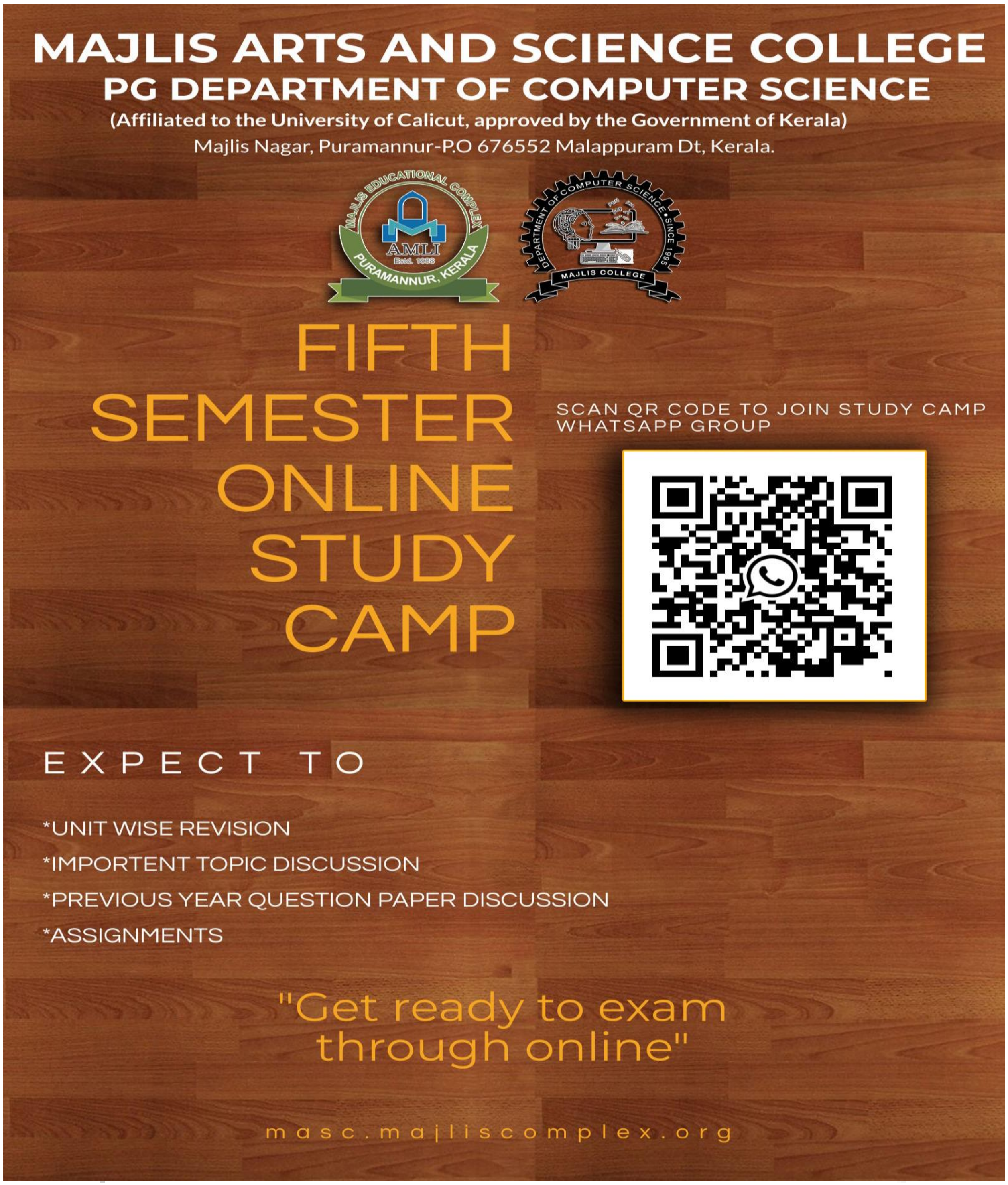


MAJLIS ARTS AND SCIENCE COLLEGE
PG DEPARTMENT OF COMPUTER SCIENCE
(Affiliated to the University of Calicut, approved by the Government of Kerala)
Majlis Nagar, Puramannur-P.O 676552 Malappuram Dt, Kerala.

FIFTH SEMESTER ONLINE STUDY CAMP

SCAN QR CODE TO JOIN STUDY CAMP WHATSAPP GROUP

EXPECT TO

*UNIT WISE REVISION
*IMPORTENT TOPIC DISCUSSION
*PREVIOUS YEAR QUESTION PAPER DISCUSSION
*ASSIGNMENTS

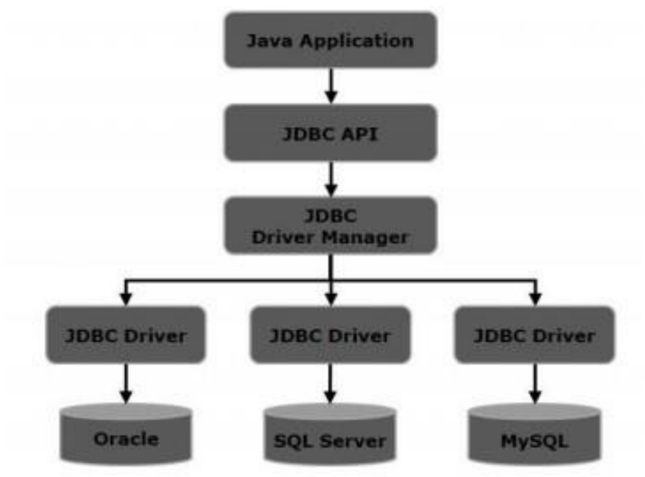
"Get ready to exam through online"

masc.majliscomplex.org

JAVA PROGRAMMING

MODULE 4

1. What is JDBC?
   JDBC (Java Database Connectivity) is an SQL-based API created by Sun Microsystems to enable Java applications to use SQL for database access.
   The JDBC API defines a set of Java interfaces that encapsulate major database functionality, such as running queries, processing results, and determining configuration information.
   These API provide communications between an application residing on a client machine and a data source residing on the same client machine or on another server computer.
2. Briefly explain the JDBC components.
   OR
   Explain JDBC Architecture.
   To communicate with the Database using JDBC need the following components.
   JDBC DriverManager: The DriverManager class of the java.sql package manages different types of JDBC drivers. This class loads the driver classes. In addition to this, whenever a new connection establishes it chooses and loads the suitable driver from the previously loaded ones.
   JDBC API: It is a Java abstraction which enables applications to communicate with relational databases. It provides two main packages namely, java.sql and javax.sql. It provides classes and methods to connect with a database, create statements (quires), execute statements and handle the results.
   JDBC-ODBC Bridge driver: This is a bridge driver which translates the JDBC method calls to ODBC function calls using this package you can communicate with the databases which uses ODBC drivers.



   Driver: This interface handles the communications with the database server. You will interact directly with Driver objects very rarely. Instead, you use DriverManager objects, which manages objects of this type. It also abstracts the details associated with working with Driver objects.

Connection: This interface with all methods for contacting a database. The connection object represents communication context, i.e., all communication with database is through connection object only.
Statement: You use objects created from this interface to submit the SQL statements to the database. Some derived interfaces accept parameters in addition to executing stored procedures.
ResultSet: These objects hold data retrieved from a database after you execute an SQL query using Statement objects. It acts as an iterator to allow you to move through its data.
SQLException: This class handles any errors that occur in a database application.

3. Explain the different steps in JDBC for connecting any database as back end of java program.

Java Database Connectivity Steps-
First import the java.sql package

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

The star '*' indicates that all of the classes in the java.sql and java.io packages are to be imported.
Loading a database driver
In this step, the driver class loaded by calling Class.forName() with the Driver class. Class is a class in java.lang package.
Syntax: Class.forName("com.mysql.jdbc.Driver");
forName() is static method of the "Class" class . This loads the driver class and returns Class instance. It takes string type value as an argument and matches the class with string.

Creating a jdbc Connection

The objects defined by DriverManager class to establish the applications with the JDBC driver. This class manages the JDBC driverswhich is installed on the system. getConnection() is the method by which it can connect. It uses the username, password, and a jdbcurl to make the connection with database.
url - Database url where stored or created your database
userName - User name of MySQL
password -Password of MySQL

```
try{
  Connection con=DriverManager.getConnection(url,"userName","password");
}
catch( SQLException x ){
  System.out.println( "e.printStackTrace" );
}
```

Creating a jdbc Statement object

When an connection is established then we can interact with the database. Connection interface defines methods for interacting with the database. It used to instantiate a Statement by using the createStatement() method.

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

Executing a statement with the Statement object

This interface defines methods which is used to communicate with database. This class has three methods to execute the statements- executeQuery(), executeUpdate(), and execute(). For SELECT statements, executeQuery() method will be used. For create or modify tables, the executeUpdate() method is used. executeQuery() method returns the result of the query in the form of ResultSet object and executeUpdate() method returns the number of rows affected from execution of the query.

```
        ResultSetrs = st.executeQuery("SELECT * from Employee");
```

Getting ResultSet object

Executing the executeQuery() method returns the result in the form of ResultSet object. We can now operate on this object to extract the rows values returned from the execution of the query. Its next() method sets the pointer to the first row and getX() methods can be used to get different types of data from the result set.

```
while( rs.next() ) {
  String data = rs.getString(1);
  System.out.println( data );
}
```

con.close()

This method of Connection interface is used for closing the connection. All the resources will be free occupied by the database.

4. Explain SQL exception class.

The SQLException Class and its subtypes provide information about errors and warnings that occurs when the data source is accessed.

This class is an extension of java.lang.exception and provides information related to failure in database connectivity.

The following information is available from an SQLException:

Text Description, SQL State, Error Code, A reference to any other exception that also occur.

An SQLException can occur both in the driver and the database. When such an exception occurs, an object of type SQLException will be passed to the catch clause.

```
catch (SQLExceptione){

          System.out.println("SQLException exception: ");

          System.out.println("Message:....." + e.getMessage());

          System.out.println("SQLState:...." + e.getSQLState());

          System.out.println("Vendor Code:." + e.getErrorCode());

e.printStackTrace();  }
```

The passed SQLException object has the following methods available for retrieving additional information about the exception –

Method	**Description**
getErrorCode( )	Gets the error number associated with the exception.
getMessage( )	Gets the JDBC driver's error message for an error, handled by the driver or gets the Oracle error number and message for a database error.
getSQLState( )	Gets the XOPEN SQLstate string. For a JDBC driver error, no useful information is returned from this method. For a database error, the five-digit XOPEN SQLstate code is returned. This method can return null.For example: 08001 where 08 is the class and 001 is the subclass. 08 class means connection exception and 001 means client unable to establish connection.
getNextException( )	Gets the next Exception object in the exception chain.
printStackTrace( )	Prints the current exception, or throwable, and it's backtrace to a standard error stream.

5. Explain Statement interface.

The Statement interface provides the method to execute the database queries. After making a connection, Java application can interact with database. The Statement interface contains the ResultSet object.

Creating Statement Object

Statement st = con.createStatement();
ResultSetrs = st.executeQuery("select * from student");

The PreparedStatementinterface extends the Statement interface. It represents precompiled SQL statements and stores it in a PreparedStatement object.It increases the performance of the application because the query is compiled only once.ThePreparedStatement is easy to reuse with new parameters.

Creating PreparedStatement Object

String sql = "Select * from Student where rollNo= ?";

PreparedStatementps = con.prepareStatement(sql);   (or)

PreparedStatementps = con.prepareStatement("insert into Student values(?, ?, ?)");
ps.setInt(1, 101);
ps.setString(2, "Suraj");
ps.setString(3, "MCA");
ps.executeUpdate();

All the parameter are represented by "?" symbol and each parameter is referred to by its origin position.

The CallableStatementinterface is used to execute the SQL stored procedure in a database. The JDBC API provides stored procedures to be called in a standard way for all RDBMS.

A stored procedure works like a function or method in a class. The stored procedure makes the performance better because these are precompiled queries.

Creating CallableStatement Interface

The instance of a CallableStatement is created by calling prepareCall() method on a Connection object.

For example: CallableStatement cs = con.prepareCall("{call procedures(?,?)}");

6. ExplainResultSetMetaData?

The ResultSetMetaData provides information about the obtained ResultSet object like, the number of columns, names of the columns, datatypes of the columns, name of the table etc…Following are some ***methods of ResultSetMetaData class.***

Method	**Description**
getColumnCount()	Retrieves the number of columns in the current ResultSet object.
getColumnLabel()	Retrieves the suggested name of the column for use.

getColumnName()	Retrieves the name of the column.
getTableName()	Retrieves the name of the table.

The getMetaData() method of ResultSet interface returns the object of ResultSetMetaData.

The following code fragment creates the ResultSet object rs,creates the ResultSetMetaData object rsmd, and uses rsmd to find out how many columns rs has,column name and columntype.

```
Statement ps=con.Statement("select * from emp");
ResultSetrs=ps.executeQuery();
ResultSetMetaDatarsmd=rs.getMetaData();
System.out.println("Total columns: "+rsmd.getColumnCount());
System.out.println("Column Name of 1st column: "+rsmd.getColumnName(1));
System.out.println("Column Type Name of 1st column:
"+rsmd.getColumnTypeName(1));
```

7. Differentiate java application and applet.

Java Application	**Java Applet**
Applications are just like a Java programs that can be execute independently without using the web browser.	Applets are small Java programs that are designed to be included with the HTML web document. They require a Java-enabled web browser for execution.
Application program requires a main function for its execution.	Applet does not require a main function for its execution.
Java application programs have the full access to the local file system and network.	Applets don’t have local disk and network access.
Applications can access all kinds of resources available on the system.	Applets can only access the browser specific services. They don’t have access to the local system.
Applications can executes the programs from the local system.	Applets cannot execute programs from the local machine.

Java Application	Java Applet
An application program is needed to perform some task directly for the user.	An applet program is needed to perform small tasks or the part of it.

8. What is applet?

   An applet is a Java program that can be embedded into a web page. It runs inside the web browser and works at client side. An applet is embedded in an HTML page using the APPLET or OBJECT tag and hosted on a web server. Applets are used to make the web site more dynamic and entertaining.

9. Explain the life cycle of an applet with a diagram.

   OR

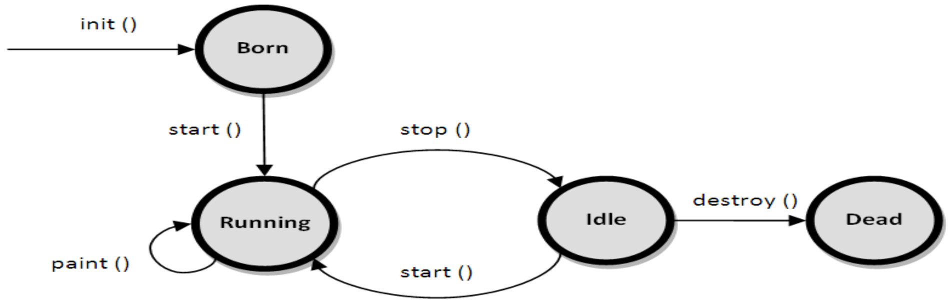
   Discuss different stages in the life cycle of an applet

   OR

   Explain the life cycle of an applet.

   When an applet is executed within the web browser or in an applet window, it goes through the four stages of its life cycle: Born, Running, Idle and Dead. These stages correspond to the applet methods init(), start(), stop() and destroy() respectively. All these methods defined in the Applet class which is called automatically by the browser or the applet viewer controlling the applet.

   The life cycle of an applet is as shown in the figure below:



   As shown in the above diagram, the life cycle of an applet starts with init() method and ends with destroy() method. Other life cycle methods are start(), stop() and paint(). The methods to execute only once in the applet life cycle are init() and destroy(). Other methods execute multiple times.

   init(): The init() method is the first method to execute when the applet is executed. Variable declaration and initialization operations are performed in this method.

   start(): The start() method contains the actual code of the applet that should run. The start() method executes immediately after the init() method. It also executes whenever the applet is restored, maximized or moving from one tab to another tab in the browser.

stop(): The stop() method stops the execution of the applet. The stop() method executes when the applet is minimized or when moving from one tab to another in the browser.
destroy(): The destroy() method executes when the applet window is closed or when the tab containing the webpage is closed. stop () method executes just before when destroy() method is invoked. The destroy () method removes the applet object from memory.
paint(): The paint() method is used to redraw the output on the applet display area. The paint() method executes after the execution of start() method and whenever the applet or browser is resized.

10. Write about applet skeleton.

An Applet Skeleton
Most applets override these four methods. These four methods forms Applet lifecycle.
init() : init() is the first method to be called. This is where variable are initialized. This method is called only once during the runtime of applet.
start() : start() method is called after init(). This method is called to restart an applet after it has been stopped.
stop() : stop() method is called to suspend thread that does not need to run when applet is not visible.
destroy() : destroy() method is called when your applet needs to be removed completely from memory.

Example of an Applet Skeleton

```
import java.awt.*;
import java.applet.*;
public class AppletTest extends Applet
{
 public void init()
 {
  //initialization
System.out.println("Applet initialized");
 }
 public void start ()
 {
  //start or resume execution
System.out.println("Applet execution started");
 }
 public void stop()
 {
  //suspend execution
System.out.println("Applet execution stopped");
 {
 public void destroy()
 {
  //perform shutdown activity
{ System.out.println("Applet destroyed");
 }
 public void paint (Graphics g)
 {
  //display the content of window
System.out.println("Painting...");
 }
}
```

The method execution sequence when an applet is executed is:

•init()

•start()

•paint()

The method execution sequence when an applet is closed is: •stop() •destroy()

11. What is an applet? Explain its working with examples.

An applet is a special kind of Java program that runs in a Java enabled browser. This is the first Java program that can run over the network using the browser. Applet is typically embedded inside a web page and runs in the browser.

Example of an Applet

```
import java.applet.*;
import java.awt.*;
public class MyApplet extends Applet
{
 int height, width;
 public void init()
 {
  height = getSize().height;
  width = getSize().width;
setName("MyApplet");
 }
 public void paint(Graphics g)
 {
g.drawRoundRect(10, 30, 120, 120, 2, 3);
 }
}
```

Every Applet application must import two packages - java.awt and java.applet.

- java.awt.* imports the Abstract Window Toolkit (AWT) classes. Applets interact with the user (either directly or indirectly) through the AWT. The AWT contains support for a window-based, graphical user interface.
- java.applet.* imports the applet package, which contains the class Applet. Every applet that you create must be a subclass of Applet class.
- The class in the program must be declared as public, because it will be accessed by code that is outside the program.Every Applet application must declare a paint() method. This method is defined by AWT class and must be overridden by the applet. The paint() method is called each time when an applet needs to redisplay its output.
- An Applet class does not have any main() method. It is viewed using JVM. The JVM can use either a plug-in of the Web browser or a separate runtime environment to run an applet application.
- JVM creates an instance of the applet class and invokes init() method to initialize an Applet.

12. Discuss the steps involved in developing and running a local applet.

There are two ways to run an applet

- By html file.
- By appletViewer tool (for testing purpose).

Simple example of Applet by html file:

To execute the applet by html file, create an applet program and compile it. After that create an html file and place the applet code in html file. Now click the html file.

```
import java.applet.Applet;

import java.awt.Graphics;

public class First extends Applet{

  public void paint(Graphics g){

g.drawString("welcome",150,150);

}  }
```

myapplet.html

```
<html>

<body>

<applet code="First.class" width="300" height="300">
```

```
</applet>

</body>

</html>
```

Simple example of Applet by appletviewer tool:

To execute the applet by appletviewer tool, create an applet that contains applet tag in comment and compile it. After that run it by: appletviewer First.java. Now Html file is not required but it is for testing purpose only.

```
import java.applet.Applet;

import java.awt.Graphics;

public class First extends Applet{

public void paint(Graphics g){

g.drawString("welcome to applet",150,150);

}   }

/*

<applet code="First.class" width="300" height="300">

</applet>

*/
```

To execute the applet by appletviewer tool, write in command prompt:

```
c:\>javac First.java

c:\>appletviewer First.java
```

13. What is the use of paint() method?
The paint() method is used to redraw the output on the applet display area. The paint() method executes after the execution of start() method and whenever the applet or browser is resized.
The method paint() gives us access to an object of type Graphics class. Using the object of the Graphics class, we can call the method of the Graphics class to write a text or draw various shapes etcin the applet window.
14. Explain how you will pass parameters to applets with an example.
OR

How will you pass parameters to applets?

Parameters specify extra information that can be passed to an applet from the HTML page.
Parameters are specified using the HTML’s param tag.
Parameters are passed on an applet When it is loaded. we can define the init( )
Method in the applet to get hold of the parameter defined in the tags.

Param Tag

The <param> tag is a sub tag of the <applet> tag.
The <param> tag contains two attributes: name and value which are used to specify the name of the parameter and the value of the parameter respectively.
For example, the param tags for passing name and age parameters looks as shown below:

```
<param name=”name” value=”john” />
<param name=”age” value=”25″ />
```

getParameter() Method

The getParameter() method of the Applet class can be used to retrieve the parameters passed from the HTML page. The syntax of getParameter() method is as follows:
String getParameter(String param-name)

```
import java.awt.*;
import java.applet.*;
public class MyApplet extends Applet
{
        String n;
        String a;
        public void init()
        {
                n = getParameter("name");
                a = getParameter("age");
        }
        public void paint(Graphics g)
        {
                g.drawString("Name is: " + n, 20, 20);
                g.drawString("Age is: " + a, 20, 40);
        }
}
/*
        <applet code="MyApplet" height="300" width="500">
                <param name="name" value="John" />
                <param name="age" value="25" />
        </applet>
*/
```