CMR INSTITUTE OF TECHNOLOGY

USN | | | | | | | | | | |

Internal Assessment Test 1– Jan. 2024

Sub:	Advanced Java & J2EE							Sub Code :	22MCA341
Date:	18/1/2024	Duration:	90 min's	Max Marks:	50	Sem:	III	Branch:	MCA

Note : Answer FIVE FULL Questions, choosing ONE full question from each Module

		MARKS	OBE	
			CO	RBT
	PART I			
1	What is an annotation? Explain how do you create a custom annotation with an example?	10	CO1	L2
	OR			
2.	With an example program describe the steps involved in connecting a java program to a database.	10	CO3	L2
	PART II			
3	Write a program to create a enum with days of week and accept a number from the user to access the constants from enum	10	CO1	L4
	OR			
4.	Develop an application to store employee data. Perform operations like update, search, delete and display the employee detail using JDBC	10	CO3	L4

	PART III			
	Discuss built-in annotations with example program	**10**	CO1	L2
	OR			
6	Explain in detail updatable and scrollable result set with an example program.	**10**	CO3	L2
	PART IV			
7	What is Enum? Explain the use of Values() and valueOf() methods with suitable examples?	**10**	CO1	L2
	OR			
8	What are Prepared Statement object and Callable Statement? How do we usethem Explain with an example	**10**	CO3	L2
	PART V			
9	Write a java program to insert 10 records in a table using batch updates.	**10**	CO3	L4
	OR			
10	What is boxing and unboxing? Explain how autoboxing and unboxing occurs in Boolean and Character values?	**10**	CO2	L2

1. What is an annotation? Explain how do you create a custom annotation with an example?

Java Annotations allow us to add metadata information into our source code,
Annotations were added to the java from JDK 5.
Annotations, does not change the actions of a program.
Thus, an annotation leaves the semantics of a program unchanged.
However, this information can be used by various tools during both development and deployment.
 Annotations start with '@'.
 Annotations do not change action of a compiled program.
 Annotations help to associate metadata (information) to the program elements i.e. instance variables, constructors, methods, classes, etc.
 Annotations are not pure comments as they can change the way a program is treated by compiler.

Java Custom annotations or Java User-defined annotations are easy to create and use. The @interface element is used to declare an annotation. For example:

```
@interface MyAnnotation{ }
```

Here, MyAnnotation is the custom annotation name.

Example

Declaring custom Annotation

```
@interface MyAnnotation{
int value1( ) default 1;
String value2( ) default;
String value3( ) default;
}
```

How to apply custom Annotation

```
@MyAnnotation(value1=10,value2=Umesh,value3=SJBIT)
```

2. With an example program describe the steps involved in connecting a java program to a database.

Seven Basic Steps in Using JDBC

1. Load the Driver
2. Define the Connection UR
3. Establish the Connection
4. Create a Statement Object

5. Execute a query
6. Process the results
7. Close the Connection

1. Load the JDBC driver

When a driver class is first loaded, it registers itself with the driver Manager
Therefore, to register a driver, just load it!
Example:

String driver = “sun.jdbc.odbc.JdbcOdbcDriver”;
Class.forName(driver); Or
Class.forName(sun.jdbc.odbc.JdbcOdbcDriver);

2. Define the Connection URL

jdbc : subprotocol : source

each subprotocol has its own syntax for the source

jdbc : odbc : DataSource
Ex: jdbc : odbc : Employee
jdbc:msql://host[:port]/database
Ex: jdbc:msql://foo.nowhere.com:4333/accounting

3. Establish the Connection

rns a Connection object

to the database from the DBMS.

nection() if access is granted;
else getConnection() throws a SQLException.

the database.

String url = jdbc : odbc : Employee;
Connection c = DriverManager.getConnection(url,userID,password);

access to the database.

Properties or Sometimes DBMS grants access to a database to anyone without using username or password.

Ex: Connection c = DriverManager.getConnection(url) ;

4. Create a Statement Object

A Statement object is used for executing a static SQL statement and obtaining the results produced by it.

Statement stmt = con.createStatement();

This statement creates a Statement object, stmt that can pass SQL statements to the DBMS using connection, con.

5. Execute a query

Execute a SQL query such as SELECT, INSERT, DELETE, UPDATE
Example String SelectStudent= "select * from STUDENT";

6. Process the results

table rows are retrieved in sequence.

7. Close the Connection
connection.close();

stpone this step if additional database operations are expected

```
package j2ee.p9;
import java.sql.*;
import java.io.*;
public class Studentdata {
        public static void main(String[] args) {
                Connection con;
                PreparedStatement pstmt;
                Statement stmt;
                ResultSet rs;
                String uname, pword;
                Integer marks,count;
                try
                {
                    Class.forName("com.mysql.jdbc.Driver"); // type1 driver
                      try{

        con=DriverManager.getConnection("jdbc:mysql://127.0.0.1/mca","root","system")
; //  type1 access connection
                            BufferedReader br=new BufferedReader(new
InputStreamReader(System.in));
                           do
                           {


                           System.out.println("\n1. Insert.\n2. Select.\n3. Update.\n4.
Delete.\n5. Exit.\nEnter your choice:");
                           int choice=Integer.parseInt(br.readLine());
                               switch(choice)
                               {
                                   case 1: System.out.print("Enter UserName :");
                                           uname=br.readLine();
                                           System.out.print("Enter Password :");
                                           pword=br.readLine();
                                           pstmt=con.prepareStatement("insert into student
values(?,?)");
                                           pstmt.setString(1,uname);
                                           pstmt.setString(2,pword);
                                           pstmt.execute();
                                           System.out.println("\nRecord Inserted
successfully.");
                                    break;
                                    case 2:
                                        stmt=con.createStatement();
                                        rs=stmt.executeQuery("select *from student");
```

```
                                        if(rs.next())
                                          {
                                        System.out.println("User Name\tPassword\n-----
--------------------------");
                                        do
                                          {
                                                uname=rs.getString(1);
                                                pword=rs.getString(2);

                                                System.out.println(uname+"\t"+pword);
                                        }while(rs.next());
                                        }
                                         else
                                                System.out.println("Record(s) are not
available in database.");
                                        break;
                                        case 3:
                                                System.out.println("Enter User Name to
update :");
                                                uname=br.readLine();
                                                System.out.println("Enter new password
:");
                                                pword=br.readLine();
                                                stmt=con.createStatement();
                                                count=stmt.executeUpdate("update
student set password='"+pword+"'where username='"+uname+"'");
                                                System.out.println("\n"+count+" Record
Updated.");
                                        break;
                                        case 4: System.out.println("Enter User Name to
delete record:");
                                                uname=br.readLine();
                                                stmt=con.createStatement();
                                                count=stmt.executeUpdate("delete from
student where username='"+uname+"'");


                                                if(count!=0)
                                                        System.out.println("\nRecord
"+uname+" has deleted.");
                                                else
                                                        System.out.println("\nInvalid
USN, Try again.");
                                        break;

                                        case 5: con.close(); System.exit(0);
                                        default: System.out.println("Invalid choice, Try
again.");
                                }//close of switch
```

```
                    }while(true);
                }//close of nested try
                catch(SQLException e2)
                    {
                        System.out.println(e2);
                    }
                catch(IOException e3)
                    {
                        System.out.println(e3);
                    }
            }//close of outer try
            catch(ClassNotFoundException e1)
            {
                System.out.println(e1);
            }
        }
    }
```

3. **Write a program to create a enum with days of week and accept a number fromthe user to access the constants from enum.**

```
enum Dayofweek
{
SUNDAY(1),MONDAY(2),TUESDAY(3),WEDNESDAY(4),THURSDAY(5),FRIDAY(6),SATURDAY(7);
int dno;
Dayofweek(int no)
{
dno=no;
}
boolean isweekday()
{
if(dno>1 && dno<7)
return true;
else
return false;
}
}
public class enumdemo {
public static void main(String arg[])
{
System.out.println(Dayofweek.FRIDAY.isweekday());
}
}
```

4. **Develop an application to store employee data. Perform operations like update,search, delete and display the employee detail using JDBC.**

```
package j2ee.p9;
import java.sql.*;
import java.io.*;
public class Employeedata {
        public static void main(String[] args) {
                Connection con;
                PreparedStatement pstmt;
                Statement stmt;
                ResultSet rs;
                String uname, pword;
                Integer marks,count;
                try
                {
                    Class.forName("com.mysql.jdbc.Driver"); // type1 driver
                      try{

        con=DriverManager.getConnection("jdbc:mysql://127.0.0.1/mca","root","system")
; //  type1 access connection
                            BufferedReader br=new BufferedReader(new
InputStreamReader(System.in));
                           do
                           {


                            System.out.println("\n1. Insert.\n2. Select.\n3. Update.\n4.
Delete.\n5. Exit.\nEnter your choice:");
                            int choice=Integer.parseInt(br.readLine());
                               switch(choice)
                               {
                              case 1: System.out.print("Enter Employee Name :");
                                        uname=br.readLine();
                                         System.out.print("Enter password:");
                                        pword=br.readLine();
                                      pstmt=con.prepareStatement("insert into Employee
values(?,?)");
                                        pstmt.setString(1,uname);
                                        pstmt.setString(2,pword);
                                        pstmt.execute();
                                        System.out.println("\nRecord Inserted
successfully.");
                                    break;
                                    case 2:
                                System.out.print("Enter Employee Name to search :");
                                        uname=br.readLine();

                                        stmt=con.createStatement();
          rs=stmt.executeQuery("select *from Employee where username=+'uname'");
                                           rs.next();
                                               uname=rs.getString(1);
                                               pword=rs.getString(2);
```

```
                                        System.out.println(uname+"\t"+pword);
                                    break;
                                    case 3:
                                        System.out.println("Enter User Name to
update :");
                                        uname=br.readLine();
                                        System.out.println("Enter new password
:");
                                        pword=br.readLine();
                                        stmt=con.createStatement();
                                        count=stmt.executeUpdate("update
Employee set password='"+pword+"'where username='"+uname+"'");
                                        System.out.println("\n"+count+" Record
Updated.");
                                    break;
                                    case 4: System.out.println("Enter User Name to
delete record:");
                                        uname=br.readLine();
                                        stmt=con.createStatement();
                                        count=stmt.executeUpdate("delete from
Employee where username='"+uname+"'");


                                        if(count!=0)
                                            System.out.println("\nRecord
"+uname+" has deleted.");
                                        else
                                            System.out.println("\nInvalid
USN, Try again.");
                                    break;

                                    case 5: con.close(); System.exit(0);
                                    default: System.out.println("Invalid choice, Try
again.");
                                }//close of switch
                            }while(true);
                            }//close of nested try
                            catch(SQLException e2)
                            {
                                System.out.println(e2);
                            }
                            catch(IOException e3)
                            {
                                System.out.println(e3);
                            }
                        }//close of outer try
                        catch(ClassNotFoundException e1)
                        {
                            System.out.println(e1);
```

```
                    }
                }
}
```

5. Discuss built-in annotations with example program

Built-In Java Annotations

There are several built-in annotations in java. Some annotations are applied to java code and some to other annotations.

Built-In Java Annotations used in java code

@Override

@SuppressWarnings

@Deprecated

Built-In Java Annotations used in other annotations

@Target

@Retention

@Inherited

@Documented

@Override

@Override annotation assures that the subclass method is overriding the parent class method. If it is not so, compile time error occurs.

Sometimes, we does the silly mistake such as spelling mistakes etc. So, it is better to mark @Override annotation that provides assurity that method is overridden. class Animal{

```
void eatSomething(){System.out.println("eating something");}
}
class Dog extends Animal{
@Override
void eatsomething(){System.out.println("eating foods");}//should be
eatSometh ing
}
class TestAnnotation1{
public static void main(String args[]){
Animal a=new Dog();
a.eatSomething();
}}
```

Output:

Comple Time Error

@SuppressWarnings

@SuppressWarnings annotation: is used to suppress warnings issued by the compiler. import java.util.*;

```
class TestAnnotation2{
@SuppressWarnings("unchecked")
public static void main(String args[]){
ArrayList list=new ArrayList();
list.add("sonoo");
list.add("vimal");
list.add("ratan");
for(Object obj:list)
System.out.println(obj);
}
}
```

Now no warning at compile time.

If you remove the @SuppressWarnings("unchecked") annotation, it will show warning at compile time because we are using non-generic collection.

@Deprecated

@Deprecated annoation marks that this method is deprecated so compiler prints warning. It informs user that it may be removed in the future versions. So, it is better not to use such methods.

```
class A{
void m(){
System.out.println("hello m");}
 @Deprecated
void n(){System.out.println("hello n");}
}
 class TestAnnotation3{
public static void main(String args[]){
 A a=new A();
a.n();
}}
```

At Compile Time:

Note: Test.java uses or overrides a deprecated API.

Note: Recompile with -Xlint:deprecation for details.

At Runtime:

hello n

@Target

@Target tag is used to specify at which type, the annotation is used.

The java.lang.annotation.ElementType enum declares many constants to specify the type of element where annotation is to be applied such as TYPE, METHOD, FIELD etc.

6. Explain in detail updatable and scrollable result set with an example program.

In JDBC 2.1 API the virtual cursor can be moved backwards or positioned at a specific row.

 Six methods are there for Resultset object.
 They are first(), last(), previous(), absolute(), relative() and getrow().
 first() Moves the virtual cursor to the first row in the Resultset.
 last() Positions the virtual cursor at the last row in the Resultset
 previous() Moves the virtual cursor to the previous row.
 absolute() Positions the virtual cursor to a specified row by the an integer value passed to the method.
 relative() Moves the virtual cursor the specified number of rows contained in the parameter. The parameter can be positive or negative integer.
 getRow() Returns an integer that represents the number of the current row in the Resultset.
 To handle the scrollable ResultSet , a constant value is passed to the Statement object that is created using the createStatement(). Three constants.

TYPE_FORWARD_ONLY restricts the virtual cursor to downward movement
TYPE_SCROLL_INSENSITIVE and TYPE_SCROLL_SENSITIVE (Permits the virtual cursor to Move in any direction)

```
try {
String query = "SELECT FirstName,LastName FROM Customers";
Statement stmt;
ResultSet rs;
```

```
stmt = con.createStatement();
rs = stmt.executeQuery (query);
while(rs.next()){
rs.first();
rs.previous();
rs.absolute(10);
rs.relative(-2);
rs.relative(2);
System.out.println(rs.getString(1) + rs. getString (2));
}
stmt.close();}catch ( Exception e ){}
```

Update ResultSet

 Once the executeQuery() of the Statement object returns a ResultSet, the updatexxx() is used to change the value of column in the current row of the ResultSet.
 The xxx in the updatexxx() is replaced with the data type of the column that is to be updated. Note: updatexxx() updateString(), updateInt()
 The updatexxx() requires two parameters. The first is either the number or name of the column of the ResultSet that is being updated and the second is the value that will replace the value in the column of the ResultSet.
 A value in a column of the ResultSet can be replaced with a NULL value by using the updateNull().
 It requires one parameter, which is the number of column in the current row of the ResultSet. The updateNull() don‟t accept name of the column as a parameter.
Note The updateRow() is called after all the updatexxx() are called.

Delete Row in the ResultSet
 The deleteRow() is used to remove a row from a ResultSet.
 The deleteRow() is passed an integer that contains the number of the row to be deleted.
 First use the absolute() method to move the virtual cursor to the row in the Resultset that should be deleted.
 The value of that row should be examined by the program to assure it is the proper row before the deleteRow() is called.
 The deleteRow() is then passed a zero integer indicating that the current row must be deleted.

```
rs.deleteRow(0);

try {
String query = "select * from customers where firstname = 'mary' and lastname = 'jones'";
stmt = con.createStatement(rs.CONCUR_UPDATABLE);
rs = stmt.executeQuery (query);
}
catch ( SQLException error ){System.out.println(error)}
try {
rs.updateString ("LastName", "Smith");
rs.updateRow();
con.close();
} catch ( SQLException e ){System.out.println(e)}
```

7. What is Enum? Explain the use of Values() and valueOf() methods with suitable examples?

Enumerations was added to Java language in JDK5. **Enumeration** means a list of named constant. In Java, enumeration defines a class type. An Enumeration can have constructors, methods and instance variables. It is created using enum keyword. Each enumeration constant is *public*, *static* and *final* by default.

Values( ) and ValueOf( ) method

All the enumerations has predefined methods **values()** and **valueOf()**. values() method returns an array of enum-type containing all the enumeration constants in it. Its general form is,

public **static** *enum-type[ ]* **values()**

valueOf() method is used to return the enumeration constant whose value is equal to the string passed in as argument while calling this method. It's general form is,

public **static** *enum-type* **valueOf** (String *str*)

Example of enumeration using values() and valueOf() methods:

```
enum Restaurants {
dominos, kfc, pizzahut, paninos, burgerking
}
class Test {
public static void main(String args[])
{
Restaurants r;
System.out.println("All constants of enum type Restaurants are:");
Restaurants rArray[] = Restaurants.values(); //returns an array of constants of type Restaurants
for(Restaurants a : rArray) //using foreach loop
System.out.println(a);

r = Restaurants.valueOf("dominos");
System.out.println("I AM " + r);
}
}
```

Output:

```
All constants of enum type Restaurants are:
dominos

kfc pizzahut
paninos
burgerking
I AM dominos
```

8. What are Prepared Statement object and Callable Statement? How do we use themExplain with an example.

The preparedStatement object allows you to execute parameterized queries. A SQL query can be precompiled and executed by using the PreparedStatement object. · Ex: Select * from publishers where pub_id=?
Here a query is created as usual, but a question mark is used as a placeholder for a value· that is inserted into the query after the query is compiled.
The preparedStatement() method of Connection object is called to return the· PreparedStatement object.
Ex: PreparedStatement stat; stat= con.prepareStatement(“select * from publisher where pub_id=?”)

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

```
public class JdbcDemo {

public static void main(String args[]){

try{

Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");

Connection con=DriverManager.getConnection("jdbc:odbc:MyDataSource","khutub","");

PreparedStatement pstmt;

pstmt= con.prepareStatement("select * from employee whereUserName=?");

pstmt.setString(1,"khutub");

ResultSet rs1=pstmt.executeQuery();

while(rs1.next()){

System.out.println(rs1.getString(2));

}

} // end of try

catch(Exception e){System.out.println("exception"); }

} //end of main

} // end of class
```

Callable Statement:

The CallableStatement object is used to call a stored procedure from within a J2EE object.
A Stored procedure is a block of code and is identified by a unique name.
The type and style of code depends on the DBMS vendor and can be written in PL/SQL Transact-SQL, C, or other programming languages.
IN, OUT and INOUT are the three parameters used by the CallableStatement object to call a stored procedure.

The IN parameter contains any data that needs to be passed to the stored procedure and· whose value is assigned using the setxxx() method.
The OUT parameter contains the value returned by the stored procedures. The OUT parameters must be registered using the registerOutParameter() method, later retrieved by using the getxxx()
The INOUT parameter is a single parameter that is used to pass information to the stored procedure and retrieve information from the stored procedure.

```
Connection con;

try{
```

```
String query = "{CALL LastOrderNumber(?)}";

CallableStatement stat = con.prepareCall(query);

stat.registerOutParameter( 1 ,Types.VARCHAR);

stat.execute();

String lastOrderNumber = stat.getString(1);

stat.close();

}

catch (Exception e){}
```

9. Write a java program to insert 10 records in a table using batch updates.

```
public class Transactions {

	public static void main(String[] args) {
		// TODO Auto-generated method stub
		String dburl="jdbc:mysql://127.0.0.1/mca?autoReconnect=true&useSSL=false";
		String dbuser="root";
		String dbpass="cmrit";
		Connection con = null;
		Statement stmt=null;
		try {
			//Step 1 : Connecting to server and database

		    con = DriverManager.getConnection(dburl, dbuser, dbpass);
		    stmt=con.createStatement();
			con .setAutoCommit(false);
		    //Step 3 : SQL Query
		    String query1="INSERT INTO stu values(301,'xxx')";
		    String query2="INSERT INTO stu values(302,'yyy')";
		    String query3="INSERT INTO stu values(303,'zzz')";
		    String query4="INSERT INTO stu values(304,'aaa')";
		    String query5="INSERT INTO stu values(305,'bbb')";
String query6="INSERT INTO stu values(306,'bbb')";
String query7="INSERT INTO stu values(307,'bbb')";
String query8="INSERT INTO stu values(308,'bbb')";
String query9="INSERT INTO stu values(309,'bbb')";
String query10="INSERT INTO stu values(310,'bbb')";

		    stmt.addBatch(query1);
		    stmt.addBatch(query2);
		    stmt.addBatch(query3);
		    stmt.addBatch(query4);
		    stmt.addBatch(query5);
		    stmt.addBatch(query6);
		    stmt.addBatch(query7);
		    stmt.addBatch(query8);
		    stmt.addBatch(query9);
		    stmt.addBatch(query10);
```

```
                    //Step 5 : Execute Batch
                    stmt.executeBatch();

                        con.commit();
                        stmt.close();



                        con.close();     }
                        catch(SQLException ex) {

                        }
        }
    }
```

10. **What is boxing and unboxing? Explain how autoboxing and unboxing occurs inBoolean and Character values?**

- Autoboxing and Unboxing features was added in Java5.
- **Autoboxing** is a process by which primitive type is automatically encapsulated(boxed) into its equivalent type wrapper
- **Auto-Unboxing** is a process by which the value of an object is automatically extracted from a type Wrapper class.

Benefits of Autoboxing / Unboxing

1. Autoboxing / Unboxing lets us use primitive types and Wrapper class objects interchangeably.
2. We don't have to perform Explicit **typecasting**.
3. It helps prevent errors, but may lead to unexpected results sometimes. Hence must be used with care.
4. Auto-unboxing also allows you to mix different types of numeric objects in an expression. When the values are unboxed, the standard type conversions can be applied.

```
// Autoboxing/unboxing a Boolean and Character.
class AutoBox5 {

  public static void main(String args[]) {
  // Autobox/unbox a boolean. Boolean
  b = true;
  // Below, b is auto-unboxed when used in
  // a conditional expression, such as an if.
   if(b) System.out.println("b is true");
  // Autobox/unbox a char. Character
  ch = 'x'; // box a char char ch2 = ch;
  // unbox a char
  System.out.println("ch2 is " + ch2);
  }
  }
```

The output is shown here:

```
b is true ch2 is
x
```

the conditional expression that controls an if must evaluate to type boolean. Because of auto-unboxing, the boolean value contained within b is automatically unboxed when the conditional expression is

evaluated. Thus, with the advent of autoboxing/unboxing, a Boolean object can be used to control an if statement.