Lab-1

Compare abstract class and interface in terms of multiple inheritence. When would I prefer to use an abstract class and when an interface?

Answer :-

Introduction:- Abstract class and interface are used in object oriented programming to achieve abstraction. This assignment compares abstract class and interface with respect to multiple inheritence and explain when to use each.

Abstract Class :- An abstract class may contain both abstract and concrete methods.

- Support single inheritence only.
- Can have variable and constructors.
- Use the extends keyword.

Interface :- An interface containg method declaration only.

- Supports multiple inheritence
- A class can implement interfaces.

- Cannot have instance variable or constructors.
- Uses the implements keyword.

Comparison table :-

Abstract Class	Interface
Multiple inheritence not supported.	Multiple inheritence supported
Abstract + concrete	Abstract only
Variable Allowed	variable constant only.
Constructor Allowed	Constructor not allowed

Use abstract class when :-
- Classes are closely related
- Code reuse needed.
- Multiple inheritence needed.

Use Interface when :
- Multiple inheritance is needed.
- Only method declaration is required
- Unrelated one unrelated but share behaviour.

Conclusion: Abstract classes are used for sharing common code while interface are used to support multiple inheritance and define a common methods. The choice depends on system requirement.

Lab 2 :-

How does encapsulation ensure data security and integrity ?

Answer :- Encapsulation means wrapping data and methods together and restricting direct access to the data using access modifier like private.

Data security :-

- Variables are declared private, so they cannot be accessed from outside the class.
- only controlled access allowed through public methods.

Data integrity :-

- setter method validate input before updating data.
- Invalid values (null, empty, negative) are rejected keeping the oject in valid state.

Example :-

```
Class BankAccounts {
    private string accountNumbers;
    private double balace;
  private void SetAccountNumber(String accountNumber)
    {
        if (accountNumber == null || accountNumber.isempty
            {
                System.out.println("Invalid account Num");
            }
        else {
            this.accountNumber = accountNumber;
        }
    }
  public void SetInitialBalance(double balance){
        if (balance<0)
            {
                System.out.println("Blance cannot be negat
            }
        else {
            this.account = balance;
        }
    }
```

```
public string getAccountNumber () {
        return accountNumber;
    }
public double getBalance () {
        return balance;
    }
public double ge
}
```

Lab-3 :-

Multithreading system based carparking system.

Answer:-

Class Description :-

1. Registrarparking (Registrarparking.java).

```
Class Registrarparking {
        private string carNumber;

    public RegistrarParking(String CarNumber) {
      this.carNumber = CarNumber;
    }
    public string getCarNumber () {
          return carNumber;
      }
}
```

- Registercar parking represents a parking request made by a car.
- Stores car details (e.g. carNumber).

2. ParkingPool (ParkingPool.java)

```
import java.util.LinkedList;
import java.util.Queue;

class ParkingPool {
    private Queue<RegistrarParking> queue =
                                    new LinkedList();

    public synchronized void addCar(RegistrarParking car)
    {
        queue.add(car);
        System.out.println("car arrived " + car.getCar
                                    , getCarNumber());

        notify();
    }
public synchronized RegistrarParking ParkCar() {
        while(queue.isEmpty()) {
            try {
                wait();
            } catch(InterruptedException e) {
                e.printStackTrace();
            }
            return queue.poll();
        }
    }
```

3. ParkingAgent (ParkingAgent.java)

```
Class ParkingAgent extends Thread {
    private ParkingPool;
    public ParkingAgent(ParkingPool pool) {
        this.pool=pool;
    }
    public void run() {
        while (true) {
            RegistrarParking car = pool.ParkCar();
            System.out.println("parking car" + car.getnumber();
        try {
            Thread.steep(1000);
        } catch (Interrupted Exception e) {
            e.printStackTrace();
        }
    }
}
```

4. Main class (Main.java)

```
Class Main {
    public static void main(String[] args) {
        ParkingPool pool = new ParkingPool();
        ParkingAgent agent = new ParkingAgent(pool);
        agent.start();
    for(int i=1; i<=5; i++){
        RegistrarParking car = new RegistrarParking("car"+i);
        pool.addcar(car);
    try{
        Thread.sleep(500);
    }catch(InterruptedException e){
        e.printStackTrace();
    }
    }
}
}
```

Lab 4 :-

Communication between java Application and relational Database using JDBC.

Answer:- JDBC is a standard Java API that allows java applications to interact with relational database such as MYSQL, Oracle. etc. It acts as a bridge between a java program and a database. JDBC uses driver to convert java method calls into database specific instruction

JDBC Architecture :-

1. Java application;
2. JDBC API,
3. JDBC Driver,
4. Relational Database.

Steps :-

step 1 : load JDBC driver
Step 2 :- Establish database connection
step 3 :- Create Statement
step 4 :- Execute select query
Step 5 :- Fetch result.
Step 6 :- Close Resource

Java Program:-

```
import java.sql.*;
public class SelectExample {
    public static void main(String[] args) {
        Connection con = null;
        Statement stmt = null;
        ResultSet rs = null;
    try {
        Class.forName("com.mysql.cj.jdbc.Driver");
        con = DriverManager.getConnection(
                "jdbc:mysql://localhost:3306/student",
                "root",
                "password");
        stmt = con.createStatement();
        rs = stmt.executeQuery("Select * from info");
        while(rs.next()) {
        System.out.println(rs.getInt("Id") + " " + rs.getString
                                                    ("name"));
        }
    } catch (ClassNotFoundException e) {
        System.out.println("JDBC Driver not found");
    }
```

```
catch (SQLException e){
    system.out.println("Database error occured");
  }
  finally {
    try {
        if (rs != null) rs.close();
        if (stmt != null) stmt.close();
        if (con != null) con.close();
    } catch (SQLException e) {
        System.out.println("Error closing resource");
    }
  }
}
```

Lab 5 :-

In java EE applications, the MVC architecture is commonly used to seperate business logic from presentation logic. A servlet acts as the controller. JSP act on the view, and java classes or database logic represent the model. The servlet controller controls the application flow by recieving client request, interacting with the model and forwarding data to to the appropriate view.

Role of Servlet Controller :-

The servlet controller perforns the following:

- Recieving HTTP request from the client.
- Interacts with the model to process data.
- Stores data in the request object.
- Forwards the JSP request to JSP for rendering the response.

Forwarding Data from Servlet to JSP :-

```
import java.io.*;
import javax.servlet.*;
public class StudentServlet extends HttpServlet {
    protected void doGet(HttpServletRequest request,
                         HttpServletResponse response)
    throws ServletException, IOException {
        String name = "Sami";
        request.setAttribute("studentName", Name);
RequestDispatcher rd = request.getRequestDispatcher
                                    ("Student");
    }
}
```

Lab 6 :-

Performance and Security improvement prepared Statement.

Answer :-

Introduction :- In JDBC, SQL statements can be executed using either statements or preparedStatement while Statement is used for simple SQL queries. preparedStatement is a more advance and secure option that allows parameterized queries. It is widely used in real world java application.

preparedStatement Vs Statements :-

performance improvement :-

preparedStatement is pre-compiled by the database only once. When the same query is executed multiple times with different values, The database does not need to recompile the query, which improves performance.

Seun Security Improvements :-

preparedStatements helps prevents SQL injection attacks because user input is treated as data, not as the part of the SQL command. This makes application more secure compared using statements.

Advantage of prepared Statements :-

- Faster execution for repeated queries.
- prevents SQL injection attacks.
- Supports parameterized queries.
- Improves code readability and maintainability.

Example :-

```
import java.sql.*;
    public class InsertExample{
        public static void main(String[] args){
    connection con = null;
    preparedStatements ps = null;
    try{
    class.forName("com.mysql.cj.jdbc.Driver");
    con = DriverManager.getConnection(
        "jdbc:mysql://localhost:3306/student",
        "root"
        "password");
    String sql = "Insert into info values(? ?)";
    ps = con.prepareStatement(sql);
    PS.setInt(1, 101);
    ps.setString(2, "Sami");
    PS.executeUpdate();
    System.out.println("record inserted");
}catch(Exception e){
    System.out.println("Error occured: "+e);
}
```

```
finally {
    try {
        if (ps != null) ps.close();
        if (con != null) con.close();
    } catch (SQLException e) {
        System.out.println("Error cleaning resources");
    }
}
}
```

12

Lab - 7 :-

What is the resultset in JDBC and how is it used to retrive data from a MySQL database ?

Answer :-

In JDBC a ResultSet is essecially a table of data representing a database resultset. It is generated by executing a statement that queries the database.

Some Methods Explained.

.next() : The method moves the cursor forward one row from its current position. It returns a boolean : "True" if there is a valid row to read and false if there are no more rows.

• getString(String columnLabel) :- Retrieves the value of the designated column in the current row as a String you can also use the column index.

• getInt(String columnLabel) :- Similar to getString but it retrieves the value as an int. this is used for numeric data like IDs or Ages.

Code Example :-

```
ResultSet res = stmt.executeQuery("Select id,
                                   name from School student");
while (rs.next()) {
    int id = rs.getInt("id");
    String name = rs.getString("name");
System.out.println("ID": + id +" Name"+ name);
}
```

Lab-8:- How Spring Boot simplifies REST/UI web service development :-

Springboot simplifies the development of RESTful services by providing :-

- Auto-configuration: which reduces manual setup and configuration.
- Embedded servers, eliminating the need for external deployment.
- Annotation-based programming, making REST APIs easier to develop and maintain.
- Automatic JSON conversion using the Jackson library.

Implementing a REST controller in Spring Boot :-

A REST controller handles HTTP request and returns data in JSON format. Spring Boot automatically converts Java objects to JSON and vice versa.

Example :- (Simple controller with JSON)

```
import org.springframework.web.bind.annotation.*;
@RestController
@RequestMapping("/student")
public class StudentController{
    @GetMapping("/{id}")
    public student getStudent(@PathVariable int id){
        return new student{id, "Rahman", 21};
    }
    @PostMapping("/add")
    public student addStudent(@RequestBody student student)
    { return student;
    }
}
public class student {
    private int id, age;
    private string name;
public student(int id, String name, int age){
        this.id = id;
        this.name = name;
        this.age = age;
    }
}
```

Lab - 9 :-

Development of a simple graphical Calculator using python and tkinter.

Objective:- To design and implement a basic calculator application with a graphical user interface that performs addition, subtraction, multiplication and division.

Tools and technologies used :-

- Programming language python 3.
- GUI Library : Tkinter
- Development Environment : Any python IDE or text editor.

Important Code Snippets :-

1. Important window setup

```
from tkinter import *
root = Tk()
root.title("Simple Calculator")
root.geometry("320x420")
root.resizable(False, False)
```

2. Display and Global Expression :-

```
expression = " "
display = StringVar()
display_field = Entry (root, font = ("arial", 18),
                textvariable = display, bd = 10,
                insertwidth = 4, width = 14
                borderwidth = 4, Justify = "Right")

display_field.grid = (row = 0, column = 0,
                columnspan = 4, padx = 10,
                pady = 10 )
```

3. Button click functions :-

```
def press (num) :
    global expression
    expression += str(num)
    display.set (expression)

def equal () :
    global expression
    ~~dpoj~~
```

```
try:
    result = str(eval(expression))
    display.set(result)
    expression = result
except:
    display.set("Error")
    expression = ""
    display.set("")
```

4. GUI Button Layout—

```
# Row 1
Button(root, text='C', font=('arial', 14)
    commanden=clear, width=5).grid(row=1,
    column=0, padx=5, pady=5)
Button(root, text='/', font('arial', 14), common=
    lamba: press('/'), width=5).grid(
    row=1, column=3, padx=5, pady=5)

# Row 2-4.
Buttons = [('7', 2, 0), ('8', 2, 1), ('9', 2, 2),
    ('x', 2, 3), ('4', 3, 0), ('5', 3, 1),
    ('6', 3, 2), ('-', 3, 3), ('1', 4, 0),
    ('2', 4, 1), ('3', 4, 2), ('+', 4, 3),]
```

```
for (text, row, col) in buttons:-
    Button (root, text=text, font=('arial', 14)
    command=lambda, t=text: press(t); width=5)
    grid (row=row =column=col, padx=5,
     pady=5)
```