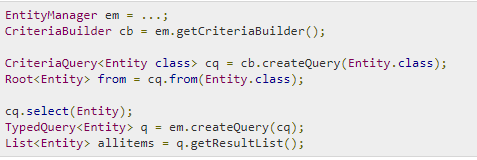
JPA

1. JPA is an api(classes and interface) to persistently store the vast amount of data into DB
2. JPA is by Oracle
3. While Introducing EJB 3.0 the persistence layer was separated and specified as JPA 1.0
4. JPA 2.1 in 2013
5. Has EMF, EM ,E, ETransaction, Query, Persistence
6. Persistence🡺 this class contains static method to obtain EMF
7. EM and ET has one to one relationship
8. ORM takes care of Converting data from Object type to relational type and Vice versa
9. Mapping.xml🡺



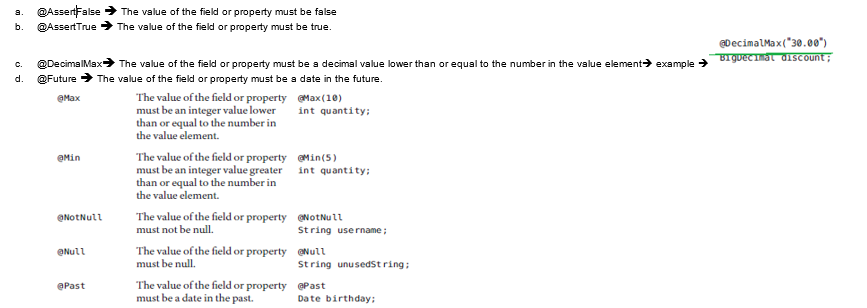
1. List of Annotations
2. @SequenceGenerator
3. @TableGenerator
4. @UniqueConstraint
5. Bean Conversion 🡺should gave getters for non-Boolean Property and must have isEmpty() for boolean properties
6. Persistence.xml



1. EntityManagerFactory emfactory = Persistence.createEntityManagerFactory( "Eclipselink\_JPA" );
2. CRUD methods 🡺 ms
3. Persist
4. Find
5. Remove
6. Update
7. 
8. CQ is Type-safe
9. 

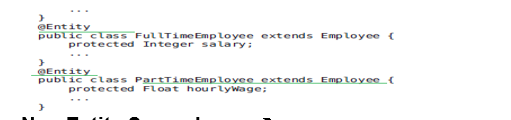
JPA Documentation

1. **Needed an Example on Arrays storage**
2. **Map Was not covered in Collections Section**
3. Bean Validation

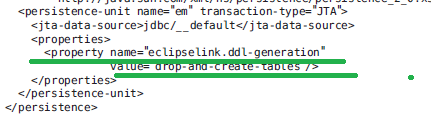


1. @Pattern and @Size🡺
2. **Example on primary key as Date field**
3. Caccade\_type.All/Detach/Merge/persist/refresh/remove
4. Abstract entities🡺 an abstract class annotated with @Entity. Abstract entities are like concrete entities but cannot be instantiated.
5. Mapped superclass🡺 Entities may inherit from super classes that contain persistent state and mapping information but are not entities. That is, the superclass is not decorated with the @Entity annotation and is not mapped as an entity by the Java Persistence provider.

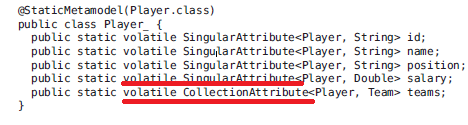
Example🡺 Parent 

Child🡺

1. @SecondaryTable🡺
2. MetaModel🡺 MS🡺CriteriaQuery uses metamodel to create query
3. **Metamodel classes model an entity’s attributes and are used by Criteria queries to navigate to an entity’s attributes**
4. Automatic table generation inJPA



1. An Equivalent Meta model class for every entity gets generated at ,**compile time🡺 in maven I think during maven install on JPA project**
2. Syntax🡺



1. The **EntityManager.createQuery and EntityManager.createNamedQuery** methods are used to query the data store by using Java Persistence query language queries
2. Syntax🡺 same as in hibernate, in case of createQuery(“Sql Query is sent as input)

And in case of createNamedQuery(Name of the Query is sent as input);

1. Same as in hibernate JPA aslo has Positional parameter prefixed with “ ?Position number” and Named parameter which is prefixed with “**:**parameter\_name”
2. Needed more information on
3. MEMBER OF
4. EXISTS
5. NOT EXISTS🡺 Scenarios
6. Set of Functions🡺 see later
7. **NEEDED MORE INFORMATION ON CASE EXAMPLE(AND ALSO IN PLSQL)**
8. **Constructor expressions allow you to return Java instances that store a query result element instead of an Object [].**

**Example🡺**

****

NOTE 🡺 ms 🡺**CriteriaQuery has methods like, select, from, where, having, groupBy and orderBy**

1. **Metamodel classes are typically generated by annotation processors either at development time or at runtime**
2. One of the programmatic way of getting metaModel class is🡺

object\_OF\_Root.getModel:

we can also get from em as shown below

**EntityType<Employee>Employee\_=em.getMetamodel().entity(Employee.class)**

1. For more information On CQ and CB methods First see the Notes
2. Ms🡺CriteriaBuilder methods is returning Predicate
3. **NEEDED MORE INFORMATION ON PREDICATE AND EXPRESSION IN JPA**
4. Methods of CriteriaBuilder,
   1. and🡺 return type is Predicate
   2. asc🡺Order
   3. des🡺Order
   4. avg🡺Expression
   5. sum🡺
   6. Count🡺Expression
   7. concat🡺 Expression
   8. between🡺Predicate
   9. equal🡺Predicate
   10. equals🡺Boolean object
   11. exists🡺P
   12. ge()🡺P
   13. greaterThan🡺P
   14. greaterThanOrEqualTo🡺p
   15. greatest🡺P
   16. gt()🡺P, lt
   17. in
   18. isEmpty()🡺P
   19. isnotEmpty
   20. isFalse🡺P
   21. isMember🡺P
   22. isNotMember
   23. like
   24. min
   25. max
   26. not
   27. notEqual
   28. notLike
   29. length
5. Methods of the Root<T> is🡺 ms actually root refers to “from” part of the SQL query
   1. get(field\_name)
   2. fetch(field\_name)
   3. in()
   4. isNull
   5. isNotNull
   6. join()
   7. joinCollection
   8. joinList
   9. joinMap
   10. joinSet

Enum

1. Example🡺

Public enum Week implements Serializable{

SUNDAY(“sun”, “First Day of the Week),

MONDAY(“mon”, “second Day of the Week),

TUESDAY(“tues”, “third Day of the Week),

WEDNESDAY(“wed”, “fourth Day of the Week),

THURSDAY(“thurs”, “fifth Day of the Week),

FRIDAY(“fri”, “sixth Day of the Week),

SATURDAY(“sat”, “seventh Day of the Week),

Private String code;

Private String description;

Private Week(String cd, String des){

this. code= cd;

this. description= des;

}

Public getcode(){

return code;

}

Public getdescription(){

return description;

}

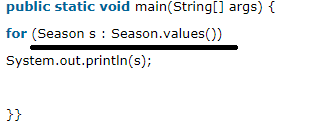
}

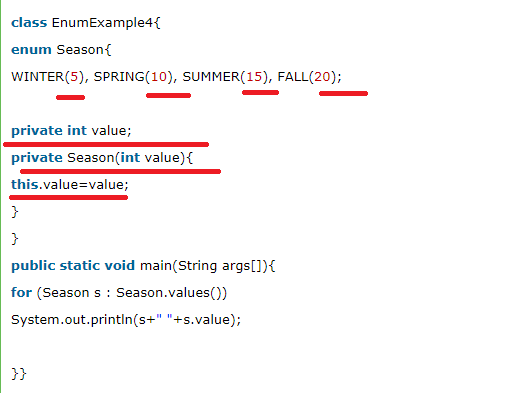
Out the class calling an enum syntax is🡺

String codeFromEnum=Week.SUNDAY.getCode();

String descriptionFromEnum=Week.SUNDAY. getdescription();

1. Enum is used to hold set of predefined values or constant values
2. Implicitly enum constants are static final
3. By default enum will have private constructor
4. Enum can be inside or outside class also.
5. enum improves type safety
6. enum can be easily used in switch
7. enum can have fields, constructors and methods🡺 example see above
8. **enum may implement many interfaces but cannot extend any class because it internally extends Enum class**
9. The java compiler internally adds the values() method when it creates an enum. The values() method returns an array containing all the values of the enum.

Example🡺

1. **we can initialize the specific value to the enum constants by defining fields and constructors🡺 as shown in above example**
2. 

Ms🡺See as shown in the above code🡺 will creating the enum only we are giving its value, and we are not passing any values from outside

1. **Can we create the instance of enum by new keyword? 🡺 No, because it contains private constructors only.**

|  |
| --- |
|  |

Java-Printout

1. **Object-oriented🡺 means we organize our software as a combination of different types of object that has both data and behavior**
2. **OOPS🡺 is a methodology that simplify software development and maintenance by providing some rules**
3. JDK🡺JRE+ debugger+ COMPILER
4. Jre🡺jvm + set of libraries
5. JVM🡺 ms where my app enters and gets executed
6. Data Types🡺 
7. Labelled for loop🡺 needed example on this
8. **Polymorphism🡺 one task is performed in different ways**
9. Encapsulation🡺 Binding or wrapping the code and data together is called encapsulation

JAVA BEAN is fully encapsulated

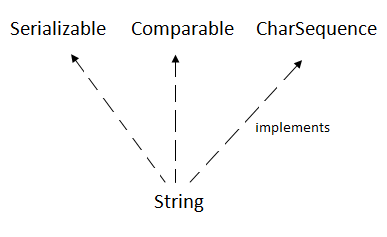
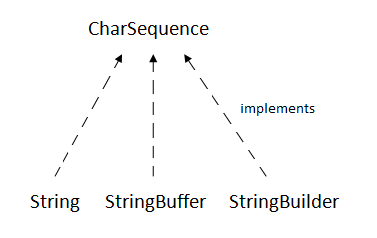
1. Needed example on newInstance() method for object creating in java
2. why the constructor has name Constructor🡺 it constructs the values i.e. provides data for the object
3. Aggregation purpose is reusability
4. **I had observed that Javac places IIB inside each constructor of Child but not in the parent constructor.**
5. Use instanceOf operator to avoid ClassCastException at run time(if both the classes does not have **“is-a”** relationship means we will get class cast exception at run time).
6. Static import🡺advantage🡺 helps in less coding
7. See Qspider notes for more information on System.out.println()
8. Primitive type into its equivalent Wrapper types is called boxing
9. Methods of object class
10. hashCode()
11. equals()
12. clone()🡺throws CloneNotSupportedException
13. toString
14. notify
15. notifyAll
16. wait🡺InterruptedException
17. finalize
18. datatype[] arrayname= new datatype[size]; 🡺or 🡺int[] arrayname ={1,2,3}
19. array has length property
20. In java, array is an object. For array object, an proxy class is created whose name can be obtained by getClass().getName() method on the object.
21. We can copy an array to another by the arraycopy method of System class. 🡺

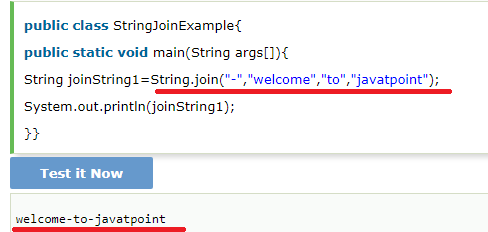
Example

**public** **static** **void** arraycopy(

Object src, **int** srcPosfromwheretocopy,Object dest, **int** destPoswheretoPaste, **int** length

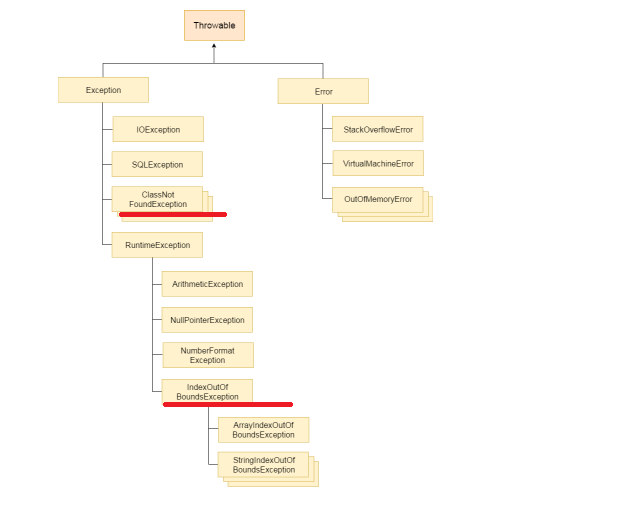
)

1. Wrapper🡺
2. valueOf()🡺 to convert primitive to object type and its static method
3. xxxValue🡺 to convert object type to primitive and its instance method🡺 example intValue()
4.  
5. Methods of String class
6. [char charAt(int index)](https://www.javatpoint.com/java-string-charat)
7. [int length()](https://www.javatpoint.com/java-string-length)
8. [static String format(String format, Object... args)](https://www.javatpoint.com/java-string-format)
9. [String substring(int beginIndex)](https://www.javatpoint.com/java-string-substring), [String substring(int beginIndex, int endIndex)](https://www.javatpoint.com/java-string-substring)
10. [boolean contains(CharSequence s)](https://www.javatpoint.com/java-string-contains)
11. [static String join(CharSequence delimiter, CharSequence... elements](https://www.javatpoint.com/java-string-join))



1. [boolean equals(Object another)](https://www.javatpoint.com/java-string-equals)
2. [static String equalsIgnoreCase(String another)](https://www.javatpoint.com/java-string-equalsignorecase)
3. [boolean isEmpty()](https://www.javatpoint.com/java-string-isempty)
4. [String concat(String str)](https://www.javatpoint.com/java-string-concat)
5. [String replace(char old, char new)](https://www.javatpoint.com/java-string-replace), [String replace(CharSequence old, CharSequence new)](https://www.javatpoint.com/java-string-replace)
6. [String[] split(String regex)](https://www.javatpoint.com/java-string-split)
7. [int indexOf(int ch)](https://www.javatpoint.com/java-string-indexof), [int indexOf(int ch, int fromIndex)](https://www.javatpoint.com/java-string-indexof), [int indexOf(String substring)](https://www.javatpoint.com/java-string-indexof), [int indexOf(String substring, int fromIndex)](https://www.javatpoint.com/java-string-indexof)
8. [String toLowerCase()](https://www.javatpoint.com/java-string-tolowercase),[String toUpperCase()](https://www.javatpoint.com/java-string-touppercase)
9. [String trim()](https://www.javatpoint.com/java-string-trim)
10. [static String valueOf(int value](https://www.javatpoint.com/java-string-valueof))
11. Needed few information on StringTokenizer
12. Methods of StringBuffer
13. append(String s)🡺Synchronized
14. insert(int offset, String s)🡺 Synchronized
15. replace(int startIndex, int endIndex, String str)🡺 Synchronized
16. delete(int startIndex, int endIndex)🡺 Synchronized
17. reverse()🡺Synchronized
18. and few String methods🡺

* capacity(),
* ensureCapacity(int minCapacity)
* charAt(int index)
* length()
* substring

1. 

Needed more information on ClassNotFoundException and ClassDefException

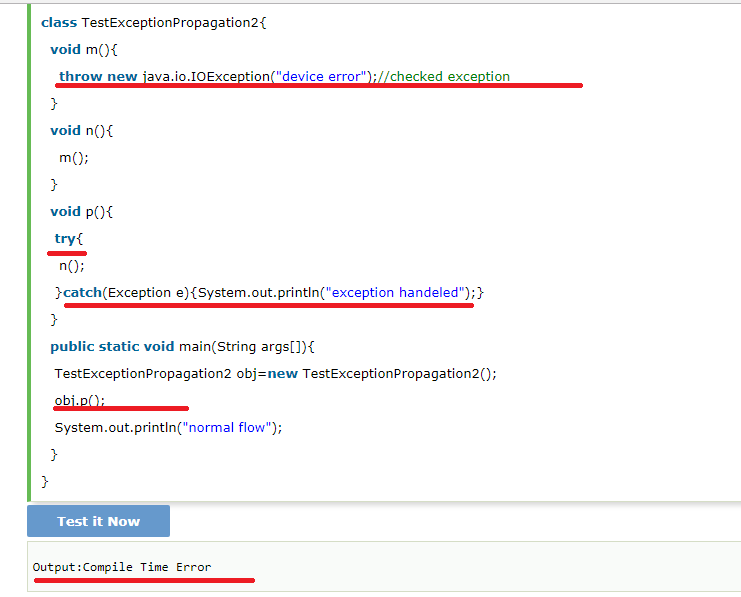
1. CheckedException, unCheckedException and errors
2. Checked Exception🡺 should be handled using try-catch block or throws keyword
3. **Suppose if method does not throw Checked Exception then I cannot simply Throw it either by using Throws or throw🡺 ms 🡺 check this again**
4. I cannot throw Checked exception using throw keyword🡺ms
5. No statements between try-catch-finally🡺 it will become unreachable
6. Throw keyword is mainly used to throw custom Exceptions
7. The Java throw keyword is used to explicitly throw an exception.

# **Java Exception propagation**

#### Rule: By default Unchecked Exceptions are forwarded in calling chain (propagated).

#### **Rule: By default, Checked Exceptions are not forwarded in calling chain (propagated).**

Example for above rule is,



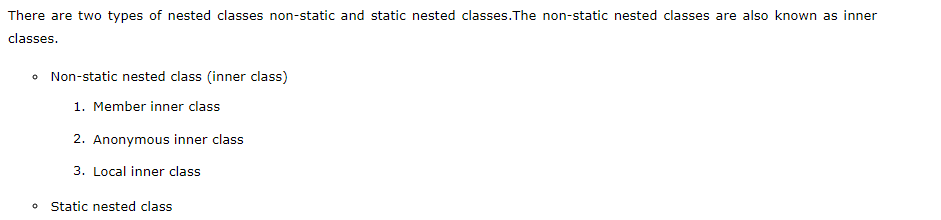
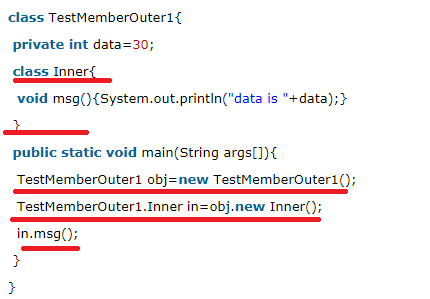
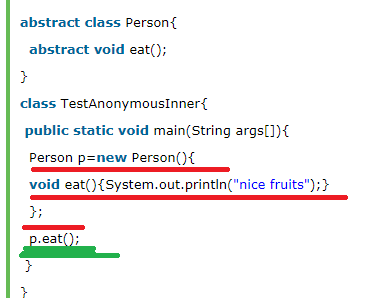
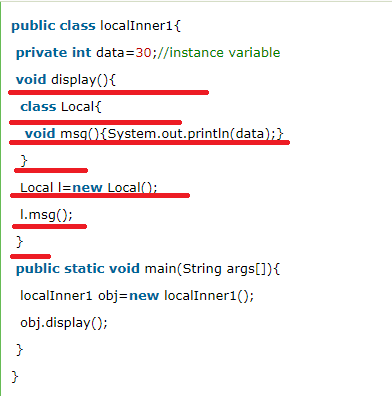
### Rule: If you are calling a method that declares an exception, you must either caught or declare the exception.

1. Lets see the above rule🡺

There are two cases:

1. **Case1:**You caught the exception i.e. handle the exception using try/catch.
2. **Case2:**You declare the exception i.e. specifying throws with the method.

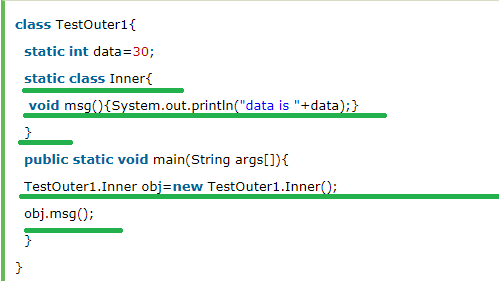
# **Java Inner Classes**

1. 
2. A non-static class that is created inside a class but outside a method is called member inner class. 🡺
3. A class that have no name is known as anonymous inner class in java. It should be used if you have to override method of class or interface. Java Anonymous inner class can be created by two ways: 🡺Interface, abstract Class
4. 
5. A class i.e. created inside a method is called local inner class in java. If you want to invoke the methods of local inner class, you must instantiate this class inside the method.
6. 
7. Rules for LIC🡺

#### Rule: Local variable can't be private, public or protected.

#### 1) Local inner class cannot be invoked from outside the method.

#### 2) Local inner class cannot access non-final local variable till JDK 1.7. Since JDK 1.8, it is possible to access the non-final local variable in local inner class.

1. Static Nested classes
   1. A static class i.e. created inside a class is called static nested class in java. It cannot access non-static data members and methods. It can be accessed by outer class name.
   2. 

Explanation🡺

* see static inner class has non static method
* Syntax to call static inner class static method🡺  TestOuter2.Inner.msg();

Java\_PrintOut1

1. Thread is a light weight **sub process**, a smallest unit of processing
2. On calling start() Thread moves from new to runnable state
3. Methods of thread class
   * 1. **run()**
     2. **start**
     3. **sleep(long miliseconds)🡺static**
     4. **join(),join(long miliseconds)🡺** The join() method waits for a thread to die. In other words, it causes the currently running threads to stop executing until the thread it joins with completes its task.

****

**Incase of join(milliseconds)🡺for the specified milliseconds thread will not release the resource🡺 example🡺** when t1 is completes its task for 1500 miliseconds(3 times) then t2 and t3 starts executing.

* + 1. **yield()**
    2. **getPriority(),setPriority()**
    3. **getName(),setName(String name)**
    4. **Thread currentThread()🡺static**
    5. **getId()**
    6. **public Thread.State getState() 🡺 returns the state of the thread**
    7. **isAlive():**
    8. **isDaemon,** **setDaemon**
    9. **interrupt, isInterrupted,** **interrupted**
    10. **suspend, resume** ,**stop🡺 all are deprecated**

1. **Thread scheduler** in java is the part of the JVM that decides which thread should run.
2. There is no guarantee that which runnable thread will be chosen to run by the thread scheduler.
3. No. After starting a thread, it can never be started again. If you does so, an IllegalThreadStateException is thrown
4. If you sleep a thread for the specified time, the thread scheduler picks up another thread and so on
5. By default each thread has a name i.e. thread-o, thread-1 and so on
6. Make a user thread daemon thread before its started otherwise we will get IllegalThreadStateException
7. Thread Group, Thread Pool, Java Shutdown hook
8. GC is a process of reclaiming runtime unused memory or is a process of destroying the unused obj

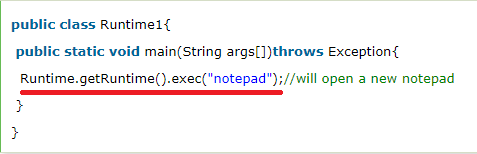
#### Note: The Garbage collector of JVM collects only those objects that are created by new keyword. So if you have created any object without new, you can use finalize method to perform cleanup processing (destroying remaining objects).

1. The gc() is found in System and Runtime classes.

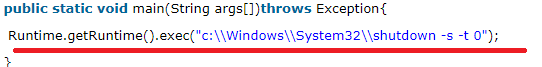
#### Note: Garbage collection is performed by a daemon thread called Garbage Collector(GC). This thread calls the finalize() method before object is garbage collected.

#### Note: Neither finalization nor garbage collection is guaranteed.

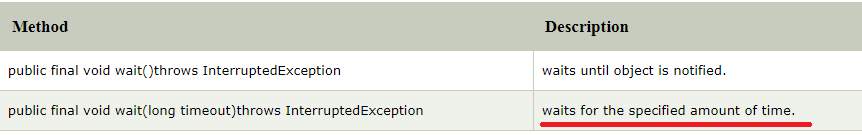
1. **Java Runtime** class is used to interact with java runtime environment
2. **Methods of Runtime,**
   1. static Runtime getRuntime()
   2. exit(int status)
   3. availableProcessors()
   4. freeMemory()
   5. totalMemory()
   6. public Process exec(String command)throws IOException 🡺 executes given command in a separate process.



## **How to shutdown system in Java using exec()**



## **restart 🡺** Runtime.getRuntime().exec("shutdown -r -t 0");

1. 
2. If any thread is in sleeping or waiting state (i.e. sleep() or wait() is invoked), calling the interrupt() method on the thread, breaks out the sleeping or waiting state throwing InterruptedException.

JDBC

1. 4 types of driver🡺

* JDBC-ODBC bridge driver
* Native-API driver (partially java driver)
* Network Protocol driver (fully java driver)
* Thin driver (fully java driver)

1. 5 Steps in connecting java app to db
   1. Register the driver
   2. Create connection
   3. Create statement
   4. Execute query
   5. Close connection🡺by closing the connection obj Statement and ResultSet will be closed automatically

Java\_PrintOut2

1. The forName() method of Class class is used to register the driver class🡺 Class.forName("oracle.jdbc.driver.OracleDriver");  🡺
2. Methods of DriverManager class

* registerDriver(Driver driver):
* deregisterDriver(Driver driver):
* getConnection(String url):🡺  static
* static Connection getConnection(String url,String userName,String password):

1. Connection interface Methods,
   * 1. **createStatement**
     2. **setAutoCommit()**
     3. **commit**
     4. **rollback**

By default, connection commits the changes after executing queries

1. Statement methods,
2. **ResultSet executeQuery(String sql)**
3. **int executeUpdate(String sql)**
4. **boolean execute(String sql):**
5. **int[] executeBatch()**
6. Initially ResultSet maintains cursor to first row, methods of ResultSet

#### By default, ResultSet object can be moved forward only and it is not updatable.

1. **boolean next():**
2. **boolean previous():**
3. **boolean first()**
4. **boolean last()**
5. **int getInt(int columnIndex)**
6. **int getInt(String columnName):**
7. **String getString(int columnIndex):**
8. **public String getString(String columnName):**
9. Methods of PreparedStatement
10. setInt(int paramIndex, int value)
11. setString(int paramIndex, String value)
12. setFloat(int paramIndex, float value)
13. setDouble(int paramIndex, double value)
14. int executeUpdate()
15. ResultSet executeQuery()
16. The metadata means data about data i
17. ResultSetMetaData methods,
18. int getColumnCount()throws SQLException
19. String getColumnName(int index)throws SQLException
20. String getColumnTypeName(int index)throws SQLException
21. public String getTableName(int index)throws SQLException
22. DatabaseMetaData interface methods,
23. **String getDriverName()throws SQLException**
24. **String getDriverVersion()throws SQLException:**
25. **String getUserName()throws SQLException:**
26. **String getDatabaseProductName()throws SQLException:**
27. **String getDatabaseProductVersion()throws SQLException:**
28. **public ResultSet getTables(String catalog, String schemaPattern, String tableNamePattern, String[] types)throws SQLException**

Java\_PrintOur3

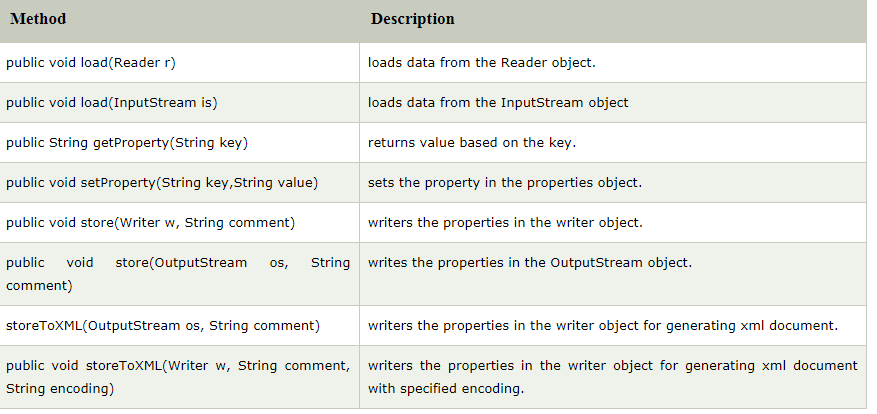
Collection

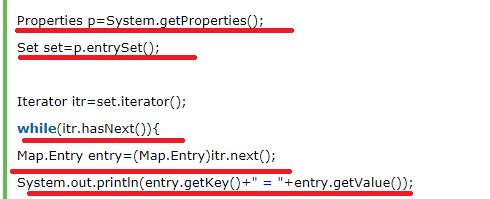
1. hierarchy of collection framework
2. Methods of Collection interface
3. boolean add(Object element)
4. boolean addAll(Collection c)
5. boolean remove(Object element)
6. boolean removeAll(Collection c)
7. boolean retainAll(Collection c)
8. boolean contains(Object element)
9. boolean containsAll(Collection c)
10. int size()
11. clear()
12. Iterator iterator()
13. Object[] toArray()🡺 Needed to do hands on this
14. boolean isEmpty()
15. boolean equals(Object element)
16. int hashCode()
17. Methods of Iterator interface,
18. boolean hasNext()
19. Object next()🡺 in case of ResultSet it’s of boolean type
20. void remove()
21. Methods of Iterator interface
22. Boolean hasPreviuos
23. Object previous
24. above methods
25. Methods of ArrayList
26. **void** add(int index, Object element)🡺 in case of Collection return type was boolean(no no see below method)
27. boolean add(Object o)
28. int indexOf(Object o)
29. int lastIndexOf(Object o)
30. boolean addAll(Collection c)
31. boolean addAll(int index, Collection c)
32. clear()
33. Object[] toArray()
34. **void trimToSize()🡺It is used to trim the capacity of this ArrayList instance to be the list's current size.**
35. Methods of List Interface
36. Add(posi, element)
37. addAll(posi, Collection obj)
38. get(position)
39. set(position, element)
40. remove(position)
41. ListIterator listIterator()
42. Linked list methods,
43. addFirst(Object o)
44. addLast(Object o)
45. Object getFirst()
46. Object getLast()
47. And all above methods of ArrayList +linked List(ms)
48. Methods of HashSet
49. boolean contains(Object o)🡺 It is used to return true if this set contains the specified element.==> Set has this method instead of Get(posi) as Set does not have key or index to get the element based on key or index.
50. Methods of TreeSet,
51. Object first()🡺It is used to return the first (lowest) element currently in this sorted set.
52. Object last() 🡺 It is used to return the last (highest) element currently in this sorted set.
53. Methods of Queue,
54. boolean add(object)🡺 It is used to insert the specified element into this queue and return true upon success.
55. boolean offer(object)🡺 It is used to insert the specified element into this queue.
56. Object remove()🡺It is used to retrieves and removes the head of this queue 🡺 I think this throws some exception
57. Object poll()🡺It is used to retrieves and removes the head of this queue, or returns null if this queue is empty.
58. Object element()🡺It is used to retrieves, but does not remove, the head of this queue.
59. Object peek()🡺 It is used to retrieves, but does not remove, the head of this queue, or returns null if this queue is empty.

NOTE🡺 CLEAR DIFFERENCE BETWEEN THIS METHODS IS GIVEN SEE NOTES OR STACK OVERFLOW

1. [Java Deque & ArrayDeque](https://www.javatpoint.com/java-deque-arraydeque)🡺 LATER
2. Methods of Map
3. Object put(Object key, Object value)
4. void putAll(Map map)
5. Object remove(Object key)
6. Object get(Object key)
7. boolean containsKey(Object key)
8. Set keySet()
9. Set entrySet()
10. Entry is the sub interface of Map. 🡺 methods are 🡺 Object getKey() and Object getValue()
11. HashMap🡺 it may have one null key and multiple null value. It contains only unique elements. it maintains no order.
12. Methods of HashMap🡺
13. boolean containsValue(Object value)
14. boolean isEmpty()
15. Collection values()🡺It is used to return a collection view of the values contained in this map.
16. LinkedHashMap🡺 It may have one null key and multiple null values. It contains only unique elements. **It is same as HashMap instead maintains insertion order.**
17. TreeMap cannot have null key but can have multiple null values.
18. Methods of TreeMap,
19. Object firstKey() 🡺 It is used to return the first (lowest) key currently in this sorted map.
20. Object lastKey()🡺It is used to return the last (highest) key currently in this sorted map.
21. Hashtable🡺 It may have not have any null key or value.
22. Methods of HashTable🡺
23. void rehash()It is used to increase the size of the hash table and rehashes all of its keys.

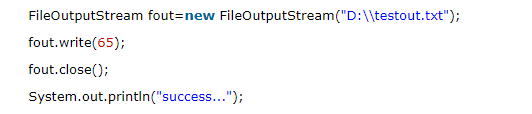
# Collections class🡺 has methods t Synchronize List, Set and Map and has the method to reverse(), and also has method to sort() a list

1. **Sorting a collection🡺 Comparable and Comparator interface🡺 needed to do hands on**
2. The java.util.Properties class is the subclass of Hashtable.
3. The Properties class provides methods to get data from properties file and store data into properties file.
4. Methods of Properties class
   1. 
5. By System.getProperties() method we can get all the properties of system

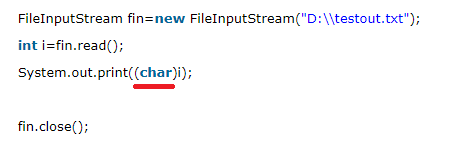


Java\_PrintOut4

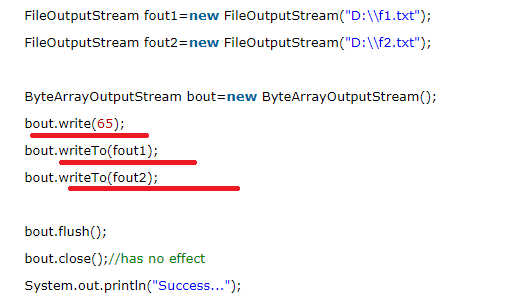
1. Java has Streams for Byte-orientation and Reader and Writer for Character-Orientation
2. OutputStream(is an abstract class and it’s the super class for all output stream class)🡺 for write and InputStream(is an abstract class and it’s the super class for all input stream class)🡺 do to read operation
3. OutputStream methods,
4. public void write(**int**)throws IOException
5. public void write(byte[])throws IOException
6. public void flush()throws IOException
7. public void close()throws IOException
8. Java output stream hierarchy
9. Java input stream hierarchy
10. FileOutputStream example🡺



1. FileIntputStream example🡺

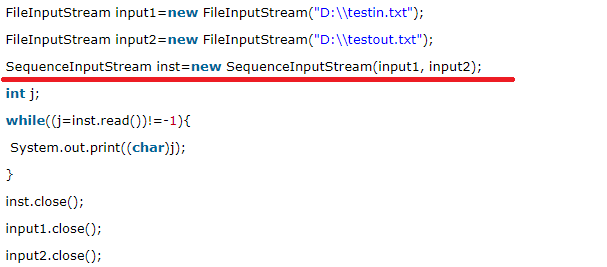


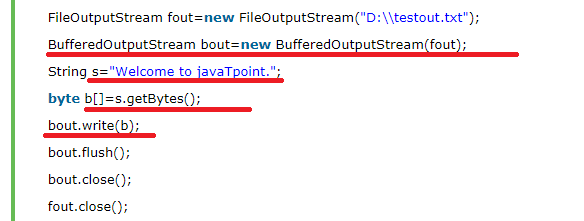
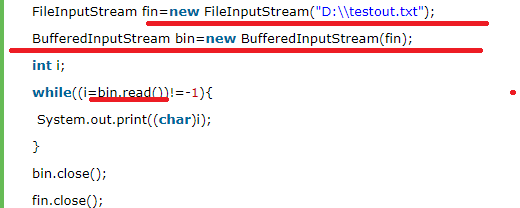
1. Java ByteArrayOutputStream class is used to **write common data** into multiple files, example,

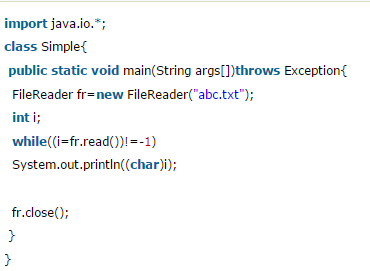


Ms🡺See here first content is written into BAOS by using write() and then writeTo() method is called to write into the files

1. Java SequenceInputStream class is used to read data from multiple streams. It reads data sequentially (one by one). Example🡺



1. **Needed more information on Enumeration**
2. BufferedOutputSTrem🡺  it makes the performance fast.
3. BufferedInputStream🡺
4. Reader and Writer🡺 needed hierarchy structure for this
5. Methods of Writer
6. append(char c), append(CharSequence csq), append(CharSequence csq, int start, int end)
7. write(char[] cbuf), write(int c), write(String str)
8. flush()
9. Reader methods🡺
10. read()
11. close()
12. reset()🡺It resets the stream.
13. skip(long n)🡺 It skips characters.
14. CharArrayWriter class can be used yto write data to multiple files
15. 



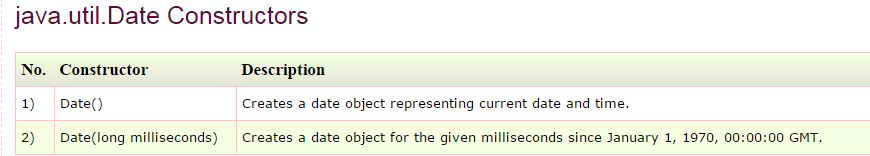
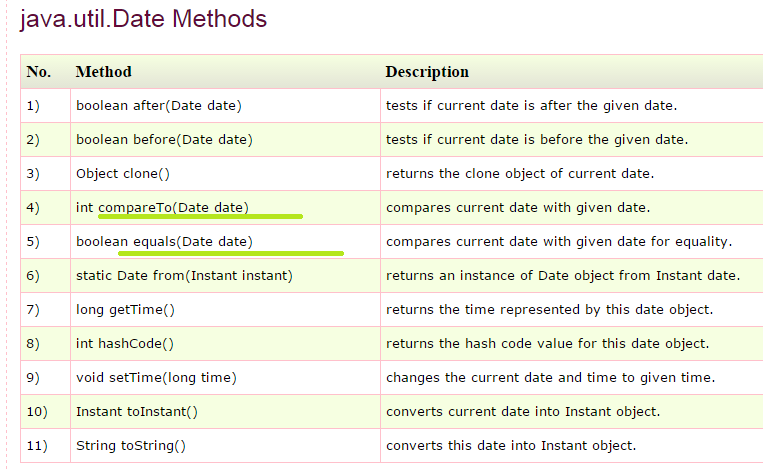
1. Reader example later

NOTE🡺 APART FROM THIS I HAVE OTHER STREAM AND READER CLASSES NEEDED TO COVER THOSE ALSO

Java PrrintOut5

1. Is the difference between Comparable and Comparator interface this we will see 🡺While doing hands on

Java PrintOut 6

1. 
2. 
3. System.currentTimeMillis();
4. java.sql.Date methods🡺
5. java.util.Calendar Java Calendar class is an abstract (Java Calendar class is an abstract) methods,
6. abstract void add(int field, int amount)🡺 It is used to add or subtract the specified amount of time to the given calendar field, based on the calendar's rules.
7. **int get(int field)**
8. static Calendar getInstance()
9. **void set(int field, int value)**
10. void setTime(Date date)🡺 It is used to set this Calendar's time with the given Date.
11. Date getTime()🡺It is used to return a Date object representing this Calendar's time value.
12. Calendar class has lot of Properties/constant fields
13. **NEEDED TO KNOW HOW TO COMPARE DATE ARRAY AS ALREADY DATE CLASS IMPLEMENTS COMPARATOR**
14. **Collect few information on Constructor Chaining and Serialization🡺Qspider**

Core Java Interview Questions(Tpoint )

1. JIT compiles parts of the byte code that have similar functionality at the same time, and hence reduces the amount of time needed for compilation. Here the term “compiler” refers to a translator from the instruction set of a Java virtual machine (JVM) to the instruction set of a specific CPU.
2. delete,next,main,exit or null IS NOT A keyword in java
3. What is the purpose of default constructor 🡺The default constructor provides the default values to the objects. The java compiler creates a default constructor only if there is no constructor in the class
4. What if the static modifier is removed from the signature of the main method🡺 Program compiles. But at runtime throws an error "NoSuchMethodError".
5. **Yes, all functions in Java are virtual by default.**
6. Can you declare the main method as final🡺yes

### What is the base class for Error and Exception? 🡺Throwable.

Can a class have an interface?==> **Yes, it is known as nested interface.**

Can an Interface have a class? 🡺 **Yes, they are static implicitely.**

1. What kind of thread is the Garbage collector thread? 🡺 DAEMON thread

How will you invoke any external process in Java? **🡺** By Runtime.getRuntime().exec(?) method.

1. Serialization is a process of writing the state of an object into a byte stream.It is mainly used to travel object's state on the network.
2. **Deserialization is the process of reconstructing the object from the serialized state.It is the reverse operation of serialization.**
3. If you define any data member as transient,it will not be serialized
4. Reflection is the process of examining or modifying the runtime behaviour of a class at runtime.
5. **The purpose of the System class is to provide access to system resources.**
6. **Singleton Class Program🡺**
7. **The shutdown hook is basically a thread i.e. invoked implicitely before JVM shuts down. So we can use it perform clean up resource**
8. ADVANTAGE OF PROPERTY FILE 🡺If you change the value in properties file, you don't need to recompile the java class. So, it makes the application easy to manage.
9. **Two different keys with the same hash value is known as hash-collision. Two different entries will be kept in a single hash bucket to avoid the collision.**