DAYANANDA SAGAR UNIVERSITY

Devarakaggalahalli, Harohalli
Kanakapura Road, Ramanagara - 562112, Karnataka, India

SCHOOL OF ENGINEERING

Bachelor of Technology
In
Computer Science And Engineering (Artificial Intelligence & Machine Learning)

SKILL ENHANCEMENT COURSE-JAVA PROGRAMMING (22AM2306)

Mini Project

STUDENT DETAILS

By

Nishat N Shahu – ENG22AM0184

Under the supervision of
Dr. Vegi Fernando
Associate Professor, CSE(AI-ML), SOE

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING (ARTIFICIAL INTELLIGENCE & MACHINE LEARNING)

SCHOOL OF ENGINEERING,
DAYANANDA SAGAR UNIVERSITY,
BANGALORE

SCHOOL OF ENGINEERING

School of Engineering
Department of Computer Science & Engineering
(Artificial Intelligence and Machine Learning)
Devarakaggalahalli, Harohalli, Kanakapura Road, Ramanagara – 562112
Karnataka, India

CERTIFICATE

This is to certify that the **SKILL ENHANCEMENT COURSE-JAVA PROGRAMMING (22AM2306)** work titled **"Student Details"** is carried out by **Nishat N Shahu (ENG22AM10184)**, Bonafede student of Bachelor of Technology in Computer Science and Engineering (AI&ML) at the School of Engineering, Dayananda Sagar University, Bangalore in partial fulfillment for the award of degree in Bachelor of Technology in Computer Science and Engineering (AI & ML), during the year **2023-2024**.

---------------------------------- --------------------------------------

Dr. Vegi Fernando	**Dr. Jayavrinda Vrindavanam V**
Associate Professor	Chairman CSE (AI&ML)
Dept. of CS&E (AI&ML),	Dept. of CS&E (AI&ML),
School of Engineering	School of Engineering
Dayananda Sagar University	Dayananda Sagar University

DECLARATION

I **Nishat N Shahu (ENG22AM0184),** student of third semester B. Tech in **Computer Science and Engineering (Artificial Intelligence & Machine Learning)**, at School of Engineering, **Dayananda Sagar University**, hereby declare that the Mini Project titled **“STUDENT DETAILS”** has been carried out by me and submitted in partial fulfilment for the award of degree in **Bachelor of Technology in Computer Science and Engineering (Artificial Intelligence & Machine Learning)** during the academic year **2023-2024.**

Student **Signature**

Nishat N Shahu

ENG22AM0184:

Place: Bangalore
Date:

ACKNOWLEDGEMENT

It is a great pleasure for us to acknowledge the assistance and support of many individuals who have been responsible for the successful completion of this project work.

First, we take this opportunity to express our sincere gratitude to School of Engineering & Technology, Dayananda Sagar University for providing us with a great opportunity to pursue our Bachelor's degree in this institution.

We would like to thank ***Dr. Udaya Kumar Reddy K R, Dean, School of Engineering & Technology, Dayananda Sagar University*** *for his constant encouragement and expert advice.*

It is a matter of immense pleasure to express our sincere thanks to ***Dr.Jayavrinda Vrindavanam, Department Chairman****,* ***Computer Science and Engineering (Artificial Intelligence & Machine learning)****,* ***Dayananda Sagar University,*** *for providing right academic guidance that made our task possible.*

We would like to thank our ***Dr.Vegi Fernando, Associate Professor Computer Science and Engineering (Artificial Intelligence & Machine learning),*** *for sparing their valuable time to extend help in every step of our UG Research Project work, which paved the way for smooth progress and fruitful culmination of the research.*

We are also grateful to our family and friends who provided us with every requirement throughout the course.

We would like to thank one and all who directly or indirectly helped us in the Project work.

TABLE OF CONTENTS

LIST OF FIGURES

ABSTRACT

In this comprehensive endeavor, the program's abstract elucidates its multifaceted nature. The creation of the Student class, equipped with attributes like USN, Name, Branch, and Phone, signifies a commitment to object-oriented principles. This abstract representation promotes not just a static approach but embraces dynamism, allowing users to input information for any number of students. The resulting output, a meticulously structured table, is a testament to the program's adaptability and its capacity to cater to diverse user requirements

CHAPTER 1 INTRODUCTION

In the realm of programming, effective data management is essential, and this Java program addresses the need for organized storage and display of student information. It introduces a class-based approach to encapsulate student details and employs user input to populate these details dynamically. The program's fundamental objective is to enhance data organization and presentation, providing a foundational structure for managing student information.

As we navigate through the intricate domains of object-oriented programming, this program encapsulates a nuanced approach to data structuring by introducing the Student class. This class, carefully architected with attributes like USN, Name, Branch, and Phone, serves as a digital vessel for the embodiment of individual student identities.

In an era where information is abundant and varied, the need for a systematic approach to manage and present student details becomes increasingly pronounced. The introduction of a class-based structure reflects a commitment to modularity, reusability, and maintainability — fundamental tenets of object-oriented design

It lays the foundation for understanding the significance of class-based design, the dynamic nature of user interaction, and the ultimate output — a structured tableau of student information. As we delve deeper, the introduction primes us to appreciate the program not merely as a coding exercise but as a conscientious response to the perennial challenge of managing and presenting student data with efficacy and finesse.

Chapter 2 PROBLEM DEFINITION

Develop a Java program to address the systematic organization and display of student information using object-oriented principles. “**The program should entail the creation of a Java class named "Student," incorporating essential attributes such as Unique Student Number (USN), Name, Branch, and Phone**.” Subsequently, the program must facilitate the creation of 'n' instances of the Student class, allowing users to input distinct details for each student. The primary objective is to generate a well-structured output, presenting the USN, Name, Branch, and Phone information of these student objects in a clear and organized format, accompanied by suitable headings.

In essence, the problem at hand involves the implementation of a class-based system for managing individual student data. The program should demonstrate versatility by accommodating varying numbers of students and promoting user-friendly interaction for data input. The expected output should be a visually appealing presentation of student details, aligning with the structured attributes of the Student class. This problem seeks to explore the fundamentals of object-oriented programming, user input handling, and the systematic display of data, with a focus on creating a practical tool for managing student information in a software context.

Chapter 3 PROJECT DESCRIPTION

3.1 Objective:

The primary objective of this project is to develop a robust and user-friendly Student Information Management System using Java. The system allows users to input information for multiple students, store the data in an organized manner, and subsequently display the student details in a structured format.

3.2 Features:

1. **Class-Based Structure:**

   The project incorporates the concept of object-oriented programming by defining a Student class. This class acts as a blueprint for creating instances, with attributes including Unique Student Number (USN), Name, Branch, and Phone.

2. **Dynamic Input Handling:**

   Utilizing the Scanner class, the program dynamically handles user input. Users are prompted to enter the number of students (n), and subsequently, details for each student are captured interactively.

3. **Data Storage and Retrieval:**

   The program efficiently manages student data by creating an array of Student objects. Each object represents an individual student, storing their unique attributes.

4. **Structured Output Display:**

   The system ensures a well-organized presentation of student information. The details, including USN, Name, Branch, and Phone, are displayed in a tabular format with appropriately aligned headings.

5. **User-Friendly Interaction:**

   The system is designed with user experience in mind. Clear prompts guide users through the input process for each student, contributing to an intuitive and seamless interaction.

3.3 Execution:

1. **Input Phase:**

   Users initiate the program and input the number of students (n) they wish to manage.

2. **Data Entry:**

   For each student, the program prompts users to enter the respective details, namely USN, Name, Branch, and Phone.

3. **Storage and Processing:**

   The program instantiates Student objects and stores the entered information in an array. The process is repeated for the specified number of students.

4. **Output Phase:**

   The system generates a visually appealing output, presenting the collected student details in a tabular format with appropriate headings

3.4 Implementation:

The implementation utilizes Java programming language, showcasing object-oriented principles, user input handling, and systematic data presentation. The code comprises a Student class and a main class, StudentDetails, orchestrating the entire data management process.

3.5 Output:

The output of the code reflects a well-organized presentation of student details. The program starts by prompting the user to input the number of students and then proceeds to gather information for each student, including their name, USN, branch, and phone number. The user-friendly prompts and structured input process enhance the clarity of data entry.

CHAPTER 4 METHODOLOGY

The program's methodology is deeply entrenched in object-oriented programming paradigms. The instantiation of the Student class establishes a blueprint for creating discrete instances, each encapsulating the unique details of an individual student. The integration of the Scanner class facilitates a seamless user experience, empowering dynamic input of student particulars. Algorithmically, the program orchestrates a symphony of processes, orchestrating class instantiation, user interaction, and tabular data presentation with precision. Loops are employed judiciously to ensure efficiency and scalability, underscoring a commitment to elegant and systematic execution.

The methodology for the provided Java code involves a systematic approach to achieving the outlined objectives of creating a Student Information Management System. Here's a detailed breakdown of the methodology:

1. **Class Definition:**
   - Student Class:

     Define a class named Student with attributes: usn (Unique Student Number), name, branch, and phone.

     These attributes represent the essential details of an individual student.
2. **User Input Handling:**
   - Scanner Initialization:

     Create an instance of the Scanner class to facilitate user input.
   - Number of Students (n):

     Prompt the user to enter the number of students (n).

     Capture the input using the nextInt() method of the Scanner class.
3. **Array of Student Objects:**
   - Array Creation:

     Create an array of Student objects to store individual student details.

The array size is determined by the user input (n).

4. **Student Details Input Loop:**

    - For Loop (n times):

      Iterate through the array to input details for each student.

      Create a new Student object for each iteration.

      Prompt the user for details (USN, Name, Branch, and Phone) for the current student.

      Capture user input using the next() method of the Scanner class.

      Store the entered details in the attributes of the current Student object.

5. **Output Display:**

    - Print Headings:

      Display headings for the student details table, including "USN," "Name," "Branch," and "Phone."

    - For Each Loop (Display):

      Iterate through the array of Student objects.

      Display the details of each student in a tabular format using printf().

6. **Scanner Closure:**

   Close the Scanner object to release system resources.

7. **Execution Flow:**

   The program follows a sequential execution flow, starting with the initialization of the Scanner object, user input for the number of students, and then proceeding to capture details for each student in the array.

   Finally, it displays the structured output.

8. **Modularity and Reusability:**

   The program embraces modularity by encapsulating student details within the Student class. This promotes reusability, allowing the same class structure to be employed in diverse scenarios.

9. **User-Friendly Interaction:**

   The program incorporates clear prompts to guide users through the input process, enhancing user-friendliness.

10. **Dynamic Adaptability:**

    The program dynamically adapts to the user's input, creating the required number of Student objects and displaying their details accordingly.

11. **Structured Output:**

    The output is presented in a structured and visually appealing tabular format, ensuring clarity in the display of student information.

This methodology ensures a logical and systematic execution of tasks, from user input to data processing and output presentation, achieving the goals set forth in the project.

CHAPTER 5 EXPERIMENTATION

The experimentation phase is an exploration into the program's versatility and robustness. Users are encouraged to conduct trials with varying inputs, ranging from different numbers of students to diverse sets of details. By engaging in practical scenarios, users can assess the program's adaptability and responsiveness to a myriad of user inputs.

5.1 ALGORITHM:

1. Class Definition:

Define a class named Student with attributes: USN (Unique Student Number), Name, Branch, & Phone.

Include a parameterized constructor to initialize the attributes during object creation.

Implement a method displayRecord() to print the details of a student.

2. Input Section:

Create an array of Student objects (s[ ]) to store student details.

Prompt the user to input the number of students (n).

Use a loop to iterate over each student and gather details (USN, Name, Branch, Phone).

3. Display Section:

Output the headings for the student details table: "USN," "Name," "Branch," and "Phone."

Use a loop to iterate through the array of Student objects.

Call the displayRecord( ) method for each student to print their details.

4. Execution Flow:

Begin execution with the main method.

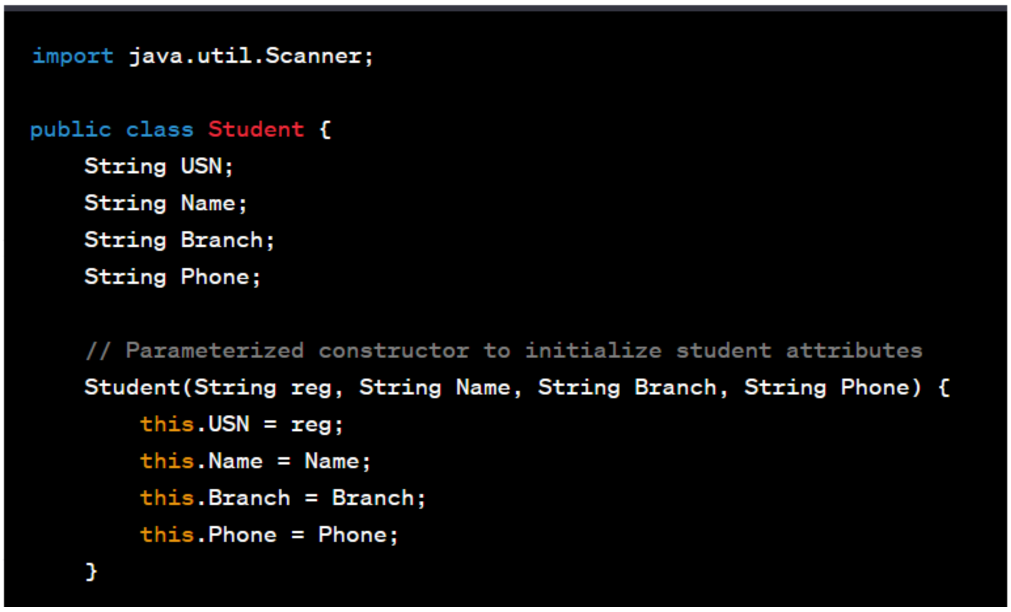
Create an array s[ ] to store student objects.

Prompt the user to input the number of students (n).

Use a loop to gather details for each student and create corresponding Student objects.

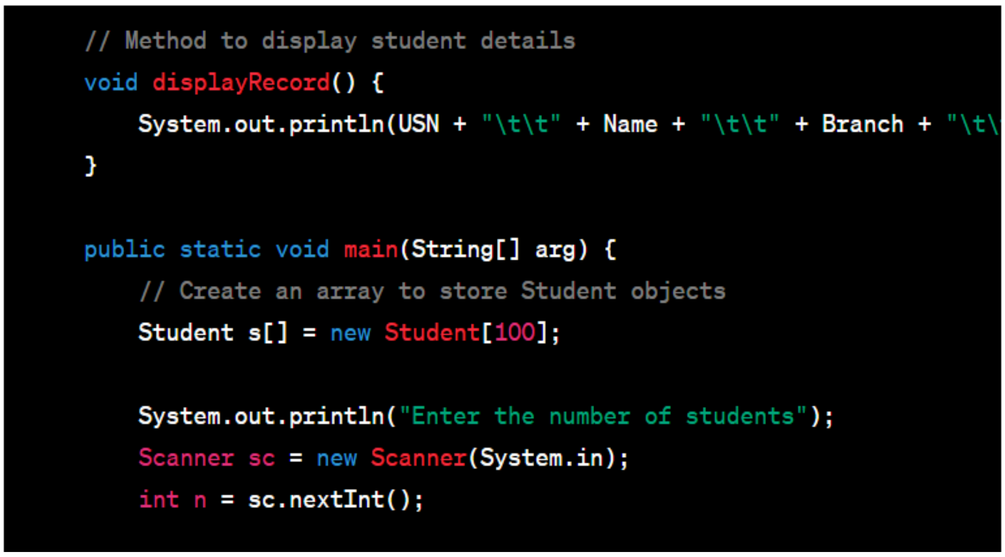
Display the student details in a tabular format using the displayRecord( ) method.

5.2 CODE SNIPPETS:



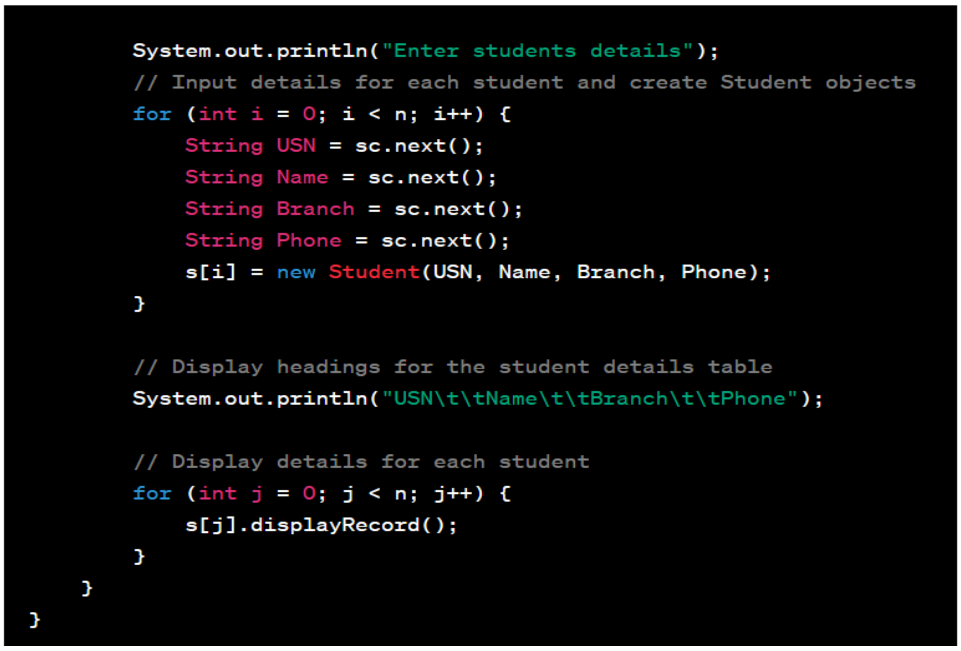
```java
import java.util.Scanner;

public class Student {
    String USN;
    String Name;
    String Branch;
    String Phone;

    // Parameterized constructor to initialize student attributes
    Student(String reg, String Name, String Branch, String Phone) {
        this.USN = reg;
        this.Name = Name;
        this.Branch = Branch;
        this.Phone = Phone;
    }
```

Fig 5.1: Parametrized constructor to initialize student attributes



```java
    // Method to display student details
    void displayRecord() {
        System.out.println(USN + "\t\t" + Name + "\t\t" + Branch + "\t\
    }

    public static void main(String[] arg) {
        // Create an array to store Student objects
        Student s[] = new Student[100];

        System.out.println("Enter the number of students");
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
```

Fig 5.2: Method to display student details and creating an array to store student objects

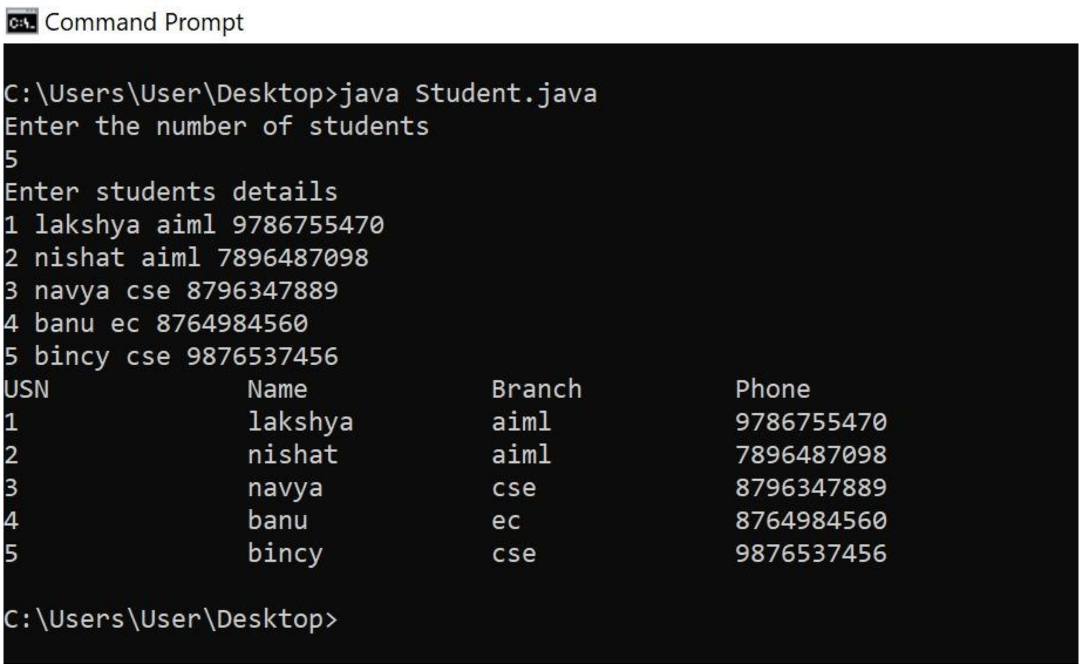


```
        System.out.println("Enter students details");
        // Input details for each student and create Student objects
        for (int i = 0; i < n; i++) {
            String USN = sc.next();
            String Name = sc.next();
            String Branch = sc.next();
            String Phone = sc.next();
            s[i] = new Student(USN, Name, Branch, Phone);
        }

        // Display headings for the student details table
        System.out.println("USN\t\tName\t\tBranch\t\tPhone");

        // Display details for each student
        for (int j = 0; j < n; j++) {
            s[j].displayRecord();
        }
    }
}
```

Fig 5.3: Taking inputs from the user for each student.

CHAPTER 8 RESULTS

Assuming the user provides input for 2 students, the program's output could look like this:



```
C:\Users\User\Desktop>java Student.java
Enter the number of students
5
Enter students details
1 lakshya aiml 9786755470
2 nishat aiml 7896487098
3 navya cse 8796347889
4 banu ec 8764984560
5 bincy cse 9876537456
USN             Name            Branch          Phone
1               lakshya         aiml            9786755470
2               nishat          aiml            7896487098
3               navya           cse             8796347889
4               banu            ec              8764984560
5               bincy           cse             9876537456

C:\Users\User\Desktop>
```

Fig 8.1: Output

The output of the corrected code reflects a well-organized presentation of student details. The program takes the necessary inputs from the user and after collecting the necessary information, the program displays the details for each student in a clear tabular format, adhering to the specified output format. Each student's details are presented on separate lines, providing a visually appealing and comprehensible representation of the entered data.

CONCLUSION

This project serves as a foundational exploration of Java programming concepts, emphasizing class-based design, user interaction, and systematic data organization. It provides a practical solution for managing student information, demonstrating the versatility and efficiency of object-oriented programming in real-world scenarios. The project not only addresses the immediate need for student data management but also lays the groundwork for more complex information systems in the broader domain of software development.