CMR
INSTITUTE OF
TECHNOLOGY

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

Internal Assessment Test 2– Feb. 2024

Sub:	**Advanced Java& J2EE**							**Sub Code:**	**22MCA341**
Date:	**16/2/2024**	**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	With an example program Explain the mechanism of Session Tracking in Servlets?	**10**	CO1	L2
	OR			
2.	Write a JAVA Servlet Program to Auto Web Page Refresh (Consider a webpage which is displaying Date and time)	**10**	CO2	L4
	PART II			
3	Describe the classes and interfaces of javax.servlet package.	**10**	CO1	L2
	OR			
4.	Explain how do you perform searching Strings along with an example program.	**10**	CO1	L2

CMR
INSTITUTE OF
TECHNOLOGY

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

Internal Assessment Test 2– Feb. 2024

Sub:	**Advanced Java& J2EE**							**Sub Code:**	**22MCA341**
Date:	**16/2/2024**	**Duration:**	**90 min's**	**Max Marks:**	**50**	**Sem:**	**III**	**Branch:**	**MCA**

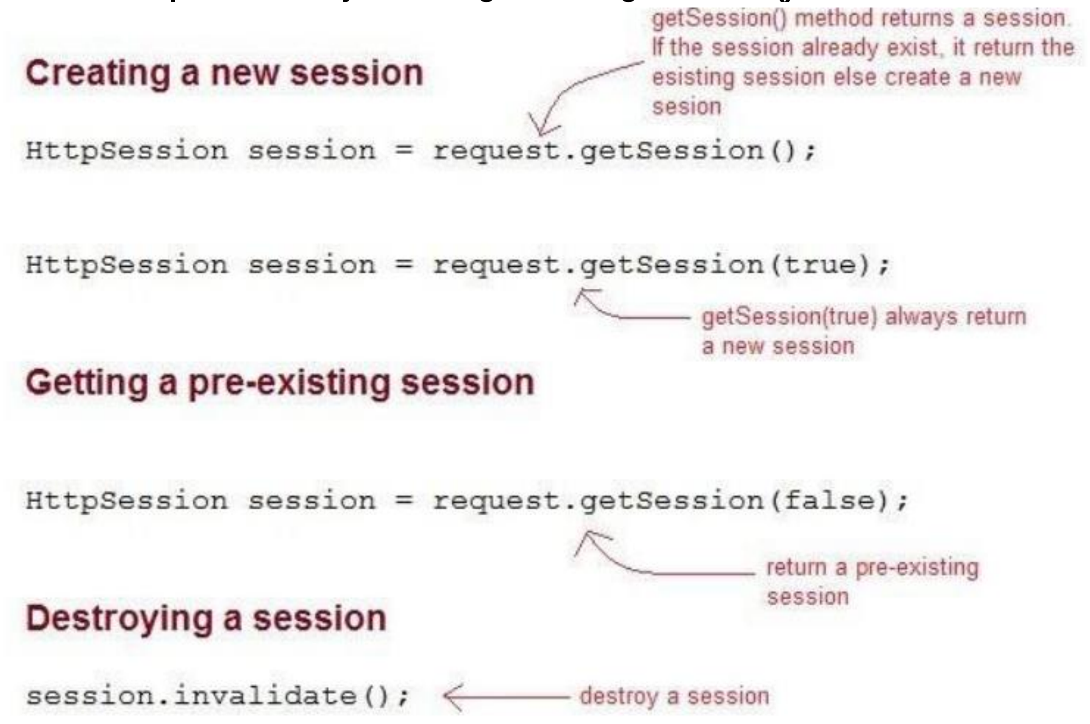
Note : Answer FIVE FULL Questions, choosing ONE full question from each Module

		MARKS	OBE	
			CO	RBT
	PART I			
1	With an example program Explain the mechanism of SessionTracking in Servlets? **OR**	**10**	CO1	L2
2.	Write a JAVA Servlet Program to Auto Web Page Refresh (Consider a webpage which is displaying Date and time)	**10**	CO2	L4
	PART II			
3	Describe the classes and interfaces of javax.servlet package.	**10**	CO1	L2
	OR			
4.	Explain how do you perform searching Strings along with an example program.	**10**	CO1	L2

	PART III			
5	What is JSP? What are different JSP tags demonstrate with an example.	**10**	CO1	L2
	OR			
6	Write a Servlet program to read data from a HTML form (gender data from radio buttons and colours data from check boxes) and display	**10**	CO2	L4
7	What is Cookie? Explain creation of Cookie and retrieving information from cookie using code snippets?	**10**	CO1	L2
	OR			
8	Write a java program to check whether a string is palindrome or not	**10**	CO2	L4
	PART V			
9	What is String? Explain all String Constructors available with code snippets.	**10**	CO1	L2
	OR			
10	Explain the life cycle of a string.	**10**	CO1	L2

	PART III			
5	What is JSP? What are different JSP tags demonstrate with an example.	**10**	CO1	L2
	OR			
6	Write a Servlet program to read data from a HTML form (gender data from radio buttons and colours data from check boxes) and display	**10**	CO2	L4
7	What is Cookie? Explain creation of Cookie and retrieving information from cookie using code snippets?	**10**	CO1	L2
	OR			
8	Write a java program to check whether a string is palindrome or not	**10**	CO2	L4
	PART V			
9	What is String? Explain all String Constructors available with code snippets.	**10**	CO1	L2
	OR			
10	Explain the life cycle of a string.	**10**	CO1	L2

1. **With an example program Explain the mechanism of Session Tracking in Servlets?**
   HttpSession object is used to store entire session with a specific client. We can store, retrieve and remove attribute from HttpSession object. Any servlet can have access to HttpSession object throughout the getSession() method



Creating a new session

```
HttpSession session = request.getSession();
```

getSession() method returns a session. If the session already exist, it return the esisting session else create a new sesion

```
HttpSession session = request.getSession(true);
```

getSession(true) always return a new session

Getting a pre-existing session

```
HttpSession session = request.getSession(false);
```

return a pre-existing session

Destroying a session

```
session.invalidate();
```

Some Important Methods of HttpSession

Methods	Description
long getCreationTime()	returns the time when the session was created, measured in milliseconds since midnight January 1, 1970 GMT.
String getId()	returns a string containing the unique identifier assigned to the session.
int getMaxInactiveInterval()	returns the maximum time interval, in seconds.
void invalidate()	destroy the session
boolean isNew()	returns true if the session is new else false

Complete Example demonstrating usage of HttpSession

index.html

```
<form method="post" action="Validate">
  User: <input type="text" name="uname " /><br/>
    <input type="submit" value="submit">
</form>
```

Validate.java

```
public class Validate extends HttpServlet {
protected void doPost(request, response)
{
       // . . . .
        String name = request.getParameter("user");
                    //creating a session
            HttpSession session =
            request.getSession();
            session.setAttribute("use
            r", uname);
            response.sendRedirect("
            Welcome");

        }
}
```

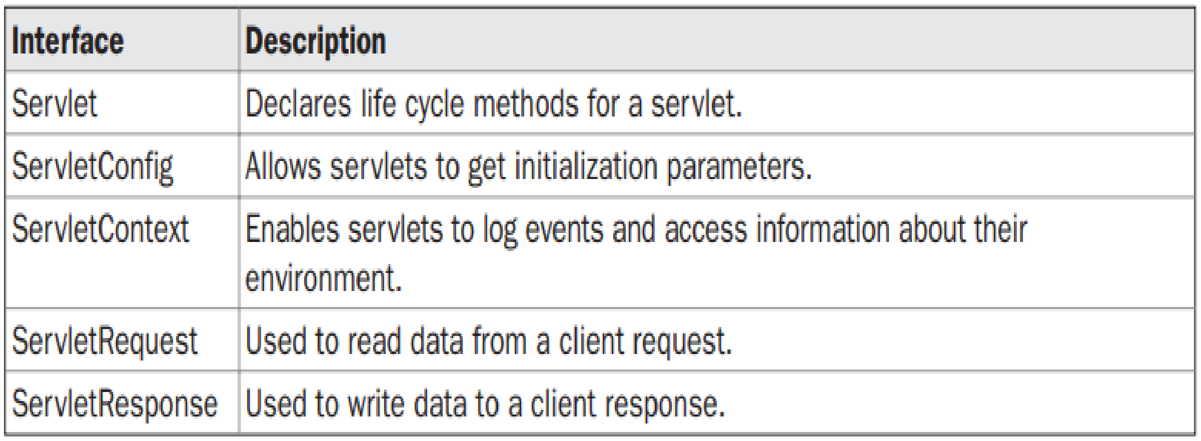
Welcome.java

```
public class Welcome extends HttpServlet {



    protected void doGet(request, response){
         // . . . .
        HttpSession session = request.getSession();
        String user =
        (String)session.getAttribute("user"
        ); out.println("Hello "+user);

      }

    }
```

2. **Write a JAVA Servlet Program to Auto Web Page Refresh (Consider a webpage which is displaying Date and time)**

```
@WebServlet("/progra m2")
public class program2 extends HttpServlet {
 protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException
{
response.setContentType("text/html");
response.addHeader("Refresh","1");
 PrintWriter out=response.getWriter();
out.println("Text servlet says hi at "+new Date());
 }
 }
```

3. **Describe the classes and interfaces of javax.servlet package.**



Interface	Description
Servlet	Declares life cycle methods for a servlet.
ServletConfig	Allows servlets to get initialization parameters.
ServletContext	Enables servlets to log events and access information about their environment.
ServletRequest	Used to read data from a client request.
ServletResponse	Used to write data to a client response.

Class	Description
GenericServlet	Implements the **Servlet** and **ServletConfig** interfaces.
ServletInputStream	Provides an input stream for reading requests from a client.
ServletOutputStream	Provides an output stream for writing responses to a client.
ServletException	Indicates a servlet error occurred.
UnavailableException	Indicates a servlet is unavailable.

Method	Description
void destroy( )	Called when the servlet is unloaded.
ServletConfig getServletConfig( )	Returns a **ServletConfig** object that contains any initialization parameters.
String getServletInfo( )	Returns a string describing the servlet.
void init(ServletConfig *sc*) throws ServletException	Called when the servlet is initialized. Initialization parameters for the servlet can be obtained from *sc*. An **UnavailableException** should be thrown if the servlet cannot be initialized.
void service(ServletRequest *req*, ServletResponse *res*) throws ServletException, IOException	Called to process a request from a client. The request from the client can be read from *req*. The response to the client can be written to *res*. An exception is generated if a servlet or IO problem occurs.

TABLE 31-1 The Methods Defined by **Servlet**

The ServletConfig Interface

The **ServletConfig** interface allows a servlet to obtain configuration data when it is loaded. The methods declared by this interface are summarized here:

Method	Description
ServletContext getServletContext( )	Returns the context for this servlet.
String getInitParameter(String *param*)	Returns the value of the initialization parameter named *param*.
Enumeration getInitParameterNames( )	Returns an enumeration of all initialization parameter names.
String getServletName( )	Returns the name of the invoking servlet.

Method	Description
Object getAttribute(String *attr*)	Returns the value of the server attribute named *attr*.
String getMimeType(String *file*)	Returns the MIME type of *file*.
String getRealPath(String *vpath*)	Returns the real path that corresponds to the virtual path *vpath*.
String getServerInfo( )	Returns information about the server.
void log(String *s*)	Writes *s* to the servlet log.
void log(String *s*, Throwable *e*)	Writes *s* and the stack trace for *e* to the servlet log.
void setAttribute(String *attr*, Object *val*)	Sets the attribute specified by *attr* to the value passed in *val*.

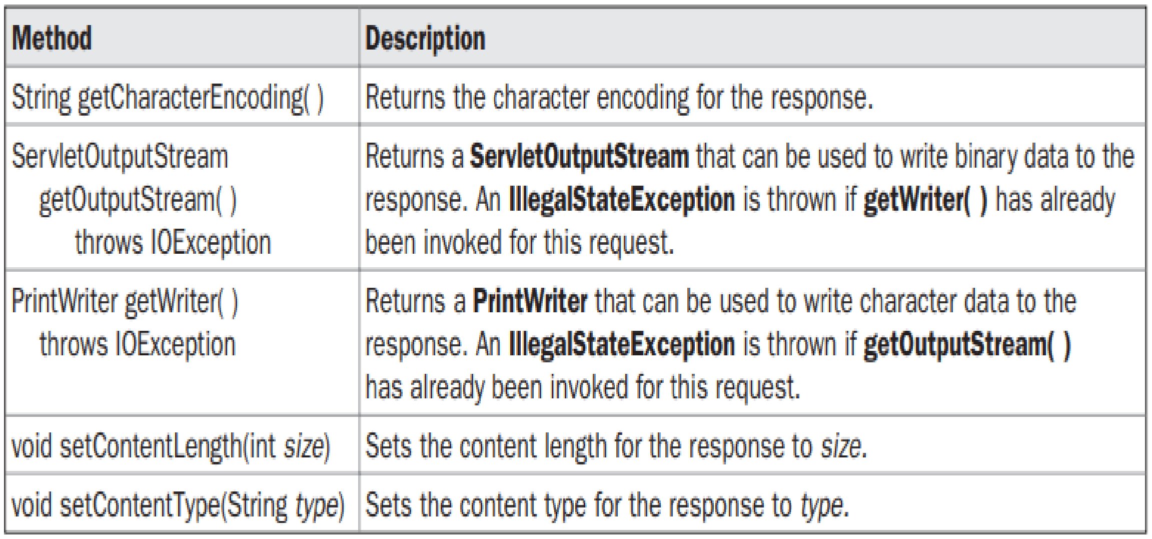
TABLE 31-2 Various Methods Defined by **ServletContext**

Method	Description
Object getAttribute(String *attr*)	Returns the value of the attribute named *attr*.
String getCharacterEncoding( )	Returns the character encoding of the request.
int getContentLength( )	Returns the size of the request. The value –1 is returned if the size is unavailable.
String getContentType( )	Returns the type of the request. A **null** value is returned if the type cannot be determined.
ServletInputStream getInputStream( ) throws IOException	Returns a **ServletInputStream** that can be used to read binary data from the request. An **IllegalStateException** is thrown if **getReader( )** has already been invoked for this request.
String getParameter(String *pname*)	Returns the value of the parameter named *pname*.
Enumeration getParameterNames( )	Returns an enumeration of the parameter names for this request.
String[ ] getParameterValues(String *name*)	Returns an array containing values associated with the parameter specified by *name*.
String getProtocol( )	Returns a description of the protocol.
BufferedReader getReader( ) throws IOException	Returns a buffered reader that can be used to read text from the request. An **IllegalStateException** is thrown if **getInputStream( )** has already been invoked for this request.
String getRemoteAddr( )	Returns the string equivalent of the client IP address.
String getRemoteHost( )	Returns the string equivalent of the client host name.
String getScheme( )	Returns the transmission scheme of the URL used for the request (for example, "http", "ftp").
String getServerName( )	Returns the name of the server.
int getServerPort( )	Returns the port number.

TABLE 31-3 Various Methods Defined by **ServletRequest**

Method	Description
String getCharacterEncoding( )	Returns the character encoding for the response.
ServletOutputStream getOutputStream( ) throws IOException	Returns a **ServletOutputStream** that can be used to write binary data to the response. An **IllegalStateException** is thrown if **getWriter( )** has already been invoked for this request.
PrintWriter getWriter( ) throws IOException	Returns a **PrintWriter** that can be used to write character data to the response. An **IllegalStateException** is thrown if **getOutputStream( )** has already been invoked for this request.
void setContentLength(int *size*)	Sets the content length for the response to *size*.
void setContentType(String *type*)	Sets the content type for the response to *type*.

TABLE 31-4 Various Methods Defined by **ServletResponse**



Method	Description
String getCharacterEncoding( )	Returns the character encoding for the response.
ServletOutputStream getOutputStream( ) throws IOException	Returns a **ServletOutputStream** that can be used to write binary data to the response. An **IllegalStateException** is thrown if **getWriter( )** has already been invoked for this request.
PrintWriter getWriter( ) throws IOException	Returns a **PrintWriter** that can be used to write character data to the response. An **IllegalStateException** is thrown if **getOutputStream( )** has already been invoked for this request.
void setContentLength(int *size*)	Sets the content length for the response to *size*.
void setContentType(String *type*)	Sets the content type for the response to *type*.

TABLE 31-4 Various Methods Defined by **ServletResponse**

Generic Servlet Class:

- The GenericServlet class provides implementations of the basic life cycle methods for a servlet.
- GenericServlet implements the Servlet and ServletConfig interfaces. In addition, a method to append a string to the server log file is available.
- The signatures of this method are shown here:

  void log(String s)

  void log(String s, Throwable e)

Here, s is the string to be appended to the log, and e is an exception that occurred.

Servlet Input Stream:

- The ServletInputStream class extends InputStream.
- It is implemented by the servlet container and provides an input stream that a servlet developer can use to read the data from a client request.
- It defines the default constructor.
- A method is provided to read bytes from the stream.

  int readLine(byte[ ] buffer, int offset, int size) throws IOException

ServletOutputStream:

- The ServletOutputStream class extends OutputStream.
- It is implemented by the servlet container and provides an output stream that a servlet developer can use to write data to a client response.
- A default constructor is defined.
- It also defines the print( ) and println( ) methods, which output data to the stream.

javax.servlet defines two exceptions.

- The first is ServletException, which indicates that a servlet problem has occurred.

- The second is UnavailableException, which extends ServletException. It indicates that a servlet is unavailable.

4. **Explain how do you perform searching Strings along with an example program.**

```
// Java Program to illustrate to Find a Substring
// in the String

// Importing required classes
import java.io.*;

// Main class
class GFG {

    // Main driver method
    public static void main(String[] args)
    {

        // A string in which a substring
        // is to be searched
        String str
            = "GeeksforGeeks is a computer science portal";

        // Returns index of first occurrence of substring
        int firstIndex = str.indexOf("Geeks");

        System.out.println("First occurrence of char Geeks"
                           + " is found at : "
                           + firstIndex);

        // Returns index of last occurrence
        int lastIndex = str.lastIndexOf("Geeks");
        System.out.println(
            "Last occurrence of char Geeks is"
            + " found at : " + lastIndex);

        // Index of the first occurrence
        // after the specified index if found
        int first_in = str.indexOf("Geeks", 10);
        System.out.println("First occurrence of char Geeks"
                           + " after index 10 : "
                           + first_in);
            int last_in = str.lastIndexOf("Geeks", 20);
            System.out.println("Last occurrence of char Geeks "
                                               + "after index 20 is : "
                                               + last_in);
    }
}
```

5. What is JSP? What are different JSP tags demonstrate with an example.

1.**<u>JSP scriptlet tag</u>** A scriptlet tag is used to execute java source code in JSP.

<% java source code %>

In this example, we are displaying a welcome message.

```
<html>
<body>
<% out.print("welcome to jsp"); %>
</body>

</html>
```

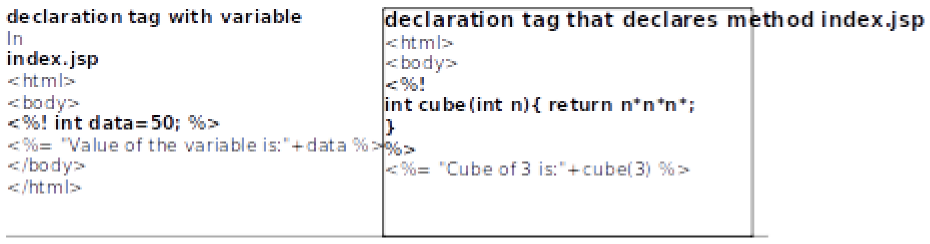
2. **<u>JSP Declaration Tag</u>**

The **JSP declaration tag** is used *to declare variables, objects and methods*.

The code written inside the jsp declaration tag is placed outside the service() method of auto generated servlet.

So it doesn't get memory at each request.

<%! field or method declaration %>



declaration tag with variable In index.jsp

```
<html>
<body>
<%! int data=50; %>
<%= "Value of the variable is:"+data %>
</body>
</html>
```

declaration tag that declares method index.jsp

```
<html>
<body>
<%!
int cube(int n){ return n*n*n*;
}
%>
<%= "Cube of 3 is:"+cube(3) %>
```

<u>JSP Expression Tag</u>

Expression Tag is used to print out java language expression that is put between the tags. An expression tag can hold any java language expression that can be used as an argument to the out.print() method.

Syntax of Expression Tag
<%= *JavaExpression* %>

<%= (2*5) %>   //note no ; at end of statement.

1. ## JSP directives

The jsp directives are messages that tells the web container how to translate a JSP page into the corresponding servlet.

Syntax <%@ directive attribute="value" %>
There are three types of directives:

1. **import directive**
2. **include directive**
3. **taglib directive**

4. JSP Comments

JSP comment marks text or statements that the JSP container should ignore. syntax of the JSP comments <%- - This is JSP comment - -%>

6. **Write a Servlet program to read data from a HTML form (gender data from radio buttons and colours data from check boxes) and display.**

```
<html>
<head>
<title>TODO supply a title</title>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
</head>
<body>
<!-- send the form data to url mapping “prg3“ and the get method is used -->
<form method ="post" action="prg3">
<!--Display 3 Colors RED, BLUE, GREEN in the dropdown Box -->
<h1> Select your colors</h1>
<input type="checkbox" name="color" value="red"/>RED</br>
<input type="checkbox" name="color" value="green"/>GREEN</br>
<input type="checkbox" name="color" value="blue"/>BLUE</br>
<h1> Select your Course</h1>
UG:<input type="radio" name="course" value="ug"/><br>
PG:<input type="radio" name="course" value="pg"/><br>
```

```
<input type="submit" value="Submit"/>
</form>
</body>
</html>
```

Prg3.java

```
import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
public class prg3 extends HttpServlet {
@Override
protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {

// Setting the HTTP Content-Type response header to text/html
response.setContentType("text/html");
// Returns a PrintWriter object out that can send character text to the client.
PrintWriter out=response.getWriter();
// To retrieve the optional values (color) from HTML page and store in the string color

String[] col = request.getParameterValues("color");
String cor = request.getParameter("course");
out.println("<html><body>");
out.println("Selected Colours");
for(String c:col)
out.println(c);

out.println("<br/>You have selected course "+cor);
out.println("</body></html>");
out.close();

}
}
```

7. **What is Cookie? Explain creation of Cookie and retrieving information from cookie using
code snippets?**

Cookies are small bits of textual information that a web server sends to a browser and that the
browser later returns unchanged when visiting the same web site or domain

Sending cookies to the client:

1.Creating a cookie object

• Cookie():constructs a cookie.
• Cookie(String name, String value)constructs a cookie with a specified name and value.
EX:
Cookie ck=new Cookie("user",”mca");
2.Setting the maximum age
setMaxAge() is used to specify how long (in seconds) the cookie should be valid.
Ex:cookie.setMaxAge(60*60*24);
3.Placing the cookie into the HTTP response headers.
We use response.addCookie to add cookies in the HTTP response header as follows:
response.addCookie(cookie);
Reading cookies from the client:
1. Call request.getCookies(). This yields an array of cookie objects.
2. Loop down the array, calling getName on each one until you find the cookie of interest.
Ex:
```
String cookieName=“userID”;
Cookie[] cookies=request.getCookies();
If(cookies!=null)
{
for(int i=0;i<cookies.length;i++){
Cookie cookie=cookies[i];
if(cookieName.equals(cookie.getName())){
doSomethingwith(cookie.getValue());
}}}
```
8. **Write a java program to check whether a string is palindrome or not**

```
// Java Program to implement
// Basic Approach to check if
// string is a Palindrome
import java.io.*;

// Driver Class
class GFG {
    // main function
    public static boolean isPalindrome(String str)
    {
            // Initializing an empty string to store the reverse
            // of the original str
            String rev = "";

            // Initializing a new boolean variable for the
            // answer
```

```
                boolean ans = false;

                for (int i = str.length() - 1; i >= 0; i--) {
                        rev = rev + str.charAt(i);
                }

                // Checking if both the strings are equal
                if (str.equals(rev)) {
                        ans = true;
                }
                return ans;
        }
        public static void main(String[] args)
        {
                // Input string
                String str = "geeks";

                // Convert the string to lowercase
                str = str.toLowerCase();
                boolean A = isPalindrome(str);
                System.out.println(A);
        }
    }
```

9. **What is String? Explain all String Constructors available with code snippets.**

- String is basically an object that represents sequence of char values. An array of characters works same as Java string.
- Java implements strings as objects of type String.

There are several constructors for String class.

1.To create an empty string, use default constructor:

```
String s= new String();
```

2. To create a string and initialize:

```
String s= new String("Hello");
```

3. To create a string object that contains same characters as another string object:

```
String(String strObj);
```

To create a string having initial values:

- **For example,**
- char ch[]={‘h’, ‘e’, ‘l’, ‘l’, ‘o’};
- String s= new String(ch); //s contains hello

5. To specify a sub-range of a character array as an initializer use the following constructor:

String(char chars[ ], int startIndex, int numChars)

- **For example**,
- char ch[]={‘a’, ‘b’, ‘c’, ‘d’, ‘e’, ‘f’’, ‘g’};
- String s= new String(ch, 2, 3); //Now, s contains cde

6. The general forms are:

  - String(byte asciiChars[ ])
  - String(byte asciiChars[ ], int startIndex, int numChars)

- **For example,**
- byte ascii[] = {65, 66, 67, 68, 69, 70 };
- String s1 = new String(ascii); // s1 contains ABCDEF String s2 = new String(ascii, 2, 3); // s2 contains CDE

- JDK 5 and higher versions have two more constructors. The first one supports the extended Unicode character set.

The general form:
String(int codePoints[ ], int startIndex, int numChars)
here, codePoints is array containing Unicode .

8. Another constructor supports **StringBuilder:**

  - String(StringBuilder strBuildObj)

10. Explain the life cycle of a servlet.

Java Servlets are programs that run on a Web or Application server

Act as a middle layer between a request coming from a Web browser or other HTTP client and databases or applications on the HTTP server.
Using Servlets, you can collect input from users through web page forms, present records from a database or another source, and create web pages dynamically.
Servlets are server side components that provide a powerful mechanism for developing web applications.

A servlet life cycle can be defined as the entire process from its creation till the destruction. The following are the paths followed by a servlet
The servlet is initialized by calling the init () method.
The servlet calls service() method to process a client's request.
The servlet is terminated by calling the destroy() method.
Finally, servlet is garbage collected by the garbage collector of the JVM.
Now let us discuss the life cycle methods in details.
The init() method :

The init method is designed to be called only once.
It is called when the servlet is first created, and not called again for each user request. So, it is used for one-time initializations, just as with the init method of applets.
The servlet is normally created when a user first invokes a URL corresponding to the servlet, but you can also specify that the servlet be loaded when the server is first started.
The init() method simply creates or loads some data that will be used throughout the life of the servlet.
The init method definition looks like this:

```
public void init() throws ServletException {
// Initialization code...
}
```

The service() method :

The service() method is the main method to perform the actual task.
The servlet container (i.e. web server) calls the service() method to handle requests coming from the client(browsers) and to write the formatted response back to the client.

Each time the server receives a request for a servlet, the server spawns a new thread and calls service. The service() method checks the HTTP request type (GET, POST, PUT, DELETE, etc.) and calls doGet, doPost, doPut, doDelete, etc. methods as appropriate.
Signature of service method:

```
public void service(ServletRequest request, ServletResponse response)
throws ServletException, IOException
{
}
```

The service () method is called by the container and service method invokes doGe, doPost, doPut, doDelete, etc.methods as appropriate.
So you have nothing to do with service() method but you override either doGet() or doPost() depending on what type of request you receive from the client.
The doGet() and doPost() are most frequently used methods with in each service request. Here is the signature of these two methods.
The doGet() Method
A GET request results from a normal request for a URL or from an HTML form that has no

METHOD specified and it should be handled by doGet() method.

```
public void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
// Servlet code
}
```

The doPost() Method

A POST request results from an HTML form that specifically lists POST as the METHOD and it should be handled by doPost() method.

```
public void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException
{
// Servlet code
}
```

The destroy() method :

The destroy() method is called only once at the end of the life cycle of a servlet.

This method gives your servlet a chance to close database connections, halt background threads, write cookie lists or hit counts to disk, and perform other such cleanup activities.

After the destroy() method is called, the servlet object is marked for garbage collection.

The destroy method definition looks like this:

```
public void destroy() {
// Finalization code...
}
```