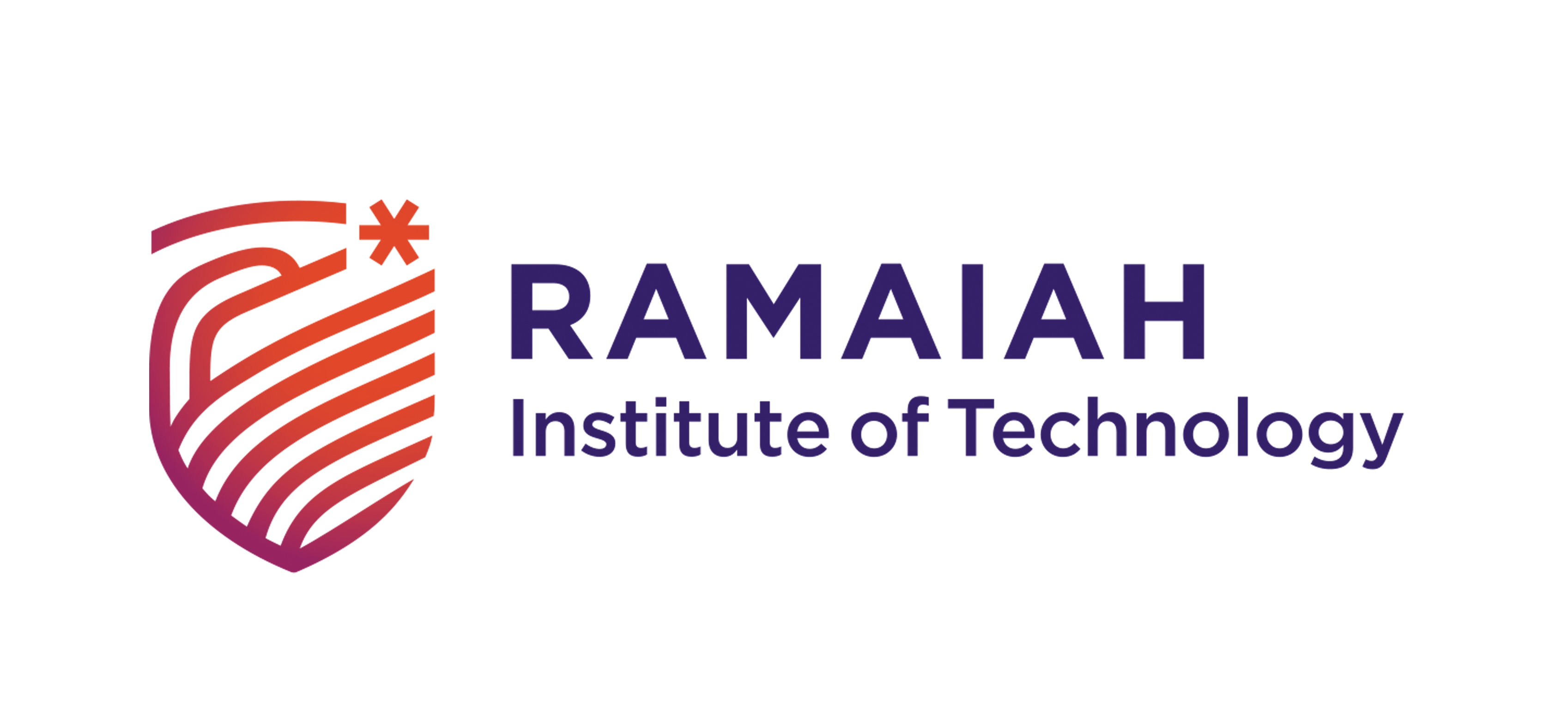
INFORMATION SCIENCE AND ENGINEERING



**Laboratory MANUAL**

**Semester: 6th**

|  |
| --- |
| **JAVA AND J2EE – ISL67** |

Prepared by: Dr Sumana M

# Academic Year : 2021-22

**TABLE OF CONTENTS**

***Sl.No. Contents PageNo.***

|  |  |  |
| --- | --- | --- |
| ***1.*** | Vision, Mission, PEO, PSO, PO | **1** |
| ***2.*** | Introduction and scope of the course | **4** |
| ***3.*** | Syllabus | **5** |
| ***4.*** | Rubrics for Evaluation | **9** |
| ***5.*** | Problem Statements |  |
| ***6.*** | References |  |

**Vision and Mission of the Department**

**Vision**

To evolve as an outstanding education and research centre of information technology to create high quality engineering professionals for the betterment of society

**Mission**

**Department of Information Science and Engineering shall create high quality IT Engineering Professionals for the betterment of society by**

* Providing education through an ever-improving curriculum and effective pedagogy techniques.
* Department of Information Science and Engineering shall create high quality IT Engineering Professionals for the betterment of society by encouraging extra and cocurricular activities to develop their overall personality along with technical skills.
* Department of Information Science and Engineering shall create high quality IT Engineering Professionals for the betterment of society by collaborating with industry and academia for strengthening research, innovation and entrepreneurship ecosystem.

**Program Educational Objectives (PEOs)**

|  |  |
| --- | --- |
| **PEO No.** | **Program Educational Objectives Statements** |
| PEO1 | Become competent information technology professionals with continuous progress in career or learning |
| PEO2 | Productively engage with society by practicing research or entrepreneurship. |
| PEO3 | Function effectively as professionals in a team environment or individually |

**Program Outcomes**

|  |  |
| --- | --- |
| PO- 1 | Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems. |
| PO- 2 | Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences |
| PO- 3 | Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations. |
| PO- 4 | Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions. |
| PO- 5 | Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations |
| PO- 6 | : The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice. |
| PO- 7 | Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development. |
| PO- 8 | Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice. |
| PO- 9 | Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings. |
| PO- 10 | Communication: Communicate effectively on complex engineering 7 activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions. |
| PO- 11 | Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one’s own work, as a member and leader in a team, to manage projects and in multidisciplinary environments. |
| PO- 12 | Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change. |

**Program Specific Objectives (PSOs)**

|  |  |
| --- | --- |
| **PSO1** | Apply Mathematical models, programming paradigms and software development practices to solve real world problems |
| **PSO2** | Adopt computing and communication models for developing IT solutions. |
| **PSO3** | Acquire data engineering skills to develop intelligent systems in a multidisciplinary environment. |

# Introduction and scope of the course

# Java is popular among programmers because about its wide adaptability features. This programming language follows the “Write once , execute anywhere” policy. It runs in all device that have a Java virtual machine rebuilt. The scope of this course is to assist students to develop real time applications that includes creating front ends, storing and retrieving from databases and performing fast computations. This course will assist students to build desktop or web applications in an efficient manner.

# J2EE also called Java Platform, Enterprise Edition (Java EE) includes several technologies that extend the functionality of the Java SE APIs, such as Servlets, Connectors, Enterprise JavaBeans, etc

# The J2EE architecture provides services to simplify the common challenges faced by developers while developing modern applications, thereby making it easier to implement industry-standard design patterns for greater efficiency and reliability. It is comprised of a significant set of APIs along with Java EE core technologies that help make the most of enterprise development. J2EE is just a part of Java with a powerful set of libraries.

**COURSE DESIGN, DELIVERY AND ASSESMENT**

|  |  |
| --- | --- |
| Course code and Title : ISL67, Java and J2EE lab | Course Credits : 0:0:1 |
| CIE : **50 Marks** | SEE : **50 Marks** |
| Total No of Theory / Tutorial / Lab Hours : 14 | |

**Prerequisites**

|  |
| --- |
| Object Oriented Programming using Java |

**Course Objectives**

1. Design java applications and be able to implement them.
2. Understand the importance of java concepts and their use in real world applications.
3. Ability to choose appropriate java concepts for problem solving.

**Syllabus**

# 1) Write a Program that simulates a telephone that records missed incoming calls. For each missed call, store the time of call, telephone number of origin, and name of the caller if the name is available. For unlisted numbers, set the name to “private caller”. Choose or extend the most appropriate collection class and provide the following features.

# a. Numbers are recalled in the order they arrive

# b. Up to 10 numbers are recorded. When the eleventh call comes in, it is stored and the oldest call is deleted so that no more than 10 numbers are ever recorded.

# c. After each number display, the user can select

# i. To delete the call

# ii. To go on to the next missed call, or

# iii. To display the call details (number, caller name and time). Delete the number if user specifies a number to delete.

# Write a helper class to represent an incoming call with fields to hold the number, name of the caller, and time of the call. Write a tester call that stores the several numbers, simulate the user pressing the missed-calls button, and finally prints the entire collection of stored calls.

# 2) Write a Java program using user-defined storage classes to create a book database and store it in a Collection List.

# a. Books collection should include title, author, publisher and price.

# b. Write a method to sort the books in ascending order of price and store it in another List. Maintain the book details with respect to an unique book id.

# c. Prompt for an author name and list all the books with the same author name. Create a new list holding all the book details with price greater than a user specified price.

# d. For a given a value by the user, find all the books that match either the whole or a part of the book title.

# e. Identify a publisher and print books from a particular publisher. Update the publisher details based on a title.

# 3) Create a desktop java application using swings to enable an user to enter student information such as name, usn, age, address, sgpa of 8 semesters, category.

# a. Perform validations on all the fields. Display appropriate messages in pop up boxes to indicate wrong entries.

# b. On clicking of the “compute” button, find the cgpa . On clicking of the “done” button, mouse place the student details in a collection.

# c. Display the collection in a textarea on the click of a button.

# 4) Write a java program using Swings to validate user login information using dialog boxes.

# a. Once validated, allow the user to enter the customer id, if the person is a new customer, else check whether the customer exists in a collection and obtain the customer id.

# b. The customer id can be obtained given a mobile number. Allow the user to enter the item purchased by giving the item id and quantity purchased.

# c. On clicking of a button, the item name and the total cost should appear in the corresponding GUI components.

# d. Using option dialog box, indicate the types of discount available for the customer. On clicking on the print button, print the details in information dialog box.

# 5) Write a program that uses Java Swing and JDBC to create a stand-alone application:

# a. Create two tables namely, Representative (RepNo, RepName, State, Comission, Rate) and Customer (CustNo, CustName, State, Credit\_Limit, RepNo) in MySQL database. Use appropriate Swing components to insert values in a form.

# Use another form to display Representative’s information when Credit\_Limit is above 15,000.

# 6) Create a Servlet to file IT returns that accepts personal information, salary information and Tax deduction details from the user and write the information into a file. Also accept the name of the person and display in on the page.

# 7) Create a servlet that accepts patient information in a hospital such as patient id, patient name, and age, date of admission, cause of admission, doctor diagnosed, and treatment proposed. Place the details into a database. Allow options to insert, update, and view and delete the contents in the database.

# 8)Write a JSP and Servlet Program to do the following to buy a T-Shirt online:

# a. A set of checkboxes to select your T-Shirt accessories such as ‘belt’, ‘cap’, ‘hair-band’ etc.

# b. A text area / text field to enter your T-Shirt tag-line, A Radio-button that allows the user to choose between T-Shirt with chest pocket and without. A Combo Box to choose your T-Shirt color, A Button called “Click Me”

# c. Insert the details entered into a table called ‘TShirts’.

# d. An OrderNo is generated by adding ‘1’ to the existing ‘OrderNo’

# e. If ‘TShirts’ table is empty the initial value of ‘OrderNo’ is 100.

# f. This ‘OrderNo’ is also inserted into the ‘TShirts’ table

# g. Display all the records of the ‘TShirts’ table in tabular form

# PS: Frontend display should be in JSP and the business logic should be written in Servlet Class.

# 9) Write a Java Program that creates two threads object of Thread class. Where one thread asks the user to enter a number not less than four digits. Split the digits of the number and display in words the value of the number. Ex: 1 – One. Second thread finding the number of vowels in a word. Ex: JAVA – Vowel - A, Count – 2.

# 10) Create a Customer Application with Spring framework. The Customer class should include attributes: name, id, Items purchased . Class Item has the following attributes: name, id, price, tax, manufacture date. Create objects and calculate the total price of the items purchased by the customer.

# 11)Create the following application with Springs and Hibernate framework

# a. Create a Telephone Directory Application that searches the database based on phone number or name. Also show database table creation with inserting 2-3 values to the table.

# b. Database Name: OnlineDirectory, Table Design: Table Name: Telephone\_Directory, Attributes: Phone\_Number, Name, Address, Company, Pin\_Code.

# Lab 12 and 13 – Miniproject

# Java and J2EE Tutorial

# 1. java.util Part 1: The Collections Framework: Collections Overview, The Collection Interfaces

# 2. The Collection Classes: The ArrayList Class, LinkedList Class, HashSet Class.

# 3. Accessing a Collection via Iterator, Storing User-Defined Classes in Collections;

# 4. Working with Maps, Arrays

# 5. Introduction to Swing, Swing Components,

# 6. Swing basic containers, Swing Components;

# 7. Working with Databases: Types of JDBC Drivers, How to access database, Using Databases, Connecting to Database.

# 8. Introduction to Servlet: Architecture of Servlet.

# 9. Basics of Servlet application programming Interface: Servlet Application Programming Interface.

# 10. The Servlet Architecture, the Servlet Life Cycle.

# 11. Multithreading

# 12. Springs Framework

# 13. Hibernate Framework

# 14. Connecting Springs and Hibernate

# Text Books

# 1. Herbert Schildt, ‘The Complete Reference Java (J2SE 5 Edition)’, TATA McGRAW-HILL Edition 2005.

# 2. Ivan Bayross, Sharanam Shah, Cyntiha Bayross and Vishali Shah, ‘Java EE 5 for Beginners’, SPD (Sharoff Publishers & Distributors Pvt. Ltd.), 2nd edition August 2008.

# Reference:

# 1. Jim Keogh,The Complete Reference J2EE‟, TATA McGRAW-HILL Edition 2002.

# 2. B V Kumar, S Sangeetha, S V Subrahmanya, J2EE Architecture, TATA McGRAW-HILL Edition 2007.

**Course Outcomes**

1. Develop solutions for the given problem using Java and J2EE (PO-1(1), 2(2), 3(3), 5(2), 6(2)) (PSO-1(2)
2. Apply java and J2EE concepts to provide solutions in various domains. (PO-1(1), 2(2), 3(3), 4(2),5(2), 6(2),9(3), 10(2), 11(3),12(1)) (PSO-1(2))
3. Interpret the results and produce the substantial document. (PO- 2(2), 4(2), 10(3)) (PSO-1(2))

**Mapping Course Outcomes with Program Outcomes:**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Course Outcomes | Program Outcomes | | | | | | | | | | | | PSO’s | | |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 1 | 2 | 3 | |
| 1 | 1 | 2 | 3 |  | 2 | 2 | - | - | - | - | - | - |  | 2 | - | |
| 2 | 1 | 2 | 3 | 2 | 2 | 2 | - | - | 3 | 2 | 3 | 1 |  | 2 | - | |
| 3 | - | 2 |  | 2 | - | - | - | - |  | 3 |  | - |  | 2 |  | |

# Rubrics for Continuous Evaluation and Record Work

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Performance indicators** | **Excellent(E)** | **Good(G)** | **Fair(F)** | **Poor(P)** | **PO’s** |
| Fundamental Knowledge  Problem Definition  (4 Marks) | Student is able to demonstrate thorough knowledge about the fundamental concepts of the course  Student is able to completely understand the problem definition  (4) | Student is able to demonstrate partial knowledge about the fundamental concepts of the course  Student is able to partially understand the problem definition  (3) | Student is able to demonstrate fair knowledge about the fundamental concepts of the course  Student is able to fairly understand the problem definition (2) | Student is not able to demonstrate knowledge about the fundamental concepts of the course  Student is not able to understand the problem definition  (1) | PO1, PO2, PO3, PO9, PO10,  PO12 |
| Design and Analysis (2 Marks) | Students exhibits excellent ability to design and analyse the given problem  2 | Students exhibits goo ability to design and analyze the given problem | Students exhibits fair ability to design and analyse the given problem  1 | Students exhibits no ability to design and analyze the given problem | PO2, PO3, PO9, PO10, PO12 |
| Implementation  (4 Marks) | Student is able to implement all the programs in the given language (4) | Student is able to implement 80% of the programs in the given language (3) | Student is able to implement 50% of the programs in the given language (2) | Student is not able to implement the programs in the given language (1) | PO5, PO9, PO10 PO12 |
| Interpretation of Results  (5 Marks) | Results obtained are clear  (5 marks) | Partial Results are obtained (4 marks) | Major part of results not obtained (3 marks) | Wrong results obtained or output not found (2) | PO9, PO10,PO12 |
| Documentation  (5 Marks) | The report was structured in an orderly manner, and the formatting (e.g: font, spacing, labelling of figures and tables, equations numbered and etc) was done properly. (5) | The report was structured and formatted (e.g: font, spacing, labelling of figures and tables, equations numbered and etc) in a satisfactory manner. (4) | The report was poorly structured, and the formatting (e.g: font, labelling of figures and tables, equations numbered etc) (3) | There was no structure of report and the formatting (2) | PO9, P10 |

# Rubrics for Miniproject

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Performance indicators** | **Excellent(E)** | **Good(G)** | **Fair(F)** | **Poor(P)** | **PO’s** |
| Creativity and Originality in problem statement (2 marks) | Project idea is very creative and original. (2) | Creativity and originality in project idea. (1) | Idea of project is somewhat creative and original. | Lack of Creativity and originality in project idea | PO1,  PO2 |
| Domain Knowledge of problem statement  (2 marks) | Excellent Domain Knowledge (2) | Good Domain Knowledge (1) | Poor Domain Knowledge | Lack of Domain Knowledge | PO1,  PO2 |
| Proper code development and implementation of all modules in project  Successfully completed Changes suggested in code Development (4 marks) | Code is executed properly and validations are used everywhere in project and exception handling is used properly.  Changes suggested in code is working properly and validations are used everywhere in project and exception handling is used properly.  (4) | Code is executed properly and validations are used at some places only and exception handling is used properly.  Changes suggested in code are working properly and validations are used at some places only and exception handling is used properly. (3) | Code is executed properly but validations are not used but exception handling is used properly.  Changes suggested in code are partially working. | Code is not executed properly and validations are not used and exception handling is not used properly.  Changes suggested in code are not working properly.  (2) | P03,  PO4, PO5, PO9,  PO10 |
| Report Writing  (2 marks) | Clear report with all the phases of the project explained and applied suitable reasoning (2) | Report with all the phases of the project explained and applied moderate reasoning (1) | Report with all the phases of the project explained and applied less reasoning | No Clear report | PO6, PO9, PO10 |

| **Lesson No/Session No** | **Topics** | **CO’s** |
| --- | --- | --- |
|  | | |
|  | Write a Program that simulates a telephone that records missed incoming calls. For each missed call, store the time of call, telephone number of origin, and name of the caller if the name is available. For unlisted numbers, set the name to “private caller”. Choose or extend the most appropriate collection class and provide the following features.  Numbers are recalled in the order they arrive  Up to 10 numbers are recorded. When the eleventh call comes in, it is stored and the oldest call is deleted so that no more than 10 numbers are ever recorded.  After each number display, the user can select   1. To delete the call 2. To go on to the next missed call, or 3. To display the call details (number, caller name and time).   Delete the number if user specifies a number to delete.  Write a helper class to represent an incoming call with fields to hold the number, name of the caller, and time of the call. Write a tester call that stores the several numbers, simulate the user pressing the missed-calls button, and finally prints the entire collection of stored calls. | CO1 |
|  | Write a Java program using user-defined storage classes to create a book database and store it in a Collection List. Books collection should include title, author, publisher and price. Also write a method to sort the books in ascending order of price and store it in another List. Maintain the book details with respect to an unique book id. Prompt for an author name and list all the books with the same author name. Create a new list holding all the book details with price greater than a user specified price. For a given a value by the user, find all the books that match either the whole or a part of the book title. Identify a publisher and print books from a particular publisher. Update the publisher details based on a title. | CO1 |
|  | Create a desktop java application using swings to enable an user to enter student information such as name, usn, age, address, sgpa of 8 semesters, category. Perform validations on age and sgpa. Display appropriate messages in pop up boxes to indicate wrong entries, on clicking of the “compute” button. Also find the cgpa based on the obtained sgpa. On clicking of the “done” button, place the student details in a collection. A click on the “display” button should display the collection in a textarea. | CO1 |
|  | Write a java program using swings to validate user login information using dialog boxes. Once validated, allow the user to enter the customer id, if the person is a new customer, else check whether the customer exists in a collection and obtain the customer id. The customer id can be obtained given a mobile number. Allow the user to enter the item purchased by giving the item id and quantity purchased. On clicking of a button, the item name and the total cost should appear in the corresponding GUI components. Using option dialog box, indicate the types of discount available for the customer. On clicking on the print button, print the details in information dialog box. |  |
|  | Write a program that uses Java Swing and JDBC to create a stand-alone application:  Create two tables namely, Representative (RepNo, RepName, State, Comission, Rate) and Customer (CustNo, CustName, State, Credit\_Limit, RepNo) in MySQL database.  Use appropriate Swing components to insert values in a form.  Use another form to display Representative’s information when Credit\_Limit is above 15,000. | CO1 |
|  | Create a Servlet to file IT returns that accepts personal information, salary information and Tax deduction details from the user and write the information into a file. Also accept the name of the person and display in on the page. | CO1 |
|  | Create a servlet that accepts patient information in a hospital such as patient id, patient name, and age, date of admission, cause of admission, doctor diagnosed, and treatment proposed. Place the details into a database. Allow options to insert, update, and view and delete the contents in the database. | CO1 |
|  | Write a JSP and Servlet Program to do the following to buy a T-Shirt online:   1. A set of checkboxes to select your T-Shirt accessories such as ‘belt’, ‘cap’, ‘hair-band’ etc. 2. A text area / text field to enter your T-Shirt tag-line 3. A Radio-button that allows the user to choose between T-Shirt with chest pocket and without. 4. A Combo Box to choose your T-Shirt color 5. Appropriate labels for these GUI Components 6. A Button called “Click Me” which when pressed will 7. Insert the details entered into a table called ‘TShirts’. 8. An OrderNo is generated by adding ‘1’ to the existing ‘OrderNo’ 9. If ‘TShirts’ table is empty the initial value of ‘OrderNo’ is 100. 10. This ‘OrderNo’ is also inserted into the ‘TShirts’ table 11. Display all the records of the ‘TShirts’ table in tabular form   PS: Frontend display should be in JSP and the business logic should be written in Servlet Class. | CO1 |
|  | Write a Java Program that creates two threads object of Thread class. Where one thread asks the user to enter a number not less than four digits. Split the digits of the number and display in words the value of the number. Ex: 1 – One. Second thread finding the number of vowels in a word. Ex: JAVA – Vowel - A, Count – 2. |  |
|  | Create a Customer Application with Spring framework. The Customer class should include attributes: name, id, Items purchased . Class Item has the following attributes: name, id, price, tax, manufacture date. Create objects and calculate the total price of the items purchased by the customer. | CO1 |
|  | Create the following application with Springs and Hibernate frameworka. Create a Telephone Directory Application that searches the database based on phone number or name. Also show database table creation with inserting 2-3 values to the table.b. Database Name: OnlineDirectory, Table Design: Table Name: Telephone\_Directory, Attributes: Phone\_Number, Name, Address, Company, Pin\_Code. | CO1 |
| 1. **And 13.** | Mini Project | CO2 |

**1. Write a Program that simulates a telephone that records missed incoming calls. For each missed call, store the time of call, telephone number of origin, and name of the caller if the name is available. For unlisted numbers, set the name to “private caller”. Choose or extend the most appropriate collection class and provide the following features.**

**Numbers are recalled in the order they arrive**

**Up to 10 numbers are recorded. When the eleventh call comes in, it is stored and the oldest call is deleted so that no more than 10 numbers are ever recorded.**

**After each number display, the user can select**

1. **To delete the call**
2. **To go on to the next missed call, or**
3. **To display the call details (number, caller name and time).**

**Delete the number if user specifies a number to delete.**

**Write a helper class to represent an incoming call with fields to hold the number, name of the caller, and time of the call. Write a tester call that stores the several numbers, simulate the user pressing the missed-calls button, and finally prints the entire collection of stored calls.**

# Steps Involved:

# 1. Create a directory of calls(data stored is phone number : double and Name : String) . This can be created using Arraylist/ LinkedList

# class CallerD

# {

# Date date = new Date();

# String name;

# Integer telNo;

# CallerD(Date d, String n,Integer t)

# {

# date =d;

# name = n;

# telNo = t;

# }

# }

HashMap<Long,String> hm = **new** HashMap<Long,String>();

2. For missed call - Prompt the user to enter telephone number .

ArrayList<Long> callList = **new** ArrayList<Long>();

ArrayList<CallerD> missedCallList = **new** ArrayList<CallerD>();

# Check if the telephone number exists in directory (arraylist/linkedlist)

# If yes, retrieve name and store in the missed call database (phNo:Double, name :String). If no, store the missed call as (phNo, “Private Caller”)

# 3. Add 10 enteries into the missed call list (use linkedlist). For the eleventh entry, call addFirst() to add from 1st position.

# 4. Scan through the missed call list and prompt the user using switch case to

1. To delete the call- add the index of the element to be deleted in a new list , say removeList(index i).

After scanning the entire missed call list, use missedCallList.removeAll(removeList)

1. To go on to the next missed call- Just traverse through the list, or
2. To display the call details (number, caller name and time) – Use System.out.println to print the missed call entry.

5.Prompt the user to mention the missed call to be removed. Use missedCallList.remove() to remove the missed call.

6.Write display() method or write toString() method in each class to display each entry of the missed call as well as directory enteries.

7. Write a main program to test the above classes.

**2. Write a Java program using user-defined storage classes to create a book database and store it in a Collection List. Books collection should include title, author, publisher and price. Also write a method to sort the books in ascending order of price and store it in another List. Maintain the book details with respect to an unique book id. Prompt for an author name and list all the books with the same author name. Create a new list holding all the book details with price greater than a user specified price. For a given a value by the user, find all the books that match either the whole or a part of the book title. Identify a publisher and print books from a particular publisher. Update the publisher details based on a title.**

**Steps Involved:**

1. Create a class Book

class Book

{

String title,author, publisher;

double price;

Book(String title,String author,String publisher,double price)

{ this.title=title;

this.author=author;

this.publisher=publisher;

this.price = price;

}

public String toString()

{

String str= "Book has "+title+"title "+"whose author is "+author;

str+="\n Book is published by "+publisher;

str+="\n Price is= "+price;

return str;

}

}

2. Override the compare method of the Comparator interface to compare based on price

class BookCompare implements Comparator<Book>

{

public int compare(Book b1,Book b2)

{

return (int) (b1.price-b2.price);

}

}

# 3. Maintain the book details with respect to an unique book id – Create hashmap for maintaining book details.

4. Prompt for an author name and list all the books with the same author name.

Call containsValue(author name) of the hashmap

5. Create a new list holding all the book details with price greater than a user specified price. Prompt the user for price

LinkedList<Book> lbook = new LinkedList<Book>();

System.out.println("To list all books with price greater than a value, enter a price");

double pri = Double.parseDouble(sc.nextLine());

set = hmbook.entrySet();

for(Map.Entry<Integer, Book> s:set)

{ Book b1=s.getValue();

if(b1.price>pri)

{

lbook.add(b1);

}

}

6. For a given a value by the user, find all the books that match either the whole or a part of the book title.

String p = sc.nextLine();

set = hmbook.entrySet();

for(Map.Entry<Integer, Book> s:set)

{ Book b1=s.getValue();

if(isSubString(b1.title,p))

{

printBookData(b1);

}

}

7. Identify a publisher and print books from a particular publisher.

System.out.println("enter a publishers name to print book details");

p = sc.nextLine();

set = hmbook.entrySet();

for(Map.Entry<Integer, Book> s:set)

{ Book b1=s.getValue();

if(b1.publisher.equals(p))

{

printBookData(b1);

}

}

8.Update the publisher details based on a title.

System.out.println("enter a title whose publisher is to be updated");

String t = sc.nextLine();

System.out.println("enter the updated publishers name");

p = sc.nextLine();

set = hmbook.entrySet();

for(Map.Entry<Integer, Book> s:set)

{ Book b1=s.getValue();

if(b1.title.equals(t))

{

b1.publisher=p;

hmbook.put(s.getKey(), b1);

//printBookData(b1);

}

}

printHashMap();

}

**3. Create a desktop java application using swings to enable an user to enter student information such as name, usn, age, address, sgpa of 8 semesters, category. Perform validations on age and sgpa. Display appropriate messages in pop up boxes to indicate wrong entries, on clicking of the “compute” button. Also find the cgpa based on the obtained sgpa. On clicking of the “done” button, place the student details in a collection. A click on the “display” button should display the collection in a textarea.**

**Steps Involved:**

1. Create a student class

public class Student {

String name,usn,address,category;

int age;

//float sgpa1,sgpa2,sgpa3,sgpa4,sgpa5,sgpa6,sgpa7,sgpa8;

float cgpa;

public Student(String name,String usn,String address,String cat,int age, float cgpa)

{

this.name=name; this.usn=usn;

this.address=address; this.category=cat;

this.age=age;

this.cgpa=cgpa;

}

public String toString()

{

String stud= name + " "+usn+" residing in "+address+" belonging to category "+category+" of age "+age;

stud+= "has cgpa "+cgpa;

return stud;

}

}

2. Write a class that creates objects of the swing components label, textfields, buttons. Let this class implements ActionListener . Also create LinkedList of Student.

3. Add these components to the JFrame.

4. In actionPerformed(ActionEvent evt) method

if(evt.getSource()==submit)

{

//check for validations

try

{

int v1=Integer.parseInt(age.getText());

if(v1<18 || v1>35)

{

String age1=JOptionPane.showInputDialog(null,"Enter valid Age");

age.setText(age1);

}

}

catch(NumberFormatException e) {

JOptionPane.showMessageDialog(f1, "Invalid entry");

age.requestFocus();

}

// Check for the validity of the SGPA

checkSGPA\_valid(1,sgpa1);

checkSGPA\_valid(2,sgpa2);

checkSGPA\_valid(3,sgpa3);

checkSGPA\_valid(4,sgpa4);

checkSGPA\_valid(5,sgpa5);

checkSGPA\_valid(6,sgpa6);

checkSGPA\_valid(7,sgpa7);

checkSGPA\_valid(8,sgpa8);

float cal\_cgpa = calculate\_cgpa();

cgpa.setText(Float.toString(cal\_cgpa));

}

else if(evt.getSource()==done)//to submit into collection

{

Student s1 = new Student(name.getText(),usn.getText(),address.getText(),String.valueOf(cat.getSelectedItem()),Integer.parseInt(age.getText()),Float.parseFloat(cgpa.getText()));

stud\_list.add(s1);

}

stud\_list\_display.setText(" ");

for(Student s:stud\_list)

{

stud\_list\_display.append(s.toString()+ "\n");

}

}

}

**4. Write a java program using swings to validate user login information using dialog boxes. Once validated, allow the user to enter the customer id, if the person is a new customer, else check whether the customer exists in a collection and obtain the customer id. The customer id can be obtained given a mobile number. Allow the user to enter the item purchased by giving the item id and quantity purchased. On clicking of a button, the item name and the total cost should appear in the corresponding GUI components. Using option dialog box, indicate the types of discount available for the customer. On clicking on the print button, print the details in information dialog box.**

**Steps Involved**

1. Create a hashmap for holding the phone number and customer id with respect to phone number.

HashMap<Long,Integer> hm = **new** HashMap<Long,Integer>();

2. Create an arraylist for holding item details

**static** ArrayList<ItemDetails> *ar\_items* = **new** ArrayList<ItemDetails>();

3. Create dialog boxes to prompt the user to give username and password

String uname = JOptionPane.*showInputDialog*(**null**,"Enter username");

String passwd = JOptionPane.*showInputDialog*(**null**,"Enter Password");

4. Check validity of the entered data. If present, make the frame visible that holds the customer id, customer phone number , item id and quantity purchased and button for proceeding.

5. Onclick of the button, check whether the item is available in the ar\_items list. If yes,

String[] ch = {"10%", "20%", "30%"};

int i = JOptionPane.showOptionDialog(null, "Enter the Discount", "Discount", 0, JOptionPane.INFORMATION\_MESSAGE, null, ch, ch[0]);

double total = Double.parseDouble(itemCostArea.getText()) \* Double.parseDouble(itemQtyField.getText())

- (Double.parseDouble(itemCostArea.getText()) \* Double.parseDouble(itemQtyField.getText())) \* (i+1) \* 10 / 100;

compute.setText(" Computed: " + total);

JOptionPane.showMessageDialog(null, String.valueOf(total), "Total", JOptionPane.INFORMATION\_MESSAGE);

**5. Write a program that uses Java Swing and JDBC to create a stand-alone application:**

**Create two tables namely, Representative (RepNo, RepName, State, Comission, Rate) and Customer (CustNo, CustName, State, Credit\_Limit, RepNo) in MySQL database.**

**Use appropriate Swing components to insert values in a form.**

**Use another form to display Representative’s information when Credit\_Limit is above 15,000.**

1. Run mySQL by using command mysql -u root -p

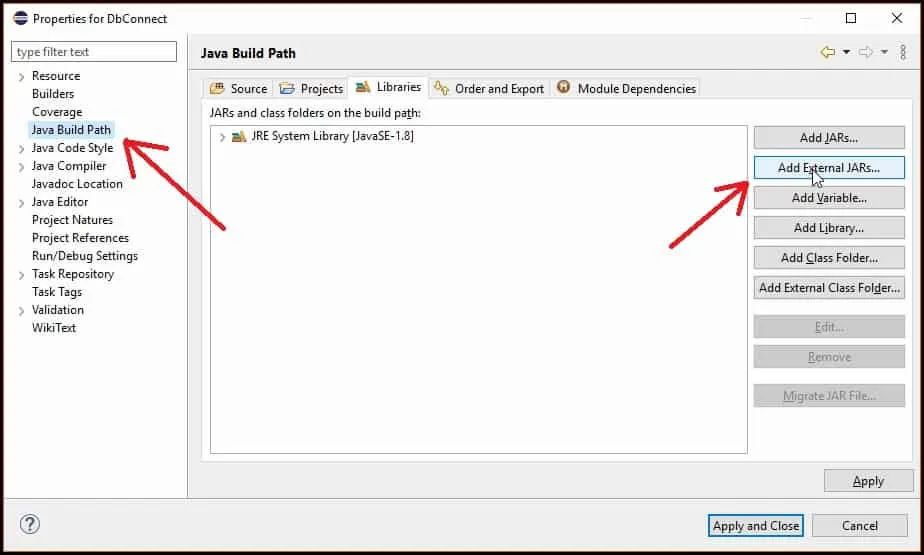
2. Create a database in mysql using mysql commands

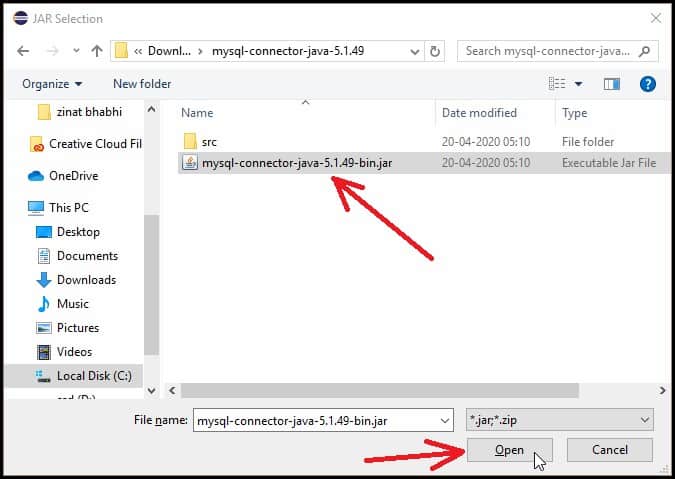
3. Create 2 tables Representative and Customer tables in mysql.

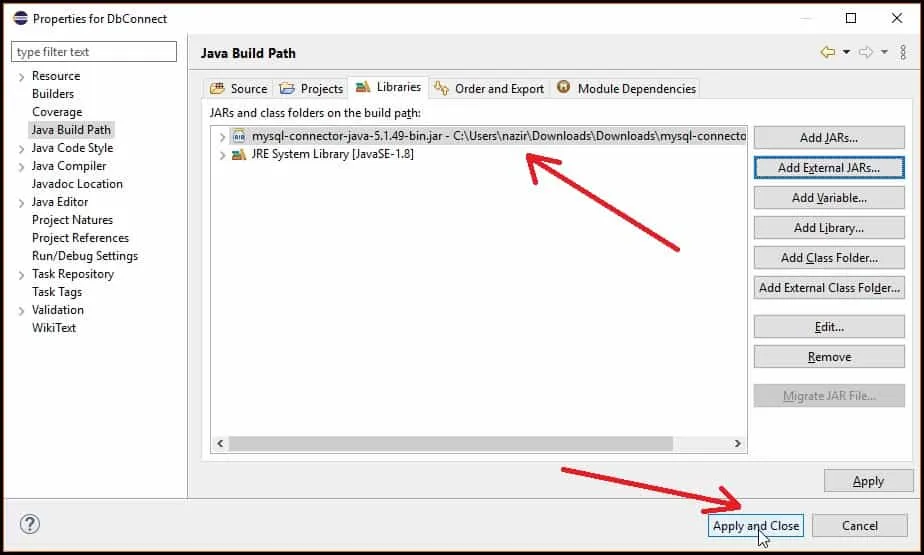
4. Create the swing component objects for entering the details into the database. Add all these objects to the panel

5. In order to connect your java program with MySQL database, you need to include MySQL JDBC driver which is a JAR file, namely mysql-connector-java.jar.

Right click on the project -> right click->build path







6. Add the code for connectivity and querying the databases

Statement stmt;

Class.*forName*("com.mysql.jdbc.Driver");

Connection conn = DriverManager.*getConnection*("jdbc:mysql://localhost:3306/datab", "root", "pass");

7. Execute the query

a. stmt = (Statement) conn.createStatement();

System.*out*.println(Repno + Repname + state + commission);

String query1 = "insert into Representative values('"Repno + "','" + Repname + "','" + state+ "','" + commission + "','" + rate + "');";

stmt.executeUpdate(query1);

b. stmt2 = (Statement) conn.createStatement();

System.*out*.println(Custno + CustName + state + cr + Rno);

String query2 = "insert into Customer values('"+ Custno + "','" + CustName + "','" + state+ "','" + cr + "','" + Rno + "');";

stmt2.executeUpdate(query2);

c. stmt = (Statement) conn.createStatement();

String query3="SELECT \* FROM Representative WHERE RepNo IN (SELECT RepNo FROM Customer WHERE Credit\_Limit > 15000 )";

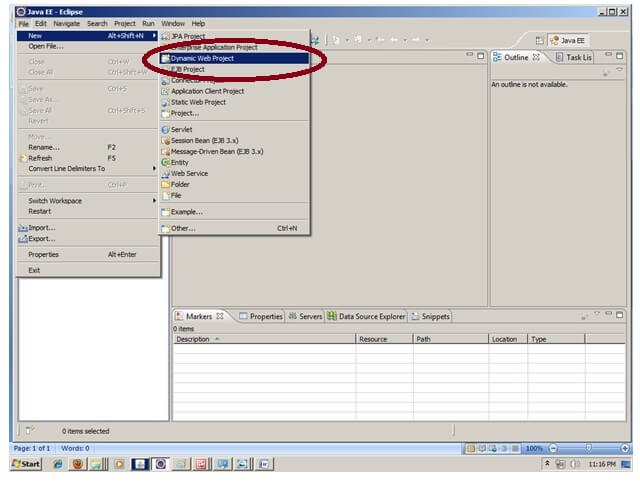
ResultSet rs = stmt.executeQuery(query3);

Display the resultset in a text area.

**6. Create a Servlet to file IT returns that accepts personal information, salary information and Tax deduction details from the user and write the information into a file. Also accept the name of the person and display in on the page**.

**Steps Involved**

1. Create a project – Dynamic Web Project



2. Create a **Servlet** within this project.

public class CalculateTax extends HttpServlet {

3. Import the following packages

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

4. Define the toGet method to read the parameters entered through the HTML form

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException

{

String name = request.getParameter("name"); String gender = request.getParameter("gender"); String salary = request.getParameter("salary");

5. Use them to calculate the taxes and salary

6. Append this to be copied back using response

response.getWriter().append("your name is" + name + "\n") .append("your gender is" + gender + "\n") .append("your salary is" + salary+ "\n" ) .append("your finaltax is" + finaltax+ "\n");

7. Using PrintWriter object copy this to a file

String newFile = new File("data.txt").getAbsolutePath(); System.out.println("your new file is "+newFile); PrintWriter pw = new PrintWriter("data.txt");

pw.println("your name is " + name);

pw.println("your gender is " + gender);

pw.println("your salary is " + salary);

pw.println("your finaltax is " + finaltax);

8. Write the HTML file

<!DOCTYPE html> <html>

<head> <meta charset="UTF-8">

<title>Income Tax Form</title> </head> <body>

<form action="CalculateTax" method="get">

<h3>Income Tax Form</h3>

Name<input type="text" name="name">

<br> Gender: Male <input type="radio" name="gender" value="Male">

Female <input type="radio" name="gender" value="Female">

<br> Salary: <input type="number" min="0" name="salary">

<br> Tax Deductions: <input type="number" name="deduction">

<br> <input type="submit" value="Submit"/> </form>

</body> </html>

9. For adding a jar file, right click on your project -> Build Path -> Configure Build Path -> click on Libraries tab in Java Build Path -> click on Add External JARs button -> select the servlet-api.jar file under tomcat/lib -> ok.

10. Right click on your project -> Run As -> Run on Server -> choose tomcat server -> next -> addAll -> finish.

11. For configuring the tomcat server in eclipse IDE, click on servers tab at the bottom side of the IDE -> right click on blank area -> New -> Servers -> choose tomcat then its version -> next -> click on Browse button -> select the apache tomcat root folder previous to bin -> next -> addAll -> Finish.

**7. Create a servlet that accepts patient information in a hospital such as patient id, patient name, age, date of admission, cause of admission, doctor diagnosed, treatment proposed. Place the details into a database. Allow options to insert, update, view and delete the contents in the database.**

1. After creating Dynamic Web Project and importing the necessary packages

Define the doPost method to retrieve the parameters read through the web page

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

String name = request.getParameter("name");

String age = request.getParameter("age");

String date = request.getParameter("date");

String cause = request.getParameter("cause");

String doctor = request.getParameter("doc");

String treat = request.getParameter("treat");

2. Connect to the Patient databases and execute query to insert

try {

Class.forName("com.mysql.jdbc.Driver");

Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/java", "root", "root");

String updateDB = "insert into PatientRecords values(?,?,?,?,?,?);";

PreparedStatement ps = con.prepareStatement(updateDB);

ps.setString(1, name);

ps.setString(2, age);

ps.setString(3, date);

ps.setString(4, cause);

ps.setString(5, doctor);

ps.setString(6, treat);

int i = ps.executeUpdate();

System.out.println("Rows inserted: "+i); s = name + "\n" + age + "\n" + date + "\n" + cause + "\n" + doctor + "\n" + treat + "\n" + i;

} catch(Exception e){

System.out.println(e); }

request.setAttribute("data", s);

3. Forward to the view.jsp file to view the data

request.getRequestDispatcher("/view.jsp").forward(request,response);

4. Create Front Ends

<!DOCTYPE html> <html>

<head> <meta charset="ISO-8859-1">

<title>Insert Patient Details</title> </head> <body>

<div align='center'>

<form action='PatientServlet' method='post'>

Name: <input type='text' name='name'/><br>

<br> Age: <input type='number' name='age'/>

<br><br> Date of Admission: <input type='date' name='date'/>

<br><br> Cause of Admission: <input type='text' name='cause'/>

<br><br> Doctor Diagnosed: <input type='text' name='doc'/>

<br><br> Treatment Proposed: <input type='text' name='treat'/>

<br><br> <input type="submit" value='update'/> </form> <br>

<br><br><br> <form action='GetServlet' method='post'>

Get Current Database data: <input type='submit' value='Get Data'/>

</form> </div> </body> </html>

**8. Write a JSP and Servlet Program to do the following to buy a T-Shirt online:**

**a. A set of checkboxes to select your T-Shirt accessories such as ‘belt’, ‘cap’, ‘hair-band’ etc.**

**b. A text area / text field to enter your T-Shirt tag-line**

**c. A Radio-button that allows the user to choose between T-Shirt with chest pocket and without.**

**d. A Combo Box to choose your T-Shirt color**

**e. Appropriate labels for these GUI Components**

**f. A Button called “Click Me” which when pressed will**

**g. Insert the details entered into a table called ‘TShirts’.**

**h. An OrderNo is generated by adding ‘1’ to the existing ‘OrderNo’**

**i. If ‘TShirts’ table is empty the initial value of ‘OrderNo’ is 100.**

**j. This ‘OrderNo’ is also inserted into the ‘TShirts’ table**

**k. Display all the records of the ‘TShirts’ table in tabular form**

**PS: Frontend display should be in JSP and the business logic should be written in Servlet Class.**

**Steps Involved**

1. Write the servlet program to access the parameters . @WebServlet("/tcon")

protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {

String acc[]=req.getParameterValues("acc");

String accList=""; for(String s:acc)

accList=accList+s+"/";

String tagline=req.getParameter("tag");

String pocket=req.getParameter("opt");

String color=req.getParameter("col");

System.out.println(tagline+ " " + pocket);

resp.setContentType("text/html");

PrintWriter out=resp.getWriter();

2. Establish database connectivity and insert into TShirts table

Class.forName("com.mysql.jdbc.Driver");

Connection conn=DriverManager.getConnection("jdbc:mysql://localhost:3306/TShirts", "root", "root");

Statement stmt=conn.createStatement();

stmt.executeUpdate("insert into TShirts values ("+i+",'"+accList.substring(0,10)+"','"+tagline+"','"+pocket+"','"+color+"');");

3. Create a JSP file in WebContent folders

<form action="tcon" method="post">

T-Shirt Accessories<input type="checkbox" name="access" value="Belt"/>Belt

<input type="checkbox" name="access" value="Cap"/>Cap

<input type="checkbox" name="access" value="Hair-Band"/>Hair-Band<br>

Tag-Line<input type="text" name="tagline" size="50"/><br>

T-Shirt Feature:<input type="radio" name="pocket" value="ChestPocket"/>Chest Pocket

<input type="radio" name="pocket" value="NoChestPocket"/>No Chest Pocket<br>

T-Shirt Color:<select name="Tshirtcolor">

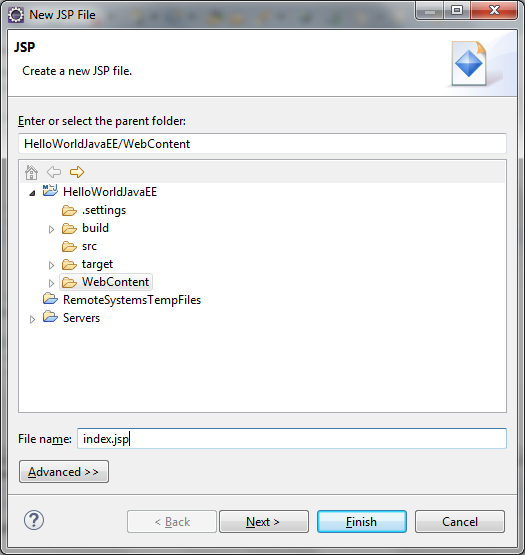
<option>Blue</option>

<option>Red</option>

<option>Green</option>

</select><br>

<input type="submit" value="Place Orders"/>



4. Write code to create a table in Servlet program for displaying the result based on the result set rs.

out.print("<td>OrderNo</td>");

out.print("<td>T-shirt Accessories</td>");

out.print("<td>T-shirt tag-line</td>");

out.print("<td>T-shirt type</td>");

out.print("<td>T-shirt color</td>");

out.println("</tr>");

if(!rs.isBeforeFirst()){

out.print("<tr>");

out.print("<td>100</td>");

out.print("<td>NULL</td>");

out.print("<td>NULL</td>");

out.print("<td>NULL</td>");

out.print("<td>NULL</td>");

out.print("<td>NULL</td>");

out.println("</tr>");

}

while(rs.next()){

out.println("<tr>");

out.print("<td>"+(Integer.parseInt(rs.getString("OrderNo"))+100)+"</td>");

out.print("<td>"+rs.getString("tshritAccessories")+"</td>");

out.print("<td>"+rs.getString("tshritTagLine")+"</td>");

out.print("<td>"+rs.getString("tcolor")+"</td>");

out.print("<td>"+rs.getString("tshritOption")+"</td>");

out.println("</tr>");

}

out.println("</table>");

out.println("<a href=\"tshirt.jsp\">click here</a>");

# 9. Write a Java Program that creates two threads object of Thread class. Where one thread asks the user to enter a number not less than four digits. Split the digits of the number and display in words the value of the number. Ex: 1 – One. Second thread finding the number of vowels in a word. Ex: JAVA – Vowel - A, Count – 2.

**Steps Involved**

1. Create a NumberThread class implementing Runnable class . In this class create a hashmap of numbers from 0 to 9.

class NumberThread implements Runnable {

private HashMap<Integer,String> numberMap = new HashMap<>();

public NumberThread() {

numberMap.put(1, "One");

numberMap.put(2, "Two");

numberMap.put(3, "Three");

numberMap.put(4, "Four");

numberMap.put(5, "Five");

numberMap.put(6, "Six");

numberMap.put(7, "Seven");

numberMap.put(8, "Eight");

numberMap.put(9, "Nine");

numberMap.put(0, "Zero"); }

2. Override run() method to identify digits by using charAt() function.- int digit = Integer.parseInt("" + number.charAt(i));

Identify the digit - numberMap.get(digit)

Repeat for each digit

3. Similarly VowelsThread class that implements Runnable

4. Create and arraylist of Vowels

class VowelsThread implements Runnable {

private ArrayList<Character> vowels = new ArrayList<>();

public VowelsThread() {

vowels.add('a'); vowels.add('e'); vowels.add('i'); vowels.add('o'); vowels.add('u'); }

5. Override run method to Read the word . Convert to lowercase.

6. Read character by character. Check with vowels in arraylist and classify them as vowels. Else consonants

7. Write a main program to create objects of NumberThread and VowelsThread class and called the start() method.

public static void main(String[] args) {

NumberThread thread1 = new NumberThread(); VowelsThread thread2 = new VowelsThread();

Thread t1 = new Thread(thread1); Thread t2 = new Thread(thread2); t1.start(); t2.start(); }

}

10. **Create a Customer Application with Spring framework. The Customer class should include attributes: name, id, Items purchased . Class Item has the following attributes: name, id, price, tax, manufacture date. Create objects and calculate the total price of the items purchased by the customer.**

1. Create Database

Use MySQL database. The application will manage data in a table named customer. The table customer has 4 fields: id, name, email and address

2. Create a Dynamic Web Project in Eclipse, and convert it to Maven project: right-click on the project, select Configure > Convert to Maven Project. The Create new POM dialog appears. Enter the following information:

- Group Id: net.codejava

- Artifact Id: CustomerManager

Make sure that the JRE version for the project is Java 8 or newer.

Next, open the Maven's project file pom.xml to configure the dependencies for the project. Declare properties for the version of Spring and Hibernate frameworks:

<properties>

<spring.version>5.1.5.RELEASE</spring.version>

<hibernate.version>5.4.1.Final</hibernate.version>

</properties>

**dependency for the core of Spring framework:**

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>${spring.version}</version>

</dependency>

**For web development with Spring MVC:**

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>${spring.version}</version>

</dependency>

To use Spring Data JPA, specify the following dependencies:

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-orm</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework.data</groupId>

<artifactId>spring-data-jpa</artifactId>

<version>2.1.5.RELEASE</version>

</dependency>

**use Hibernate as a provider of JPA, so add the following dependency:**

<dependency>

<groupId>org.hibernate</groupId>

<artifactId>hibernate-core</artifactId>

<version>${hibernate.version}</version>

</dependency>

**dependency for MySQL JDBC driver:**

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

<version>8.0.14</version>

<scope>runtime</scope>

</dependency>

**And for Java Servlet, JSP and JSTL:**

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>javax.servlet-api</artifactId>

<version>3.1.0</version>

<scope>provided</scope>

</dependency>

<dependency>

<groupId>javax.servlet.jsp</groupId>

<artifactId>javax.servlet.jsp-api</artifactId>

<version>2.3.1</version>

<scope>provided</scope>

</dependency>

<dependency>

<groupId>jstl</groupId>

<artifactId>jstl</artifactId>

<version>1.2</version>

</dependency>

You can see the complete code of the pom.xml file in the sample project attached along this tutorial.

Create two Java packages under the source folder:

- net.codejava.config: for configuration classes.

- net.codejava.customer: for application-specific classes.

3. Create JPA Configuration File

Since JPA is used, we need to specify database connection properties in the persistence.xml file instead of hibernate.cfg.xml file. Create a new directory named META-INF in the source folder to put the persistence.xml file as follows:

persistence xml file

And write XML code for the persistence.xml file like this:

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

<persistence xmlns="http://xmlns.jcp.org/xml/ns/persistence"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence

http://xmlns.jcp.org/xml/ns/persistence/persistence\_2\_1.xsd"

version="2.1">

<persistence-unit name="SalesDB">

<properties>

<property name="javax.persistence.jdbc.url" value="jdbc:mysql://localhost:3306/sales" />

<property name="javax.persistence.jdbc.user" value="root" />

<property name="javax.persistence.jdbc.password" value="password" />

<property name="javax.persistence.jdbc.driver" value="com.mysql.jdbc.Driver" />

<property name="hibernate.show\_sql" value="true" />

<property name="hibernate.format\_sql" value="true" />

</properties>

</persistence-unit>

</persistence>

As you can see, we specify database connection properties such as URL, user, password and JDBC driver class. Note that the persistence unit name SalesDB will be used in the configuration code.

4. Code Model Class

Create the domain class Customer to map with the table customer in the database as following:

package net.codejava.customer;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

@Entity

public class Customer {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String name;

private String email;

private String address;

protected Customer() {

}

protected Customer(String name, String email, String address) {

this.name = name;

this.email = email;

this.address = address;

}

// getters and setters are not shown for brevity

}

As you can see, we use the annotation @Entity to map this class to the table customer (the class has same name as the table). All the class' field names are also identical to the table's ones. The field id is annotated with @Id and @GeneratedValue annotations to indicate that this field is primary key and its value is auto generated.

5. Code Configuration Classes

Next, let's write some Java code to configure Spring MVC and Spring Data JPA. We use Java-based configuration as it's simpler than XML.

Configure Spring Dispatcher Servlet:

To use Spring MVC for our Java web application, we need to register the Spring Dispatcher Servlet upon application's startup by coding the following class:

package net.codejava.config;

import javax.servlet.ServletContext;

import javax.servlet.ServletException;

import javax.servlet.ServletRegistration;

import org.springframework.web.WebApplicationInitializer;

import org.springframework.web.context.support.AnnotationConfigWebApplicationContext;

import org.springframework.web.servlet.DispatcherServlet;

public class WebAppInitializer implements WebApplicationInitializer {

@Override

public void onStartup(ServletContext servletContext) throws ServletException {

AnnotationConfigWebApplicationContext appContext = new AnnotationConfigWebApplicationContext();

appContext.register(WebMvcConfig.class);

ServletRegistration.Dynamic dispatcher = servletContext.addServlet(

"SpringDispatcher", new DispatcherServlet(appContext));

dispatcher.setLoadOnStartup(1);

dispatcher.addMapping("/");

}

}

The onStartup() method of this class will be automatically invoked by the servlet container when the application is being loaded. The Spring Dispatcher Servlet handles all the requests via the URL mapping "/" and it looks for configuration in the WebMvcConfig class, which is described below.

Configure Spring MVC:

Create the WebMvcConfig class under the net.codejava.config package with the following code:

package net.codejava.config;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.ComponentScan;

import org.springframework.context.annotation.Configuration;

import org.springframework.web.servlet.view.InternalResourceViewResolver;

@Configuration

@ComponentScan("net.codejava ")

public class WebMvcConfig {

@Bean(name = "viewResolver")

public InternalResourceViewResolver getViewResolver() {

InternalResourceViewResolver viewResolver = new InternalResourceViewResolver();

viewResolver.setPrefix("/WEB-INF/views/");

viewResolver.setSuffix(".jsp");

return viewResolver;

}

}

This class is annotated with the @Configuration annotation to tell Spring framework that this is a configuration class. The @ComponentScan annotation tells Spring to scan for configuration classes in the net.codejava package.

In this class, we simply create a view resolver bean that specifies the prefix and suffix for view files. So create the directory views under WebContent/WEB-INF directory to store JSP files.

You can add more Spring MVC configurations here.

Configure Spring Data JPA:

To enable Spring Data JPA, we need to create two beans: EntityManagerFactory and JpaTransactionManager. So create another configuration class named JpaConfig with the following code:

package net.codejava.config;

import javax.persistence.EntityManagerFactory;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.data.jpa.repository.config.EnableJpaRepositories;

import org.springframework.orm.jpa.JpaTransactionManager;

import org.springframework.orm.jpa.LocalEntityManagerFactoryBean;

import org.springframework.transaction.annotation.EnableTransactionManagement;

@Configuration

@EnableJpaRepositories(basePackages = {"net.codejava.customer"})

@EnableTransactionManagement

public class JpaConfig {

@Bean

public LocalEntityManagerFactoryBean entityManagerFactory() {

LocalEntityManagerFactoryBean factoryBean = new LocalEntityManagerFactoryBean();

factoryBean.setPersistenceUnitName("SalesDB");

return factoryBean;

}

@Bean

public JpaTransactionManager transactionManager(EntityManagerFactory entityManagerFactory) {

JpaTransactionManager transactionManager = new JpaTransactionManager();

transactionManager.setEntityManagerFactory(entityManagerFactory);

return transactionManager;

}

}

Here, two important annotations are used:

- @EnableJpaRepositories: this tells Spring Data JPA to look for repository classes in the specified package (net.codejava) in order to inject relevant code at runtime.

- @EnableTransactionManagement: this tells Spring Data JPA to generate code for transaction management at runtime.

In this class, the first method creates an instance of EntityManagerFactory to manage the persistence unit SalesDB (this name is specified in the persistence.xml file above).

And the last method creates an instance of JpaTransactionManagerfor the EntityManagerFactory created by the first method.

That's the minimum required configuration for using Spring Data JPA.

6. Code Respository Interface

Next, create the CustomerRepository interface that extends the CrudRepository interface defined by Spring Data JPA with the following code:

package net.codejava.customer;

import java.util.List;

import org.springframework.data.repository.CrudRepository;

import org.springframework.data.repository.query.Param;

public interface CustomerRepository extends CrudRepository<Customer, Long> {

}

Declare an interface that extends the CrudRepository interface, which defines CRUD methods like save(), findAll(), findById(),deleteById(), etc. At runtime, Spring Data JPA automatically generates the implementation code.

Note that we must specify the type of the model class and type of the primary key field when extending the CrudRepository interface: CrudRepository<Customer, Long>

7. Code Service Class

Next, code the CustomerService class in the business/service layer with the following code:

package net.codejava.customer;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import org.springframework.transaction.annotation.Transactional;

@Service

@Transactional

public class CustomerService {

@Autowired CustomerRepository repo;

public void save(Customer customer) {

repo.save(customer);

}

public List<Customer> listAll() {

return (List<Customer>) repo.findAll();

}

public Customer get(Long id) {

return repo.findById(id).get();

}

public void delete(Long id) {

repo.deleteById(id);

}

}

Note that this class is annotated with the @Transactional annotation so all of its methods will be intercepted by Spring Data JPA for transaction management. And an instance of CustomerRepository interface will be injected into this class:

@Autowired CustomerRepository repo;

This is like magic, as we don't write any DAO code but Spring Data JPA will generate an implementation automatically at runtime.

And as you can see, all the methods in this class are for CRUD operations. It simply delegates all the call to a CustomerRepositoryobject. This class seems to be redundant, but it is needed to decouple the business/service layer from the repository/DAO layer.

8. Code Spring MVC Controller Class

Next, in the controller layer, create the CustomerController class to handle all requests from the clients with the following code:

package net.codejava.customer;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Controller;

@Controller

public class CustomerController {

@Autowired

private CustomerService customerService;

// handler methods will go here...

}

This is a typical Spring MVC controller class, which is annotated with the @Controller annotation. You can see an instance of CustomerService is injected into this class using the @Autowired annotation.

We will write code for the handler methods in the following sections.

9. Code List Customer Feature

The application's home page displays all customers, so add the following handler method to the CustomerController class:

@RequestMapping("/")

public ModelAndView home() {

List<Customer> listCustomer = customerService.listAll();

ModelAndView mav = new ModelAndView("index");

mav.addObject("listCustomer", listCustomer);

return mav;

}

And code the view page (index.jsp) as follows:

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

pageEncoding="ISO-8859-1"%>

<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01

Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">

<title>Customer Manager</title>

</head>

<body>

<div align="center">

<h2>Customer Manager</h2>

<form method="get" action="search">

<input type="text" name="keyword" /> &nbsp;

<input type="submit" value="Search" />

</form>

<h3><a href="/new">New Customer</a></h3>

<table border="1" cellpadding="5">

<tr>

<th>ID</th>

<th>Name</th>

<th>E-mail</th>

<th>Address</th>

<th>Action</th>

</tr>

<c:forEach items="${listCustomer}" var="customer">

<tr>

<td>${customer.id}</td>

<td>${customer.name}</td>

<td>${customer.email}</td>

<td>${customer.address}</td>

<td>

<a href="/edit?id=${customer.id}">Edit</a>

&nbsp;&nbsp;&nbsp;

<a href="/delete?id=${customer.id}">Delete</a>

</td>

</tr>

</c:forEach>

</table>

</div>

</body>

</html>

10. Code Create New Customer Feature

To implement the create new customer feature, we need to write two handler methods. The first one is to display the new customer form:

@RequestMapping("/new")

public String newCustomerForm(Map<String, Object> model) {

Customer customer = new Customer();

model.put("customer", customer);

return "new\_customer";

}

And write code for the JSP page new\_customer.jsp as follows:

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">

<title>New Customer</title>

</head>

<body>

<div align="center">

<h2>New Customer</h2>

<form:form action="save" method="post" modelAttribute="customer">

<table border="0" cellpadding="5">

<tr>

<td>Name: </td>

<td><form:input path="name" /></td>

</tr>

<tr>

<td>Email: </td>

<td><form:input path="email" /></td>

</tr>

<tr>

<td>Address: </td>

<td><form:input path="address" /></td>

</tr>

<tr>

<td colspan="2"><input type="submit" value="Save"></td>

</tr>

</table>

</form:form>

</div>

</body>

</html>

**References**

# Text Books

# 1. Herbert Schildt, ‘The Complete Reference Java (J2SE 5 Edition)’, TATA McGRAW-HILL Edition 2005.

# 2. Ivan Bayross, Sharanam Shah, Cyntiha Bayross and Vishali Shah, ‘Java EE 5 for Beginners’, SPD (Sharoff Publishers & Distributors Pvt. Ltd.), 2nd edition August 2008.

# Reference:

# 1. Jim Keogh,The Complete Reference J2EE‟, TATA McGRAW-HILL Edition 2002.

# 2. B V Kumar, S Sangeetha, S V Subrahmanya, J2EE Architecture, TATA McGRAW-HILL Edition 2007.

[**https://www.codejava.net/frameworks/spring/spring-mvc-spring-data-jpa-hibernate-crud-example**](https://www.codejava.net/frameworks/spring/spring-mvc-spring-data-jpa-hibernate-crud-example)

[**https://www.javatpoint.com/creating-servlet-in-eclipse-ide**](https://www.javatpoint.com/creating-servlet-in-eclipse-ide)

[**https://www.edureka.co/blog/servlet-and-jsp-tutorial/**](https://www.edureka.co/blog/servlet-and-jsp-tutorial/)