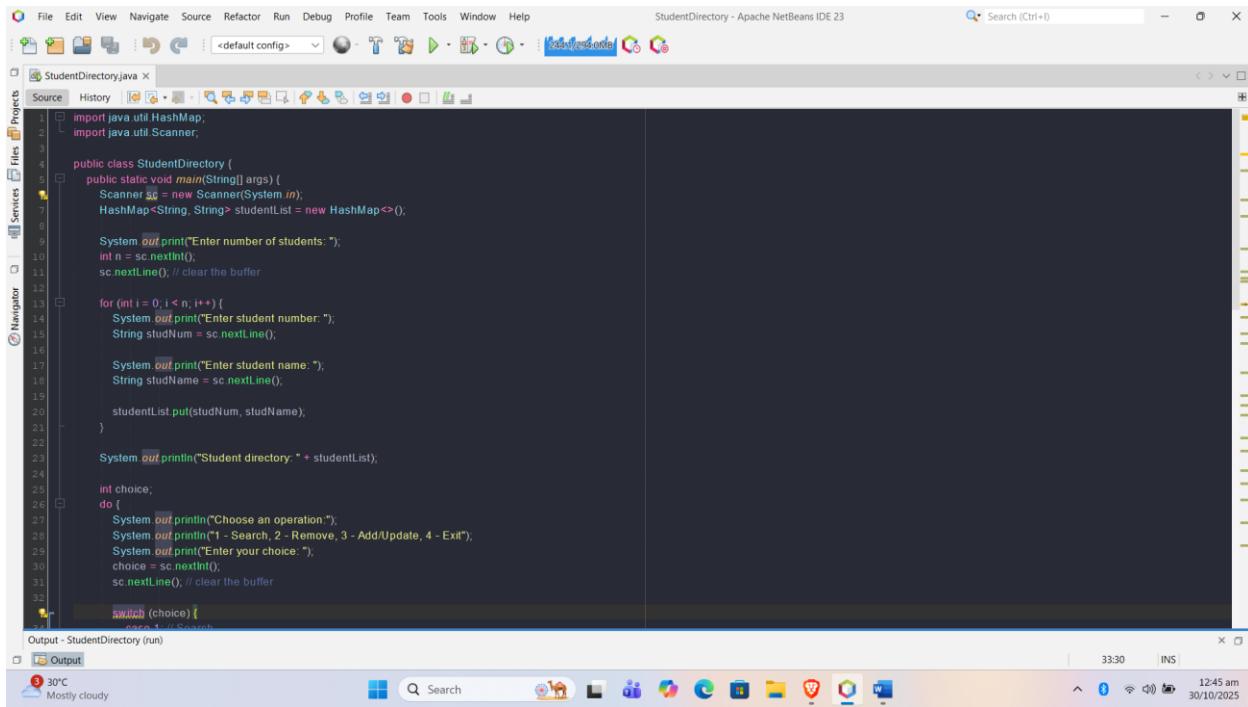


Code;



The screenshot shows the Apache NetBeans IDE 23 interface with the file `StudentDirectory.java` open. The code implements a student directory using a `HashMap`. It first prompts the user for the number of students and then for each student's number and name, storing them in the map. It then enters a loop where it prints a menu and performs operations based on user input (Search, Remove, Add/Update). The code uses `Scanner` to read input and `System.out.println` to output messages.

```
import java.util.HashMap;
import java.util.Scanner;

public class StudentDirectory {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        HashMap<String, String> studentList = new HashMap<>();

        System.out.print("Enter number of students: ");
        int n = sc.nextInt();
        sc.nextLine(); // clear the buffer

        for (int i = 0; i < n; i++) {
            System.out.print("Enter student number: ");
            String studNum = sc.nextLine();

            System.out.print("Enter student name: ");
            String studName = sc.nextLine();

            studentList.put(studNum, studName);
        }

        System.out.println("Student directory: " + studentList);

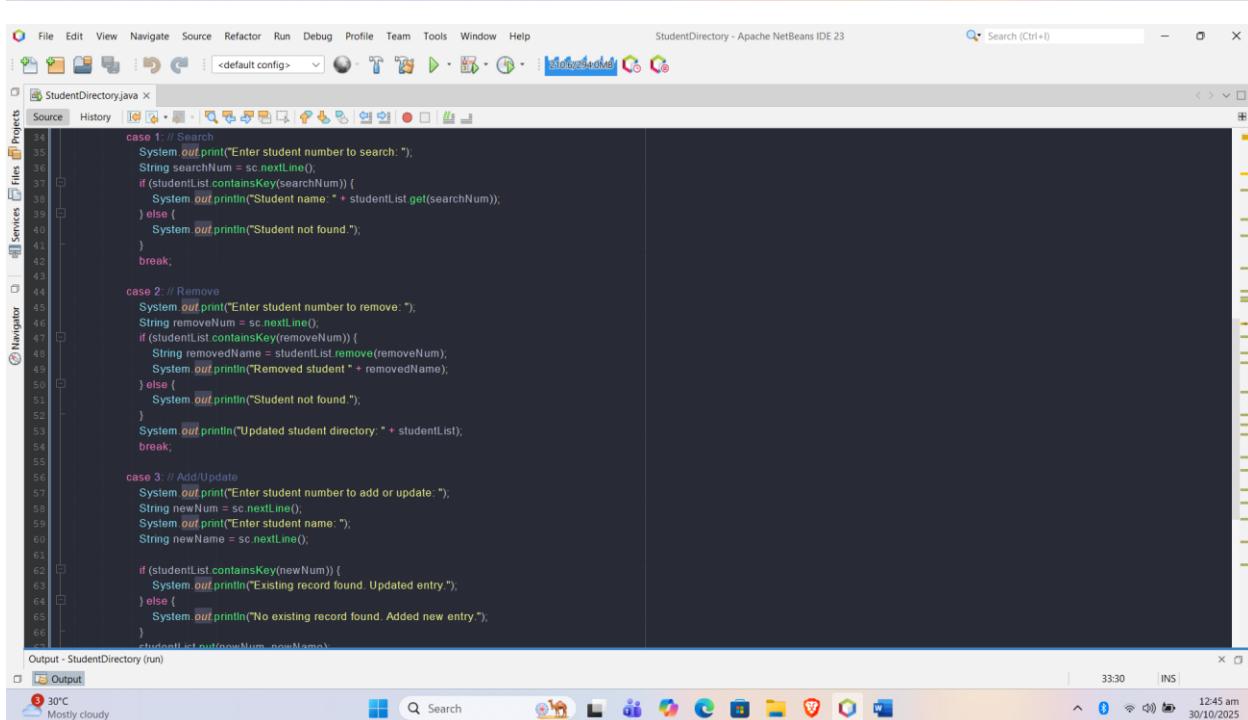
        int choice;
        do {
            System.out.println("Choose an operation:");
            System.out.println("1 - Search, 2 - Remove, 3 - Add/Update, 4 - Exit");
            System.out.print("Enter your choice: ");
            choice = sc.nextInt();
            sc.nextLine(); // clear the buffer

            switch (choice) {
                case 1: // Search
                    System.out.print("Enter student number to search: ");
                    String searchNum = sc.nextLine();
                    if (studentList.containsKey(searchNum)) {
                        System.out.println("Student name: " + studentList.get(searchNum));
                    } else {
                        System.out.println("Student not found.");
                    }
                    break;

                case 2: // Remove
                    System.out.print("Enter student number to remove: ");
                    String removeNum = sc.nextLine();
                    if (studentList.containsKey(removeNum)) {
                        String removedName = studentList.remove(removeNum);
                        System.out.println("Removed student " + removedName);
                    } else {
                        System.out.println("Student not found.");
                    }
                    System.out.println("Updated student directory: " + studentList);
                    break;

                case 3: // Add/Update
                    System.out.print("Enter student number to add or update: ");
                    String newNum = sc.nextLine();
                    System.out.print("Enter student name: ");
                    String newName = sc.nextLine();

                    if (studentList.containsKey(newNum)) {
                        System.out.println("Existing record found. Updated entry.");
                    } else {
                        System.out.println("No existing record found. Added new entry.");
                    }
                    studentList.put(newNum, newName);
            }
        } while (choice != 4);
    }
}
```



The second screenshot shows the same IDE interface with the same file open, but with additional code added to handle the `case 3` (Add/Update) more robustly. It checks if the new student number already exists in the map. If it does, it prints a message indicating an existing record was updated. If it doesn't, it prints a message indicating a new entry was added. This ensures that the program can handle both cases of adding a new student and updating an existing one correctly.

```
import java.util.HashMap;
import java.util.Scanner;

public class StudentDirectory {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        HashMap<String, String> studentList = new HashMap<>();

        System.out.print("Enter number of students: ");
        int n = sc.nextInt();
        sc.nextLine(); // clear the buffer

        for (int i = 0; i < n; i++) {
            System.out.print("Enter student number: ");
            String studNum = sc.nextLine();

            System.out.print("Enter student name: ");
            String studName = sc.nextLine();

            studentList.put(studNum, studName);
        }

        System.out.println("Student directory: " + studentList);

        int choice;
        do {
            System.out.println("Choose an operation:");
            System.out.println("1 - Search, 2 - Remove, 3 - Add/Update, 4 - Exit");
            System.out.print("Enter your choice: ");
            choice = sc.nextInt();
            sc.nextLine(); // clear the buffer

            switch (choice) {
                case 1: // Search
                    System.out.print("Enter student number to search: ");
                    String searchNum = sc.nextLine();
                    if (studentList.containsKey(searchNum)) {
                        System.out.println("Student name: " + studentList.get(searchNum));
                    } else {
                        System.out.println("Student not found.");
                    }
                    break;

                case 2: // Remove
                    System.out.print("Enter student number to remove: ");
                    String removeNum = sc.nextLine();
                    if (studentList.containsKey(removeNum)) {
                        String removedName = studentList.remove(removeNum);
                        System.out.println("Removed student " + removedName);
                    } else {
                        System.out.println("Student not found.");
                    }
                    System.out.println("Updated student directory: " + studentList);
                    break;

                case 3: // Add/Update
                    System.out.print("Enter student number to add or update: ");
                    String newNum = sc.nextLine();
                    System.out.print("Enter student name: ");
                    String newName = sc.nextLine();

                    if (studentList.containsKey(newNum)) {
                        System.out.println("Existing record found. Updated entry.");
                    } else {
                        System.out.println("No existing record found. Added new entry.");
                    }
                    studentList.put(newNum, newName);
            }
        } while (choice != 4);
    }
}
```

The screenshot shows the Apache NetBeans IDE 23 interface. The main window displays the Java code for `StudentDirectory.java`. The code implements a menu-based application for managing student records. It includes methods for adding, updating, and removing students from a list. The IDE's toolbar, project tree, and various toolbars are visible around the code editor.

```
case 3: // Add/Update
    System.out.print("Enter student number to add or update: ");
    String newNum = sc.nextLine();
    System.out.print("Enter student name: ");
    String newName = sc.nextLine();

    if (studentList.containsKey(newNum)) {
        System.out.println("Existing record found. Updated entry.");
    } else {
        System.out.println("No existing record found. Added new entry.");
    }
    studentList.put(newNum, newName);
    System.out.println("Updated student directory: " + studentList);
    break;

case 4: // Exit
    System.out.println("Exiting program.");
    break;

default:
    System.out.println("Invalid choice. Please try again.");
}
} while (choice != 4);

sc.close();
}
```

Output

The screenshot shows the Windows taskbar at the bottom of the screen. It displays the title bar for the running Java application, "StudentDirectory (run)", along with other open application icons like File Explorer, Edge, and FileZilla. The system tray shows the date and time as 30/10/2025, 12:48 am.

Output:

```
run:
Enter number of students: 3
Enter student number: 02000345888
Enter student name: Justin Adsuara
Enter student number: 2000344888
Enter student name: Lebron James
Enter student number: 2000334888
Enter student name: Cristiano Ronaldo
Student directory: {02000345888=Justin Adsuara, 2000344888=Cristiano Ronaldo, 2000334888=Lebron James}
Choose an operation:
1 - Search, 2 - Remove, 3 - Add/Update, 4 - Exit
Enter your choice:
2
Enter student number to remove: 2000344888
Removed student Lebron James
Updated student directory: {02000345888=Justin Adsuara, 2000334888=Cristiano Ronaldo}
Choose an operation:
1 - Search, 2 - Remove, 3 - Add/Update, 4 - Exit
Enter your choice:
```