Learning Objective

Click to continue

We will learn the following things,

- To develop a CUI application using Java.
- Few of the software design patterns used in application development.
- Understand how to develop the middle tier using Java.
- Integrating the CUI application with the middle tier.

What is a Layered Architecture?

Click to continue

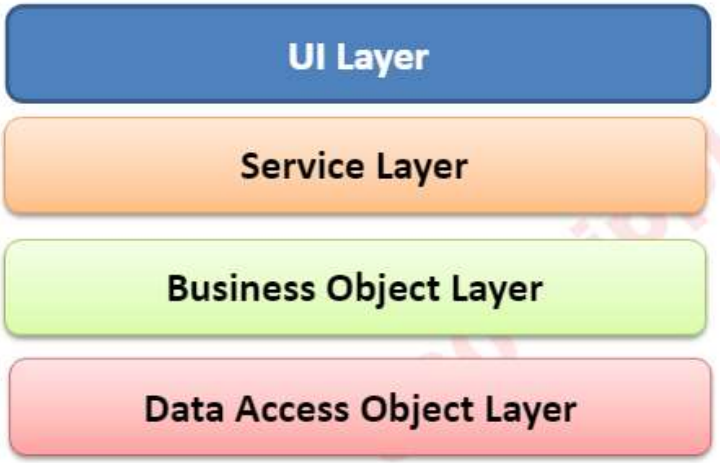
Software application is a collection of objects. The objects are logically grouped into units based on their core functionality.

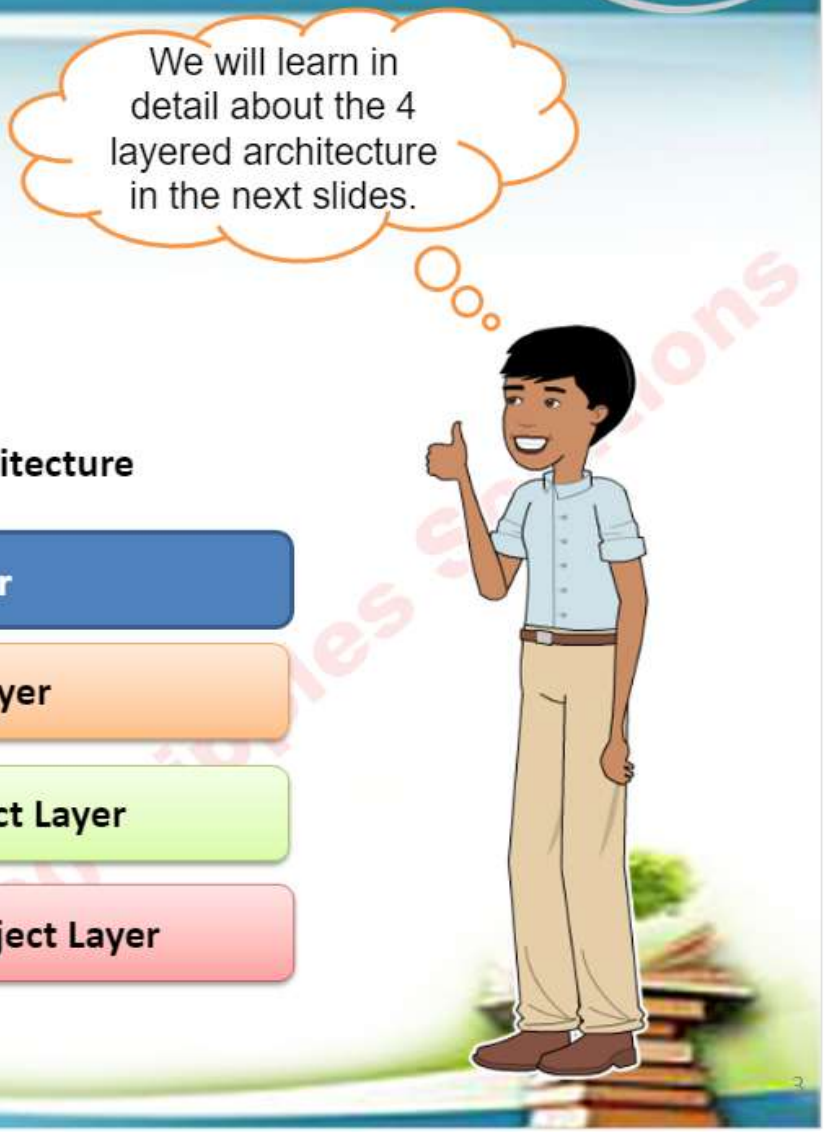
The groupings are represented as layers of applications, This is why we call it as ***layered application architecture***.

Benefits Layered Application:

- Easy to understand.
- Improves maintainability.
- Improves reusability.

4 Layered Architecture





Real Time Scenario

Scenario: Assume a theatre is opening a portal to book ticke… be booked the following flows should happen,

- ✓ Customer blocks the seats.
- ✓ Customer orders the snacks
- ✓ Customer pays the total fare though a payment gateway.

Successful transaction : Only if the above 3 transactions happens the ticket will be booked.

For Developer to develop the application, they need to develop components for all the 4 layers

Here we will develop UI as a character user interface (CUI) and build the other components using Java.

- **UI Layer** — Developer will develop any UI application like HTML, Angular , react or any UI framework
- **Service Layer**
- **Business Object Layer**
- **Data Access Object Layer**

All these 3 layers will be developed using Java or Python or PHP or any programming language..

Business Object Layer & Service Layer

Click to continue

Service Layer: This is the layer which will consolidate all the business flows of ticket booking process In our case this should consolidate 3 flows reserve seats, snacks booking and payment transaction.

TicketService

Developer will identify a TicketService class with a method bookTicket to encompass all the three business flows.

Business Object Layer: This is the layer which will perform the business logics in the application.

TicketBO | **SnackBO** | **PaymentBO**

Developer will create three components,

TicketBO - business object for reserving the tickets with a method **reserveTicket**

SnackBO - business object for pre-ordering snacks with a method **orderSnacks**.

PaymentBO - business object for paying the bill with a method **payBill**.

IMPORTANT: The service layer can have multiple methods for different transactions such as cancelling ticket, getting ticket details, print ticket etc.

Similarly the business object could have other method for example TicketBO will have methods such as cancel Ticket print ticket etc.

Data Access Object Layer & Transfer Objects

Click to continue

Data Access Object Layer: This is the layer which will perform all the database transactions for the appropriate business object. For example TicketBo needs to store the ticket details in database so it will use TicketDAO.

TicketDAO **SnackDAO** **PaymentDAO**

Developer will create three DAO,
TicketDAO - DAO for saving ticket details in database with method **addTicket**.
SnackDAO - DAO for saving snack orders in database with method **addOrder**.
PaymentDAO - DAO for saving payment details in database with method **storePayment.**

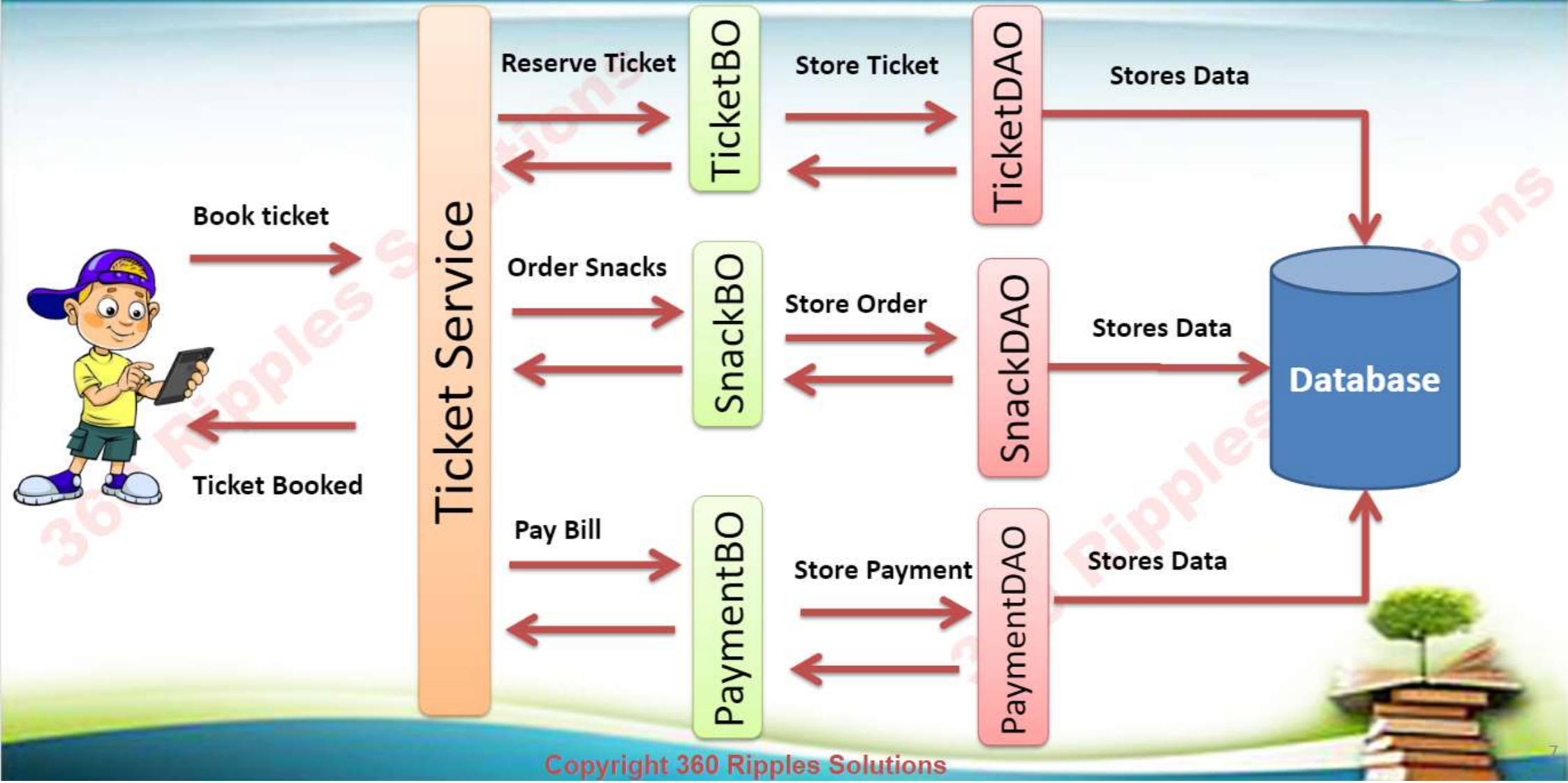
Transfer/Value Object : This is the object which will be used to send all the details from the UI to the back end. In our case the seat reserved, the snacks details and the payment details (card number,amount etc.)

TicketVO

This is the same value object which we learnt in Java basics course with getters and setters.

Overall Sequence Flow

Click to continue



Try it Out- Develop Our first CUI

Click to continue

For the configuration and demonstration of the CUI application please refer the video activity of this session

Create a CUI based application with the following menu's. Please use Scanner utility for the same.

- Add Students
- Fetch Student By Id

The student table will have the following attributes id , Name , Phone, Address, DOB (Java.sql.date), Standard.

The uses case should be built using the following layers

StudentMain→StudentService→ StudentBO → StudentDAO.

Requirement Details

Click to continue

Menu 1: Add student:

- This will capture the following details from the user name, DOB, phone, address, standard..
- When adding a student if any exception arises a user define Exception **StudentManagementException** should be thrown.
- If insertion is success system should display a message “Student added successfully”.

Menu 2 Fetch student by id:

- User will be asked to enter a student id and system should retrieve the student details and display it in the below format

```
==========================================================================
Student Name    Age     Standard        Fees     Contact Number  Address
==========================================================================
ABC             10      20000           977334444        Temple Road Chennai 87
```

If student does not exist for the given student id the system should throw a **StudentNotFounException**

HINT: Use \t escape sequence for space between each columns.

Requirement Details

Click to continue

Menu 1: Add student:

- This will capture the following details from the user name, DOB, phone, address, standard..
- When adding a student if any exception arises a user define Exception **StudentManagementException** should be thrown.
- If insertion is success system should display a message "Student added successfully".

Menu 2 Fetch student b

- User will be asked to ... display it in the below format

IMPORTANT: Since you are doing this for the first time it will be confusing but don't worry as you work on more case studies you will gain confidence.

```
==========================================================================
Student Name    Age     Standard        Fees     Contact Number  Address
==========================================================================
ABC             10      20000           977334444       Temple Road Chennai 87
```

If student does not exist for the given student id the system should throw a **StudentNotFounException**

HINT: Use \t escape sequence for space between each columns.

Try it Out- Develop the value object & Exception

Click to continue

Create the Value/Transfer object for transferring data from one layer to another layer.

```
package com.demo.vo;

import java.sql.Date;

public class StudentVO {
    private String sname;
    private Long phone;
    private String addres;
    private Date dob;
    private int standard;
    private int sid;

    public int getSid() {
        return sid;
    }

    public void setSid(int
        this.sid = sid;
    }
```

Package should be com.demo.vo

Add all the fields necessary for storing student details.

Generate the getter/setter

Create two exception classes as below. **StudentBOException** and **StudentDAOException**

```
package com.demo.exception;

public class StudentBOException extends Exception {

    public StudentBOException() {
    }

    public StudentBOException(String message) {
        super(message);
    }

    public StudentBOException(Throwable cause) {
        super(cause);
    }

    public StudentBOException(String s, Throwable cause) {
        super(cause);
    }

}
```

Package should be com.demo.exception

Refer the video for more details on creating the VO and Exception classes.

Try it Out- Develop the DAO

Click to continue

Develop the **StudentDAO** add two methods **addStudent** and **fetchStudent** details.

Package should be com.demo.dao

Import all the required libraries using organize imports - ctrl+shift+o

```
package com.demo.dao;
import java.sql.Connection;

public class StudentDAO {
    public boolean addStudentDetails(StudentVO vo) throws StudentDAOException {
        boolean flag = false;
        String userName = "root";
        String password = "password";
        String url = "jdbc:mysql://localhost:3306/project";
        Connection connection = null;
        PreparedStatement ps = null;
        try {
            connection = DriverManager.getConnection(url, userName, password);
            String sql = "INSERT INTO student (sname,phone,addres,"
                    + "dob,standard)VALUES(?,?,?,?,?)";
            ps = connection.prepareStatement(sql);
            ps.setString(1, vo.getSname());
            ps.setLong(2, vo.getPhone());
            ps.setString(3, vo.getAddres());
            ps.setDate(4, vo.getDob());
            ps.setInt(5, vo.getStandard());
            ps.executeUpdate();
            flag = true;
        } catch (SQLException e) {
            throw new StudentDAOException("Error When Adding Student",e);
        } finally {
            try {
                ps.close();
                connection.close();
            } catch (SQLException e) {
                throw new StudentDAOException("Error When Adding Student",e);
            }
        }
        return flag;
```

```
public StudentVO fetchStudentById(int sid) throws StudentDAOException {
    String userName = "root";
    String password = "password";
    String url = "jdbc:mysql://localhost:3306/project";
    Connection connection = null;
    PreparedStatement ps = null;
    ResultSet rs = null;
    StudentVO vo = null;
    try {
        connection = DriverManager.getConnection(url, userName, password);
        String query = "select * from student where sid=?";
        System.out.println("studentd id="+sid);
        ps = connection.prepareStatement(query);
        ps.setInt(1, sid);
        rs = ps.executeQuery();
        while (rs.next()) {
            vo=new StudentVO();
            vo.setSid(rs.getInt("sid"));
            vo.setSname(rs.getString("sname"));
            vo.setPhone(rs.getLong("phone"));
            vo.setAddres(rs.getString("addres"));
            vo.setDob(rs.getDate("dob"));
            vo.setStandard(rs.getInt("standard"));
        }
        if(vo==null)
        {
            throw new StudentDAOException("Given StudentId "
                    + "does Not Exist in DataBase");
        }

    } catch (SQLException e) {
        throw new StudentDAOException("Error in SQL",e);

    } finally {

        try {
            ps.close();
            connection.close();
        } catch (SQLException e) {

            throw new StudentDAOException("Database Error",e);
        }

    }
    return vo;
```

Refer the video for more details on creating the DAO

Try it Out- Develop the BO

Click to continue

```
package com.demo.bo;
import com.demo.dao.StudentDAO;

public class StudentBO {

    public boolean addStudent(StudentVO vo)
            throws StudentBOException {
        try {
            StudentDAO dao = new StudentDAO();
            boolean flag;
            flag = dao.addStudentDetails(vo);
            return flag;
        } catch (StudentDAOException e) {

            throw new StudentBOException("Error When"
                    + " Adding Student",e);
        }
    }
    public StudentVO fetchStudent(int sid)
            throws StudentBOException {

        try {
            StudentDAO dao = new StudentDAO();
            StudentVO vo = new StudentVO();
            vo = dao.fetchStudentById(sid);
            return vo;
        } catch (StudentDAOException e) {

            throw new StudentBOException("Error when "
                    + "fetching student",e);
        }
    }

}
```

Develop the **StudentBO** add two methods **addStudent** and **fetchStudent** details.

Package should be com.demo.bo

Import all the required libraries using organize imports - ctrl+shift+o

Create the instance of DAO and invoke the appropriate DAO methods from the respective BO methods.

Now we have completed the following integration,
StudentBO → StudentDAO

Refer the video for more details on creating the BO.

Try it Out- Develop the Service Layer

Click to continue

```
package com.demo.service;
import com.demo.bo.StudentBO;

public class StudentService {
    StudentBO bo = new StudentBO();
    StudentResponseObject obj = new StudentResponseObjec

    public String addStudent(StudentVO vo) throws Studen
        String msg;
        boolean flag;
        try {
            flag = bo.addStudent(vo);
            if (flag)
                msg = "Student Added Successfully";
            else
                msg =  "Error When Adding Student Details, "
                     + "Reach out Administrator";
        } catch (StudentBOException e) {
            msg= e.getMessage();
        }
        return msg;
    }
    public StudentVO fetchStudent(int sid)
            throws StudentBOException {
        StudentVO vo = null;
        try {
            vo = bo.fetchStudent(sid);
        } catch (StudentBOException e) {
            e.printStackTrace();
            // null object returned if exception thrown.
            vo=null;
        }
        return vo;
    }
}
```

Develop the **StudentService** add two methods **addStudent** and **fetchStudent** details.

Package should be com.demo.service

Import all the required libraries using organize imports - ctrl+shift+o

Create the instance of BO and invoke the appropriate BO methods from the respective Service methods.

Now we have completed the following integration,
StudentService→ StudentBO → StudentDAO

Refer the video for more details on creating the BO.

Try it Out- Develop the CUI

Click to continue

Develop the **StudentMain** which will be the **CUI** for user to provide inputs, add two methods **addStudent** and **fetchStudent** details. Also add a main method to invoke the other two methods based on scanner inputs.

Package should be com.demo.main

Import all the required libraries using organize imports - ctrl+shift+o

```
package com.demo.main;

import java.sql.Date;

public class StudentMain {

    public static void main(String[] args)
            throws StudentNotFoundException, StudentBOException {
        System.out.println("Please select one of the below options");
        System.out.println("1. Add students");
        System.out.println("2. Fetch Student by student id");
        System.out.println("3. Exit");
        Scanner s = new Scanner(System.in);
        int menuSelected = s.nextInt();
        switch (menuSelected) {
        case 1:
            addStudent();
            break;
        case 2:
            fetchStudent();
            break;
        default:
            System.exit(0);
        }
    }
```

Add Student Output

```
Please select one of the below options
1. Add students
2. Fetch Student by student id
3. Exit
1
Please Enter The Name:
john
Please Enter The PhoneNo:
9776654383
Please Enter The Address:
temple road
Please Enter The DOB (YYYY-MM-DD):
2010-12-03
Please Enter The Standard:
3
Student Added Successfully
```

CUI Add method

```
private static void addStudent() throws StudentBOException {
    StudentService student = new StudentService();
    Scanner s = new Scanner(System.in);
    System.out.println("Please Enter The Name:");
    String sname = s.nextLine();
    System.out.println("Please Enter The PhoneNo:");
    Long phone = Long.parseLong(s.nextLine());
    System.out.println("Please Enter The Address:");
    String address = s.nextLine();
    System.out.println("Please Enter The DOB (YYYY-MM-DD):");
    java.sql.Date dob = Date.valueOf(s.nextLine());
    System.out.println("Please Enter The Standard:");
    int standard = s.nextInt();


    StudentVO vo = new StudentVO();
    vo.setSname(sname);
    vo.setPhone(phone);
    vo.setAddres(address);
    vo.setDob(dob);
    vo.setStandard(standard);
    String obj;
    obj = student.addStudent(vo);
    System.out.println(obj);
}
```

Try it Out- Develop the CUI

Click to continue

CUI Fetch method

```
private static void fetchStudent()
        throws StudentBOException {

    Scanner s = new Scanner(System.in);
    System.out.println("Please Enter The StudentID To Be Fetched:");
    Integer sid = Integer.parseInt(s.nextLine());
    StudentService student = new StudentService();
    StudentVO vo;
    vo = student.fetchStudent(sid);
    if (vo != null) {
        System.out.println(
                "=============================================================================================");
        System.out.println("StudentId" + '\t' + "StudentName" + '\t' + "PhoneNumber" + '\t' + "Address" + '\t'
                + '\t' + "DOB" + '\t' + "Standard" );
        System.out.println(
                "=============================================================================================");

        System.out.println(vo.getSid() + "\t\t" + vo.getSname() + "\t\t " + vo.getPhone() + "\t\t " + vo.getAddres()
                + "\t" + vo.getDob() + "\t\t" + vo.getStandard() + "\t\t");

    }
}
```

Fetch Student Output

```
Please select one of the below options
1. Add students
2. Fetch Student by student id
3. Exit
2
Please Enter The StudentID To Be Fetched:
2
studentd id=2
=============================================================================================
StudentId       StudentName     PhoneNumber     Address         DOB             Standard
=============================================================================================
2               john            9776654383      temple road     2010-12-03              3
```