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

Internal Assessment Test 3 – July 2022

Solutions

Sub:	Advanced Java and J2EE					Sub Code:	18CS644	Branch:	CSE / ISE	
Date:	01.07.2022	Duration:	90 mins	Max Marks:	50	Sem / Sec:	6 A,B,C		OBE	

Answer any FIVE FULL Questions

		MARKS	CO	RBT

1 (a)	Explain the methods used to handle cookies in servlets and JSP
	Cookie(String *name*, String *value*)
	• **A servlet can write a cookie to a user's machine via the addCookie() method of the HttpServletResponse interface.**
	• **Some information saved for each cookie includes**
	  • **name of the cookie**
	  • **value of the cookie**
	  • **expiration data of the cookie**
	  • **Determines when cookie is deleted from the user's machine.**
	  • **it not assigned it is deleted when current browser session ends.**
	  • **domain and path of the cookie – determine when it is included in the header.**
	  • **If the user enters a URL whose domain and path match these values, the cookie is then supplied to the web server. Otherwise, it is not**
	void setComment(String c) Sets the comment to c.
	void setDomain(String d) Sets the domain to d.
	void setHttpOnly(boolean httpOnly) If httpOnly is **true**, then the HttpOnly attribute is added to the cookie.
	If httpOnly is ***false,*** the HttpOnly attribute is removed.
	void setMaxAge(int secs) Sets the maximum age of the cookie to secs. This is the ***number of seconds after which the cookie is deleted***.
	void setPath(String p) Sets the path to p.
	void setSecure(boolean secure) Sets the security flag to secure.
	void setValue(String v) Sets the value to v.
	void setVersion(int v) Sets the version to v.
	
	String getComment( ) Returns the comment.
	String ( ) Returns the maximum age (in seconds).
	String getName( ) Returns the name.
	String getPath( ) Returns the path.
	boolean getSecure( ) Returns **true** if the cookie is secure. Otherwise, returns ***false***.
	String getValue(getDomain( ) Returns the domain.
	int getMaxAge) Returns the value.
	int getVersion( ) Returns the version
	
	In JSP
	To add a cookie to be stored in client's hard disk, o Initialize cookie, use the constructor
	o pass the created cookie to the **addCookie() method**
		[05]	CO4	L2

	• To retrieve, initialize a cookie object o use request.**getCookies() method, which returns an array of cookies.**			

(b)	Write either a servlet or JSP program to write a cookie with name "Email" and cookie value abc@gmail.com.
	
	Using Servlet
	
	addCookieServlet.java
	import java.io.*;
	import javax.servlet.*;
	import javax.servlet.http.*;
	public class addCookieServlet **extends** HttpServlet {
	private static final long *serialVersionUID* = 1L;
	protected void doPost(HttpServletRequest req, HttpServletResponse res) **throws** ServletException, IOException {
	// get parameter from HTTP request
	String data = req.getParameter("data");
	// create cookie
	Cookie cookie = **new** Cookie("MyCookie", data);
	// Add Cookie to HTTP response
	res.addCookie(cookie);
	//Write output to browser
	res.setContentType("text/html");
	PrintWriter pw = res.getWriter();
	pw.println(" Add Cookie
	");
	pw.println("My cookie has been set to :"+data);
	pw.close();
	}
	}
	
	Using JSP
	<html>
	<head><title> Working with cookies </title>
	</head>
	<body>
	  Add a Cookie
	<%! String MyCookieName = "Email";
	String MyCookieValue= "abc@gmail.com";
	%>
	<%Cookie c = new Cookie(MyCookieName, MyCookieValue);
	response.addCookie(c);
	%>
	  Name: <%=MyCookieName %>
	  Value: <%=MyCookieValue %>
	</body>
	</html>
		[05]	CO4	L3

2 (a)	Compare and Contrast Strings and StringBuffers using their constructors.	[06]	CO3	L2

(b)	User has filled out a form containing USN. USN is the primary key in my database to retrieve student details. When I match the USN entered by user with the USN in the table, I get an error. I printed the USN entered by user and it's length. I get this output: jshell> System.out.println(usn) 1CR20CS001 jshell> System.out.println(usn.length()) 16 What is the problem? What is the solution? The problem is the extra spaces on the left and right of the USN. The solution is to use trim().	[04]	CO3	L3

3 (a)	Explain using syntax and code snippets, the following methods of StringBuffer:
	(i) append (ii) insert (iii) delete
	
	• StringBuffer append(String *str*)
	• StringBuffer append(int *num*)
	• StringBuffer append(Object *obj*)
	• concatenates the string representation of any other type of data to the end of the invoking **StringBuffer** object
	• The result is appended to the current **StringBuffer** object
	StringBuffer str = new StringBuffer("Hello");
	String s = str.append(" Jim").append(" How are you?").toString();
	System.out.println(s);
	
	• inserts one string into another
	• StringBuffer insert(int *index*, String *str*)
	• StringBuffer insert(int *index*, char *ch*)
	• StringBuffer insert(int *index*, Object *obj*)
	• overloaded to accept values of all the primitive types, plus **String**s, **Object**s, and **CharSequence**s
	• *index* specifies the index at which point the string will be inserted into the invoking **StringBuffer**
	StringBuffer str = new StringBuffer("Hello Jim. How are you?");
	str.insert(6, "Cara and ");
	
	System.out.println(str.toString());
	
	• StringBuffer delete(int *startIndex*, int *endIndex*)
	– deletes a sequence of characters from the invoking object
	– *startIndex* specifies the index of the first character to remove
	– *endIndex* specifies an index one past the last character to remove
	– the substring deleted runs from *startIndex* to *endIndex*–1
	• StringBuffer deleteCharAt(int *loc*) - deletes the character at the index specified by *loc*
	StringBuffer str = new StringBuffer("Hello Jim. How are you?");
	
	str.delete(5, 10);
	System.out.println(str.toString());
		[06]	CO3	L2

(b)	Write a program using StringBuffer to replace every occurrence of the word “Java” with “ JSP” in the following string.
	StringBuffer sb =”Programming in Java, Java is easy to learn, Java is simple to use”
	
	Methods you may require:
	• indexOf(String *str*): Searches the invoking **StringBuffer** for the first occurrence of *str*
	• StringBuffer replace(int *startIndex*, int *endIndex*, String *str*)
	
	Program:
	public class Main
	{
	public static void main(String[] args) {
	StringBuffer sb = new StringBuffer("Programming in Java, Java is easy to learn, Java is simple to use");
	String s = "Java";
	String r = "JSP";
	while(true){
	int i = sb.indexOf(s);
	if (i==-1) break;
	sb.replace(i,i+s.length(),r);
	}
	System.out.println(sb);
	}
	}
	Output:
	Programming in JSP, JSP is easy to learn, JSP is simple to use
		[04]	CO3	L3

4 (a)	Explain the lifecyle of servlets and HTTP servlets. Servlets typically run inside multithreaded Servlet containers that can handle multiple requests concurrently. Developers must be aware to synchronize access to any shared resources Resource include files, network connections. They should synchronize the servlet's class and instance variables. init() : invoked only when servlet is first loaded into memory. possible to pass initialization parameters to the servlet. **void init(ServletConfig *sc)* throws ServletException** ❑ Called when the servlet is initialized. ❑ Initialization parameters obtained from ***sc***. ❑ ServletException should be thrown if the servlet cannot be initialized. Service() : called to process HTTPRequest. it may also have to formulate a HTTP response. **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. **HttpServlet** **Provides methods to handle HTTP request and responses**	[5M]	CO4	L2

	called for each HTTPRequest Service() calls appropriate doGet() or doPost() method **HttpServletRequest** **Enables servlets to read from a HTTP request** **HttpServletResponse** **Enables servlets to write data to a HTTP request** **void doGet(HttpServletRequest req, HttpServletResponse res)** **throws IOException, ServletException -** Handles an HTTP GET request void **doPost**(HttpServletRequest *req,* HttpServletResponse *res)* throws IOException, ServletException - Handles an HTTP POST request void **doPut**(HttpServletRequest *req,* HttpServletResponse *res)* throws IOException, ServletException - Handles an HTTP PUT request void **doDelete**(HttpServletRequest *req,* HttpServletResponse *res)* throws IOException, ServletException - Handles an HTTP DELETE request. void **doHead**(HttpServletRequest *req,* HttpServletResponse *res)* throws IOException, ServletException - Handles an HTTP HEAD request. void **doOptions**(HttpServletRequest *req,* HttpServletResponse *res)* throws IOException, ServletException - Handles an HTTP OPTIONS request. void **doTrace**(HttpServletRequest *req,* HttpServletResponse *res)* throws IOException, ServletException - Handles an HTTP TRACE request. **void service(HttpServletRequest req,** **HttpServletResponse res)** throws IOException, ServletException Called by the server when an HTTP request arrives for this servlet. The arguments provide access to the HTTP request and response, respectively. Destroy():returns any resources allocated to servlets memory allocated for servlet and objects are garbage collected.			
(b)	Write an HTTP servlet to obtain Name of user and their color preference. Display a customized greeting for the user within a of dimensions 300px by 200px with the preferred background color. <!DOCTYPE html> <html> <head> <meta charset="ISO-8859-1"> <title>color</title> </head> <body> Background color - Servlet <form name=colorForm method=get action=servlets/colorServlet> <label> Name </label> <input type=text name=uname /> <label> Choose a color	[5M]	CO4	L3

```
                </label>
                <select name =color>
                        <option value=red>Red</option>
                        <option value=green>Green</option>
                        <option value=blue>Blue</option>
                </select>
                <p>
                <input type=submit value=SUBMIT>
                </p>
                </form>
        </div>

</body>
</html>

import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class colorServlet extends HttpServlet{


        protected void doGet(HttpServletRequest req, HttpServletResponse res) throws
ServletException, IOException {
                        String color = req.getParameter("color");
                        PrintWriter pw = res.getWriter();
                        String uname = req.getParameter("uname");

                        pw.println("<body>");
                        pw.println("<div                                style='background-
color:"+color+";width:300px; height:200px'>Have a nice day "+uname+"</div>");

                        pw.println("</body>");
                        pw.close();
        }
}

Deployment descriptor:
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE web-app
PUBLIC "-//Sun Microsystems, Inc.//DTD Web
Application 2.3//EN"  "http://java.sun.com/dtd/web-app_2_3.dtd">

<web-app>
<servlet>
  <servlet-name>colorServlet</servlet-name>
  <servlet-class>colorServlet</servlet-class>
</servlet>
<servlet-mapping>
  <servlet-name>colorServlet</servlet-name>
  <url-pattern>/servlets/colorServlet</url-pattern>
```

	</servlet-mapping> </web-app> Output:  			
5 (a)	Write short notes on various tags in JSP. **A JSP program consists of a combination of HTML tags and JSP tags.** **• JSP tags define Java code that is to be executed before the output of the JSP program is sent to the browser** **• JSP tags begin with <% followed by java code %>** **• XML version : <jsp:TagID></JSP:TagID>** **• JSP tags are embedded into HTML component of a JSP program** **• It is processed by a JSP virtual engine such as Tomcat** **• HTML tags are sent to browser.** **• JSP tags are executed by Tomcat, the result is sent to the process.** **Directive Tags** **<%@ %>** **commands the JSP virtual engine to perform a specific task such as importing a Java package.** **<%@ page import = "java.sql.*" %>** **%@include file="jim\books.html" %>** **<%taglib uri="mytags.tld" %>** **3 commonly used directives :** **import : used to import Java packages** **include : inserts a specified file into the JSP program** **taglib : specifies a file that contains a tag library.** **Declaration Statement Tags** defines variables, objects, methods available to other components of the JSP program **Expression Tags** <%= %>	[05]	CO4	L2

used for an expression statement whose result replaces the expression tag when the JSP virtual engine resolves JSP tags

Comment Tag

<%-- comment --%>

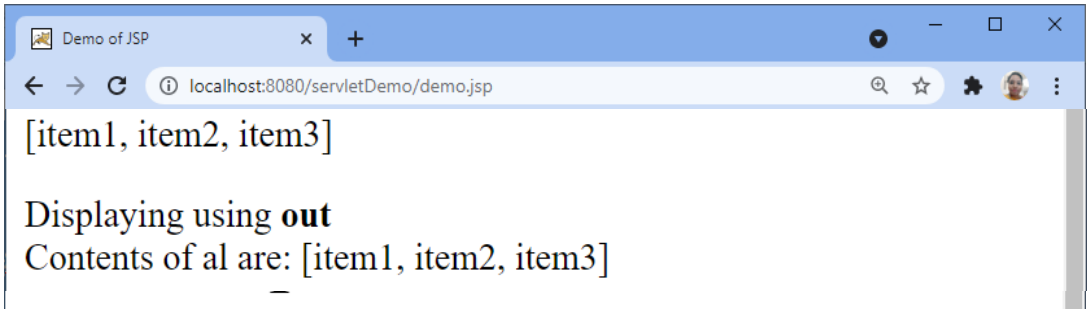
commands the JSP virtual engine to perform a specific task such as importing a Java package.

Scriptlet Tags

<% %>

contains commonly used Java control statements and loops

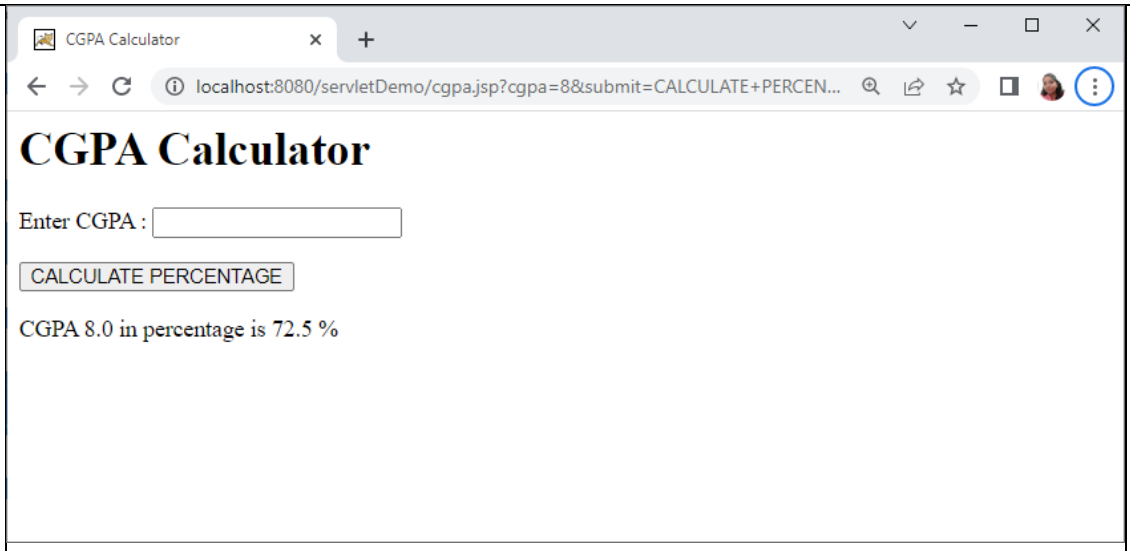
```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>

<%@ page import="java.util.*" %>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Demo of JSP</title>
</head>
<body>
<%--Declarations in  JSP --%>
<%! ArrayList<String> al = new ArrayList<String>();
    %>
<%
Collections.addAll(al, "item1","item2","item3");
%>
<p><%=al %></p>
<p>Displaying using <strong>out</strong><br>
   <%  out.println("Contents of al are: "+al); %></p>

</body>
</html>
```



(b)	Write JSP code to obtain CGPA out of 10 from the user using a form. Calculate the percentage and display. Use this formula: [CGPA - 0.75] * 10 Ensure that you convert the floating point value obtained as a string to float. float f=Float.parseFloat("23.6");	[05]	CO4	L3

cgpa.jsp

```
<html>
<head>
    <title>CGPA Calculator</title>
</head>

<body>
<h1> CGPA Calculator</h1>
    <form method=get action=cgpa.jsp>
    <p>
            <label>Enter CGPA :</label>
            <input type=number step=any name=cgpa />
    </p>
            <p>
            <input type=submit name=submit value="CALCULATE
PERCENTAGE" /></p>

    </form>
    <% if (request.getParameter("submit")!=null) {
       double cgpa=Double.parseDouble(request.getParameter("cgpa"));
            double per = (cgpa-0.75) *10;

    %>
    <p>  CGPA <%=cgpa %> in percentage is <%=per %> %</p>
    <%
    }
    %>


</body>

</html>
```



6	Explain various steps involved in JDBC process with code snippet.
	
	1. **Loading the JDBC driver**
	2. **Connecting to the DBS**
	3. **Creating and executing a statement**
	4. **Processing data returned by the DBMS**
	5. **Terminating the *connection with the DBMS***
	Loading the JDBC driver
	Class.forName() is used to load Java EE component
	• For connecting to Microsoft Access, write a routing that loads the JDBC/ODBC Bridge driver called sun.jdbc.odbc. JdbcOdbcDriver.
	Class.forName("sun.jdbc.odbc.JdbcOdbcDriver)
	If you had used a MySQL driver
	try{
	Class.*forName("com.mysql.cj.jdbc.Driver");*
	}**catch(ClassNotFoundException err) {**
	System.***out.println("Unable to connect:"+err);***
	System.*exit(1);*
	}
	Connecting to the database system.
	• Connect using **DriverManager.getConnection()** method.
	• **java.sql.DriverManager** class is the higher class in the java.sql hierarchy and is responsible for managing driver information.
	• Parameters passed o URL:
	- **a string object** that contains the driver name and name of database being accessed.
	- **Username (optional)**
	- **Password (optional)**
	• Returns a **Connection interface** that is used to reference the database.
	
	Creating and Executing a statement
	After the connection is made, **SQL query** can be sent to DBMS for processing
	• **SQL query** – consists of a series of SQL command that direct DBMS.
	• The **createStatement ()** method of the Connection interface is used to create a Statement object.
	• The **Statement object** is then used to execute a query that returns a **ResultSet** object that contains a response from the DBMS.
		[10]	CO5	L2

Processing data returned by the DBMS

The **java.sql.ResultSet** object is assigned the results received from the DBMS after the query is processed.

• **next()** method of the ResultSet class

- The first time next() method is called, positions ResultSet pointer at the first row of the ResultSet.
- Returns a **Boolean** value that if false indicates that no rows are present in the ResultSet.
- A **true value** means **at least one row** is present in ResultSet.

• **do…while()** loop is used to loop through each record

• **getString()** method – used to copy the column contents in the ResultSet

- can pass the column name or the column number.

Terminating the connection with the DBMS

Use the **close()** method of the Connection object to terminate the connection to the DBMS.

• Throws an exception if a problem is encountered.

ResultSet closes automatically once Connection, but it is better to close it explicitly.

```
mysql> create database custDB;
Query OK, 1 row affected (0.57 sec)
mysql> use custDB;
Database changed
mysql> create table cust_tbl(ID int(11) primary key auto_increment, custNumber varchar(10) unique, fname varchar(20), lname varchar(20), street varchar(20));
mysql> insert into cust_tbl (custNumber,fname, lname, street) values('A100','Bobby','Holland', '#14, AG Avenue');
Query OK, 1 row affected (0.24 sec)
mysql> insert into cust_tbl (custNumber,fname, lname, street) values('B100','Max','Owens', '#16 Maple Street');
Query OK, 1 row affected (0.29 sec)
mysql> insert into cust_tbl (custNumber,fname, lname, street) values('C100','Jen','Watson', '#18 Oak Street');
Query OK, 1 row affected (0.13 sec)
```

dbConn.java

```
package demo;
import java.sql.*;
import javax.sql.*;
public class dbConn {
        public Connection conn;
        public dbConn() {
                        try {
                                Class.forName("com.mysql.cj.jdbc.Driver");
                                 System.out.println("Driver loaded Successfully");
                        }catch(ClassNotFoundException ce) {
                                System.out.println("Unable to Connect");
                                ce.printStackTrace();
                        }

try {
```

```
        String url="jdbc:mysql://localhost:3306/custDB";
        String uname = "root";
        String pswd ="admin";// if password is not set, "";
        Connection conn = DriverManager.getConnection(url,username, password);
                System.out.println("Connected to database successfully!");
    }catch(SQLException err) {
        System.out.println("Unable to connect to database");
                err.printStackTrace();
                                }

        }
}
```

dbDemo.java

```
package demo;
import java.sql.*;
import javax.sql.*;
public class dbDemo {
        public static void main(String[] args) {
                dbConn dc = new dbConn();
                Connection conn = dc.conn;
                Statement stmt;
                ResultSet rs;
                try {
                        String query = "SELECT * from cust_tbl";
                        stmt=conn.createStatement();
                        rs = stmt.executeQuery(query);
boolean rec = rs.next();
                        if(!rec) {
                                System.out.println("No data returned");
                        }else {
                                System.out.println(String.format("%11s                %20s
%20s","cust_num","Name","Street"));
                                do {
                                        String cust_num = rs.getString("custNumber");
                                        String fname = rs.getString("fname");
                                        String lname = rs.getString("lname");
                                        String street = rs.getString("street");
                                System.out.println(String.format("%11s                %20s
%20s",cust_num,fname+" "+lname,street));
                                        }while(rs.next());
                        }
                        stmt.close();
                        rs.close();


                }catch(SQLException err) {
                        err.printStackTrace();
                }
        }
}
```

	Output: cust_num Name Street A100 Bobby Holland #14, AG Avenue B100 Max Owens #16 Maple Street C100 Jen Watson #18 Oak Street			

CI CCI HoD

CO PO Mapping

CO-PO and CO-PSO Mapping																			
Course Outcomes		**Blooms Level**	**Modules covered**	**PO1**	**PO2**	**PO3**	**PO4**	**PO5**	**PO6**	**PO7**	**PO8**	**PO9**	**PO10**	**PO11**	**PO12**	**PSO1**	**PSO2**	**PSO3**	**PSO4**
CO1	Interpret the need for advanced JAVA concepts like Enumerations, Wrapper Classes and Annotation.	**L1,L 2,L3**	**1**	2	2	2	0	3	0	0	0	3	0	0	0	3	0	0	2
CO2	Explain Collections Interface and Framework in development of modular applications.	**L1,L 2,L3**	**2**	2	2	2	0	3	0	0	0	3	0	0	0	3	0	0	2
CO3	Use In-Built String Handling Functions for development of Java Programs.	**L1,L 2,L3**	**3**	2	2	2	0	3	0	0	0	3	0	0	0	3	0	0	2
CO4	Describe how servlet fit into Java – Based web application architecture	**L1,L 2,L3**	**4**	3	2	2	0	3	0	0	0	3	0	0	0	3	0	0	3
CO5	Illustrate database access and details for managing information using the JDBC API	**L1,L 2,L3**	**5**	3	3	2	0	3	0	0	0	3	0	0	0	3	0	0	3

COGNITIVE LEVEL	REVISED BLOOMS TAXONOMY KEYWORDS
L1	List, define, tell, describe, identify, show, label, collect, examine, tabulate, quote, name, who, when, where, etc.
L2	summarize, describe, interpret, contrast, predict, associate, distinguish, estimate, differentiate, discuss, extend
L3	Apply, demonstrate, calculate, complete, illustrate, show, solve, examine, modify, relate, change, classify, experiment, discover.
L4	Analyze, separate, order, explain, connect, classify, arrange, divide, compare, select, explain, infer.
L5	Assess, decide, rank, grade, test, measure, recommend, convince, select, judge, explain, discriminate, support, conclude, compare, summarize.

PROGRAM OUTCOMES (PO), PROGRAM SPECIFIC OUTCOMES (PSO)				CORRELATION LEVELS	
PO1	Engineering knowledge	PO7	Environment and sustainability	0	No Correlation
PO2	Problem analysis	PO8	Ethics	1	Slight/Low

PO3	Design/development of solutions	PO9	Individual and team work	2	Moderate/ Medium
PO4	Conduct investigations of complex problems	PO10	Communication	3	Substantial/ High
PO5	Modern tool usage	PO11	Project management and finance		
PO6	The Engineer and society	PO12	Life-long learning		
PSO1	Develop applications using different stacks of web and programming technologies				
PSO2	Design and develop secure, parallel, distributed, networked, and digital systems				
PSO3	Apply software engineering methods to design, develop, test and manage software systems.				
PSO4	Develop intelligent applications for business and industry				