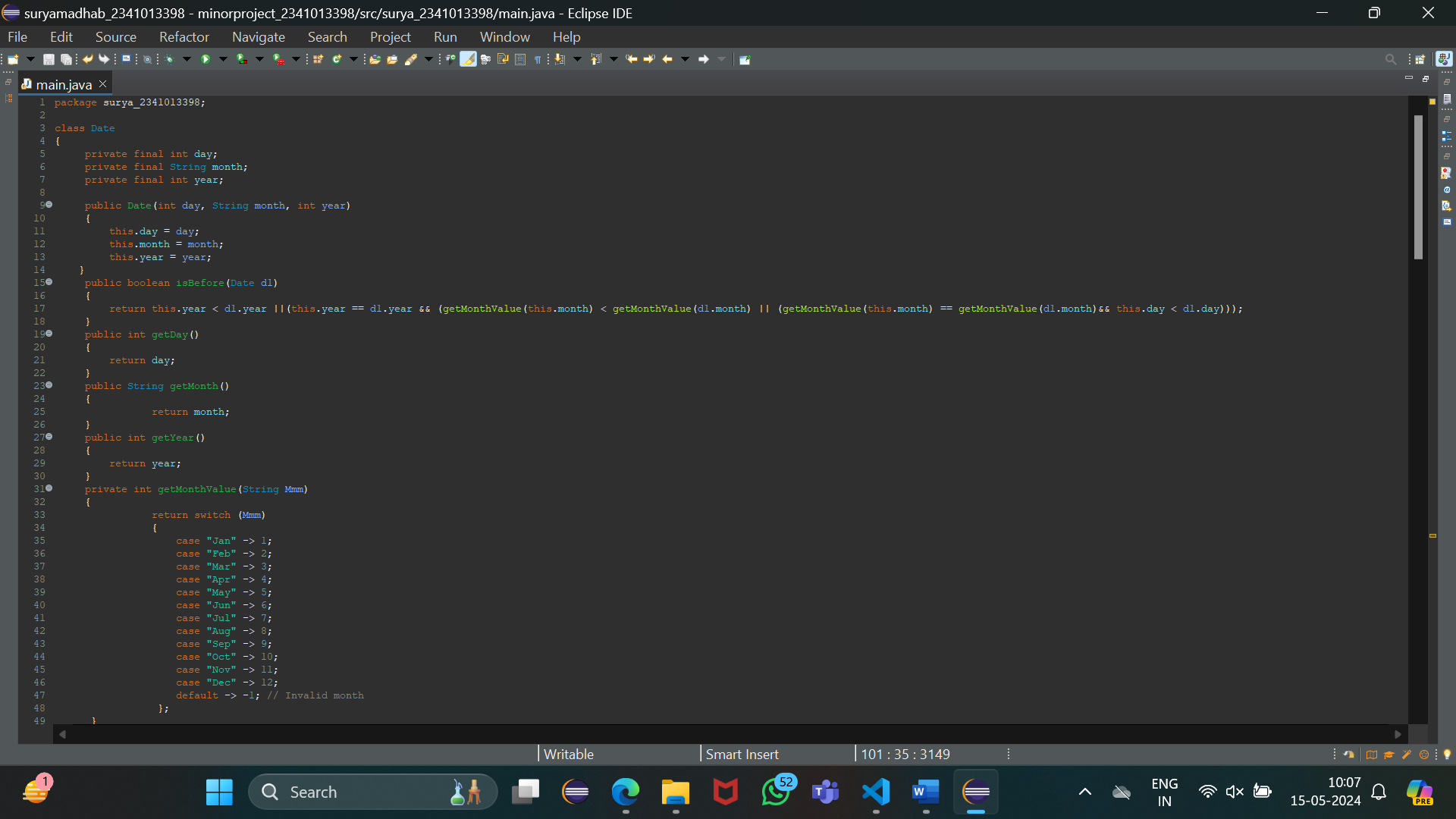
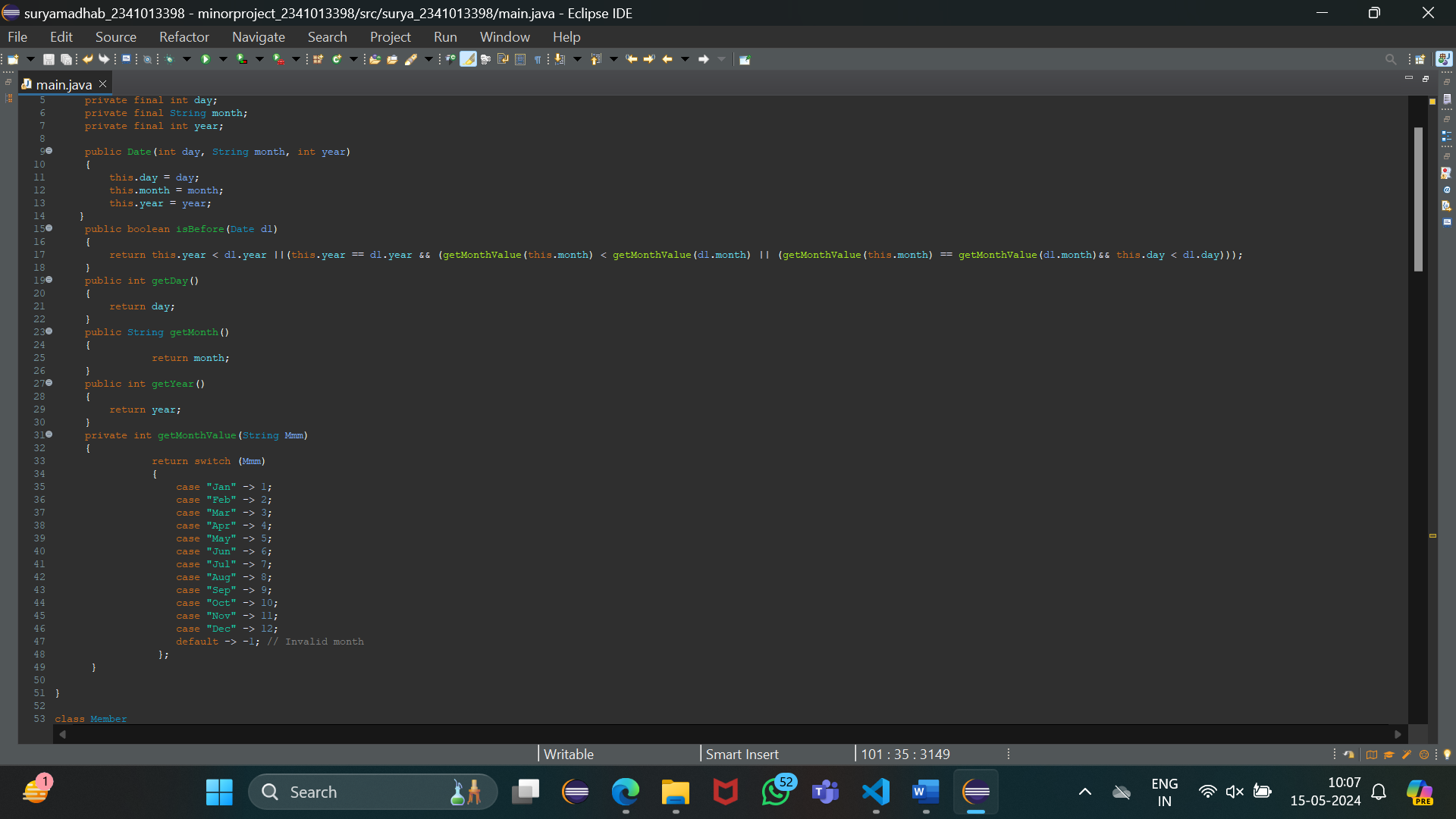
**DATA STRUCTURES AND ALGORITHM** (CSE 2001)

Minor Project – 1, 2024

Objective: To create a member database and retrieve records from it.

class **Date**:-

**Class Declaration:** Defines a class named Date within the package suryamadhab\_2341013398.

**Instance Variables:**

day: Stores the day component of the date.

month: Stores the month component of the date.

year: Stores the year component of the date.

**Constructor:** Initializes the day, month, and year variables when a Date object is created.

**isBefore Method**:

Compares two Date objects to determine if the current Date object is before the provided Date object.

Compares years first, then months, and finally days using nested conditions.

Returns true if the current Date object is before the provided Date object; otherwise, returns false.

**Getter Methods:**

**getDay():** Returns the day component of the date.

**getMonth():** Returns the month component of the date.

**getYear():** Returns the year component of the date.

Private Method - **getMonthValue:**

Converts the three-letter month abbreviation to its corresponding numeric value.

Uses a switch-case statement to map month abbreviations to numeric values.

Returns -1 for invalid month abbreviations.

**Summary:**

The provided code defines a Date class with instance variables representing the day, month, and year components of a date. It includes a constructor to initialize these variables and methods to compare dates and retrieve individual components. The isBefore method determines if one date comes before another, considering year, month, and day values. Additionally, there's a private method getMonthValue to convert month abbreviations to numeric values. Overall, this class facilitates date manipulation and comparison operations.

class **Member:-**

**Class Definition:**

The code defines a class named Member in the package minor\_project\_2341004117.

**Attributes:**

The class has several private attributes including **memberId, lastName, firstName, handicap, gender, team, memberType, coach, phone, and joinDate.**

These attributes represent different characteristics of a member in a sports club.

**Constructors:**

There are multiple constructors defined for the Member class, allowing for different combinations of member attributes to be initialized.

Constructors facilitate creating instances of Member with various sets of parameters.

**Methods:**

**getHandicapScore():** Returns the handicap score of the member.

**getGender():** Returns the gender of the member.

**getTeam():** Returns the team of the member.

**getMemberType():** Returns the type of the member.

**getJoinDate():** Returns the join date of the member.

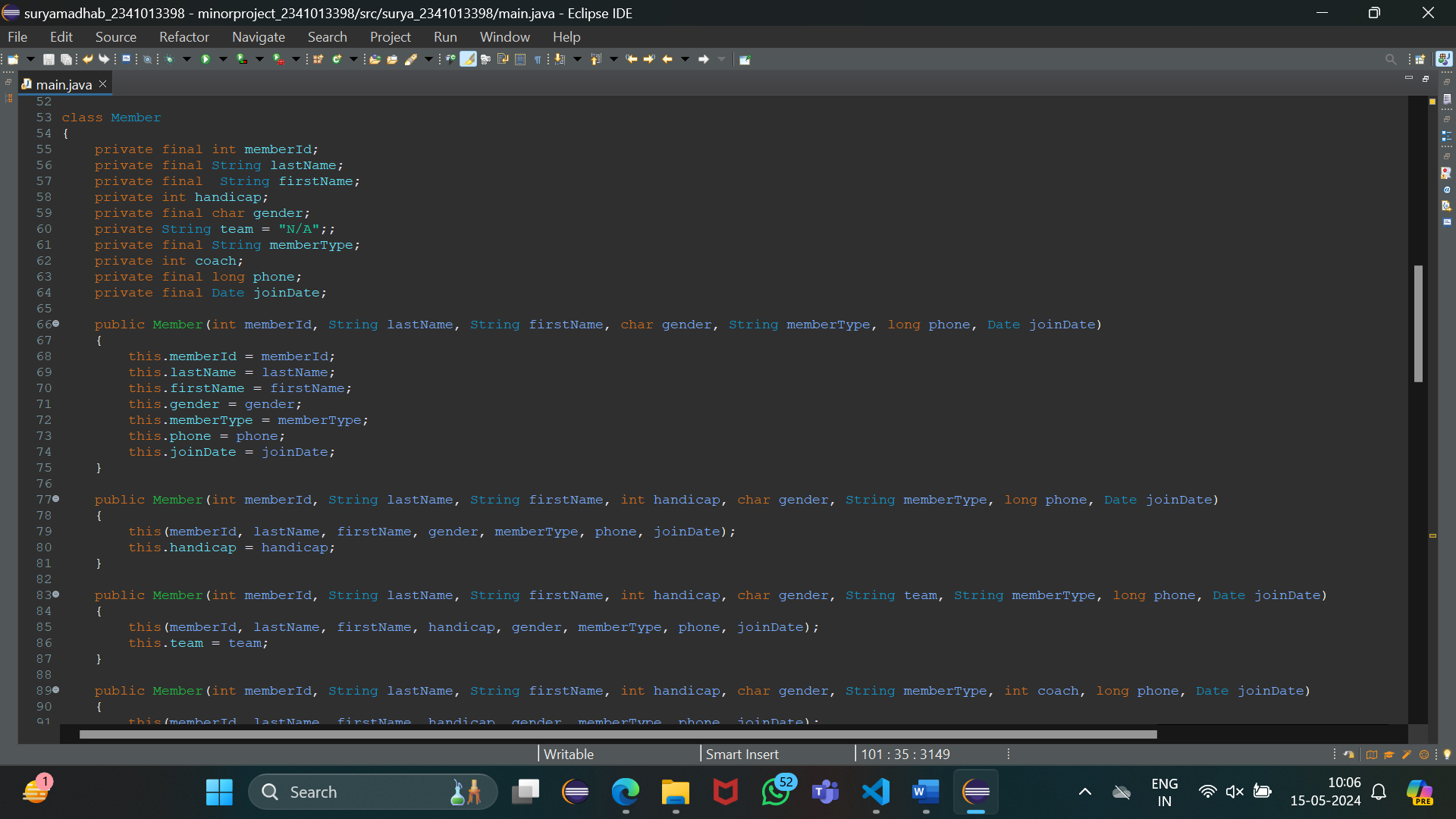
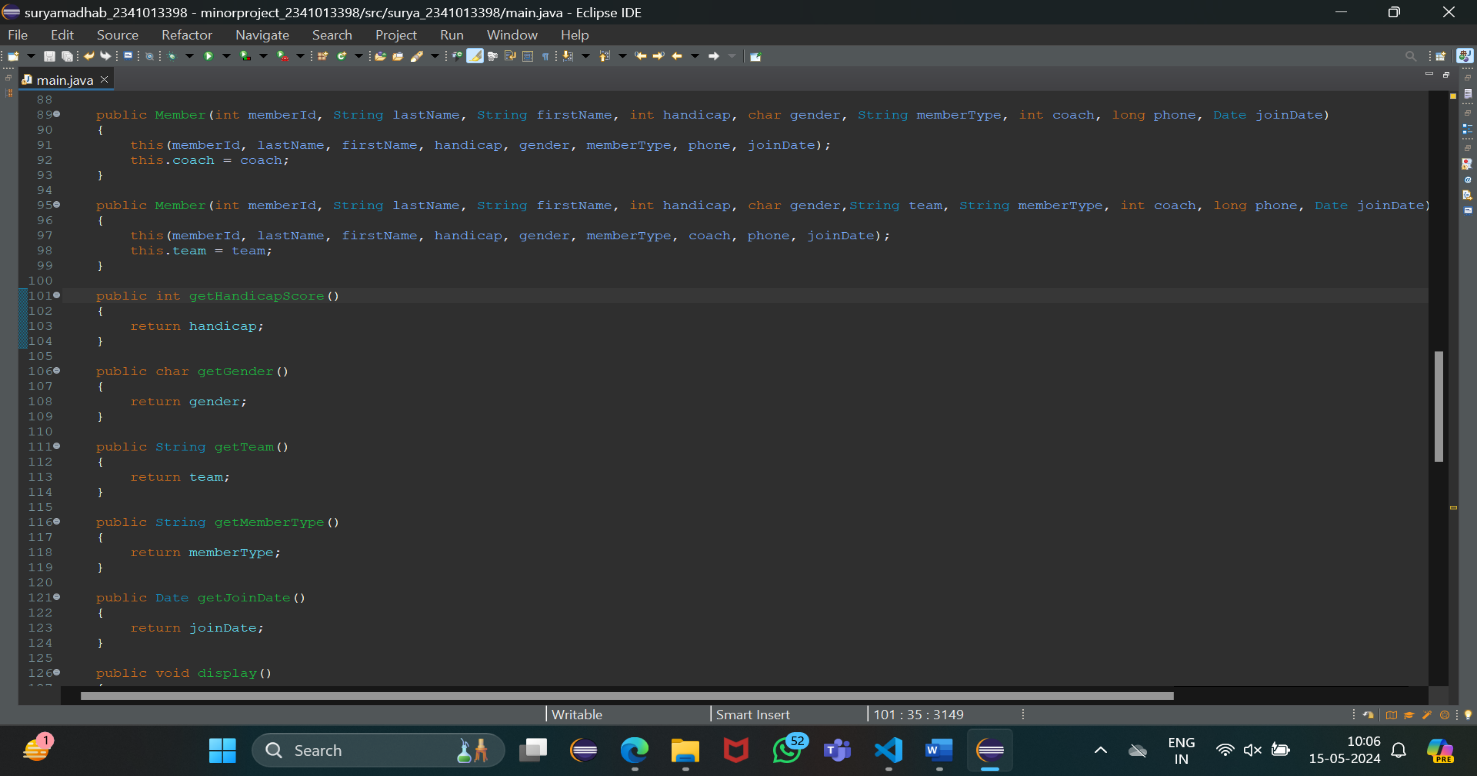
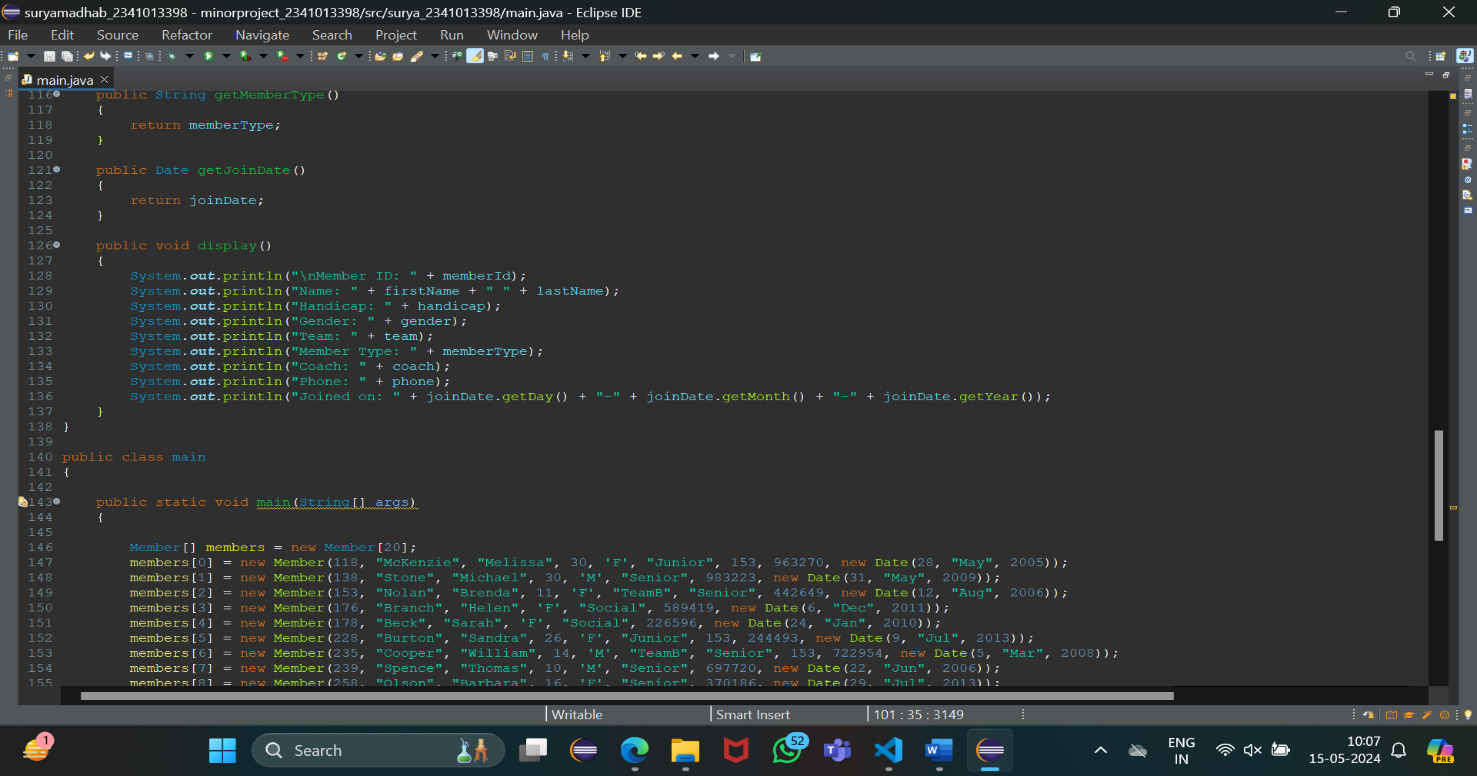
**display():** Displays the details of the member including ID, name, handicap, gender, team, member type, coach, phone, and join date.

**Summary:**

The provided Java code defines a class named Member which represents a member in a sports club. It encapsulates various attributes such as **ID, name, handicap, gender, team, member type, coach, phone, and join date**. The class

provides multiple constructors to initialize member objects with different sets of attributes. Additionally, it offers methods to retrieve specific attributes and a display() method to print the

member details to the console. Overall, the Member class provides a flexible and organized way to manage information about club members.



class **main:- (Main method)**

**Array Initialization:**

An array members of type Member with a capacity of 20 is initialized.

Objects of Member class are created and assigned to array elements.

Display Members with Join Date Earlier Than a Specific Date:

Define a required date (07-Apr-09).

Iterate over each member in the array.

Check if the member's join date is before the required date.

If true, display the member's details.

Display Senior Members with Handicap Score Less Than 12:

Iterate over each member in the array.

Check if the member's type is "Senior" and their handicap score is less than 12 and greater than 0.

If true, display the member's details.

Display Female Senior Members in Team B:

Iterate over each member in the array.

Check if the member's type is "Senior", gender is female, and they are part of "TeamB".

If true, display the member's details.

**Summary:**

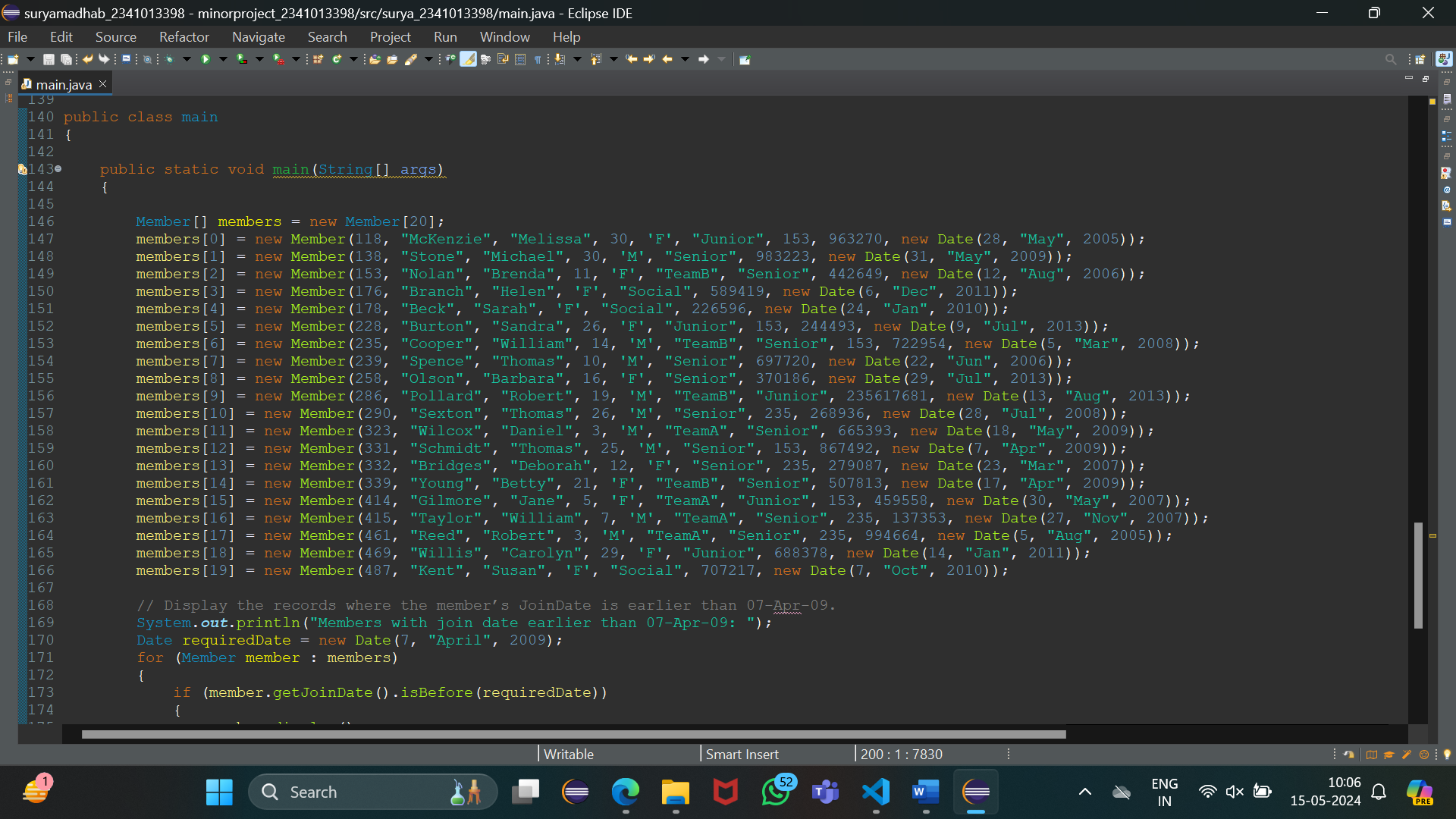
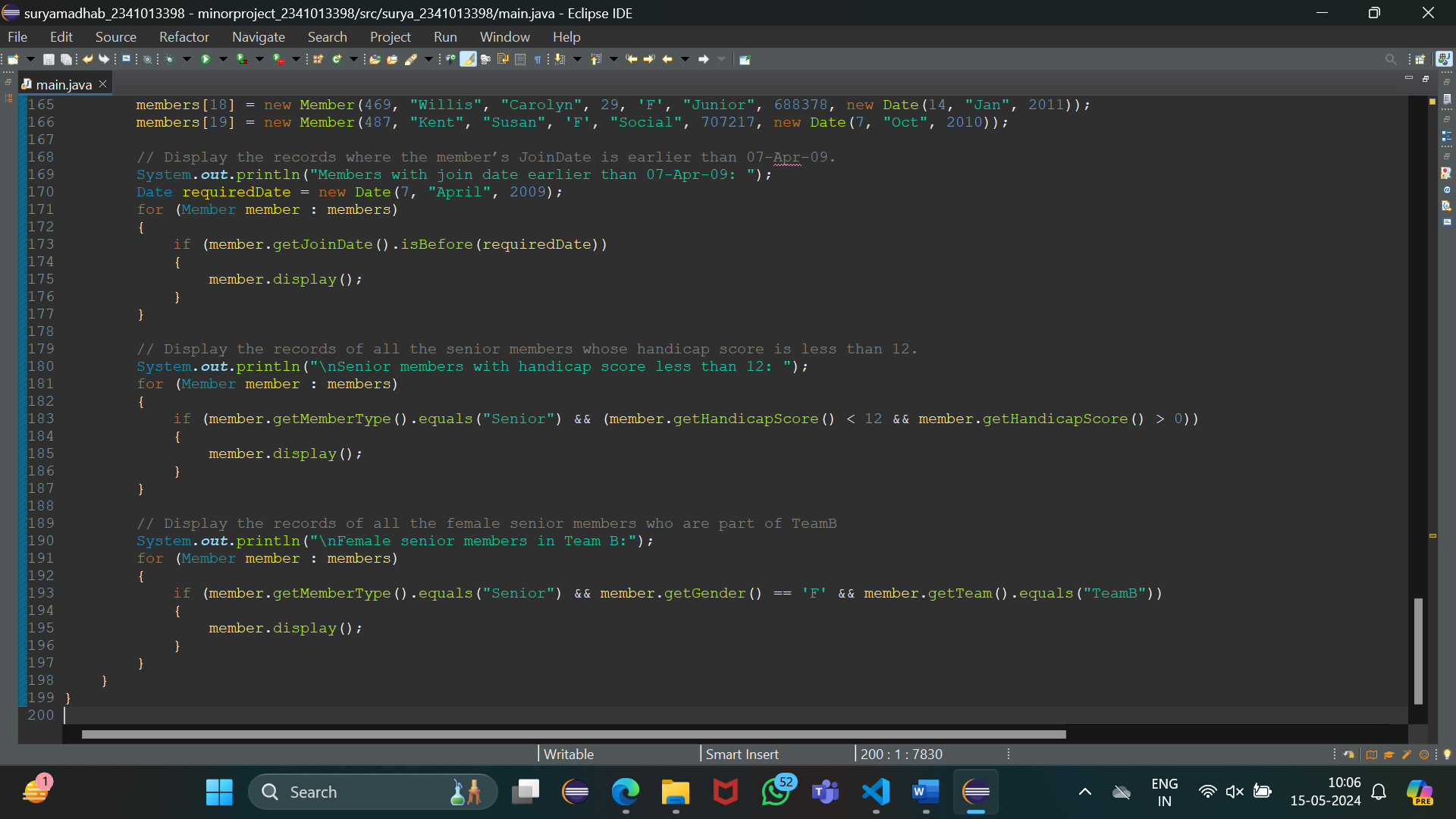
The code represents a Java program that manages a list of members. It initializes an array with member objects, each containing specific information such as ID, name, gender, member type, handicap score, and join date.

The program then performs three tasks:

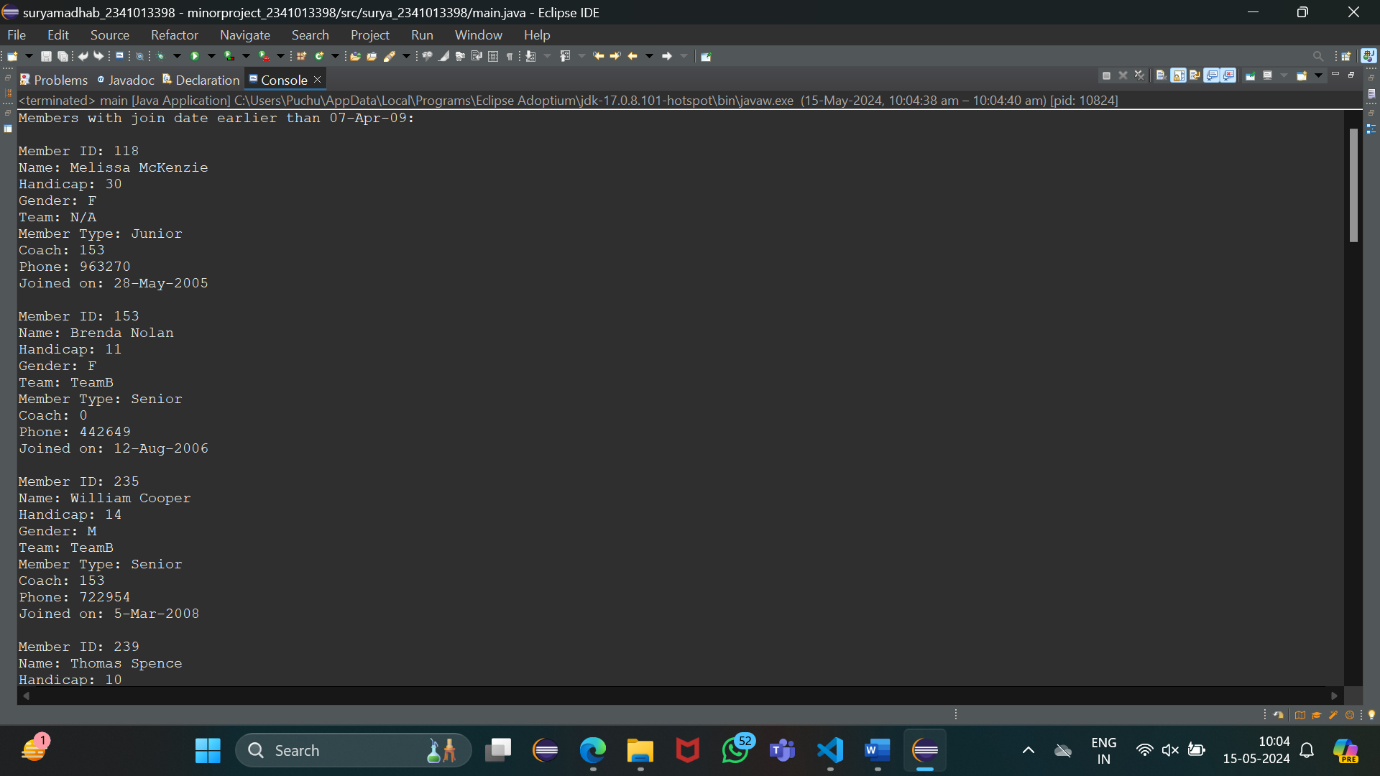
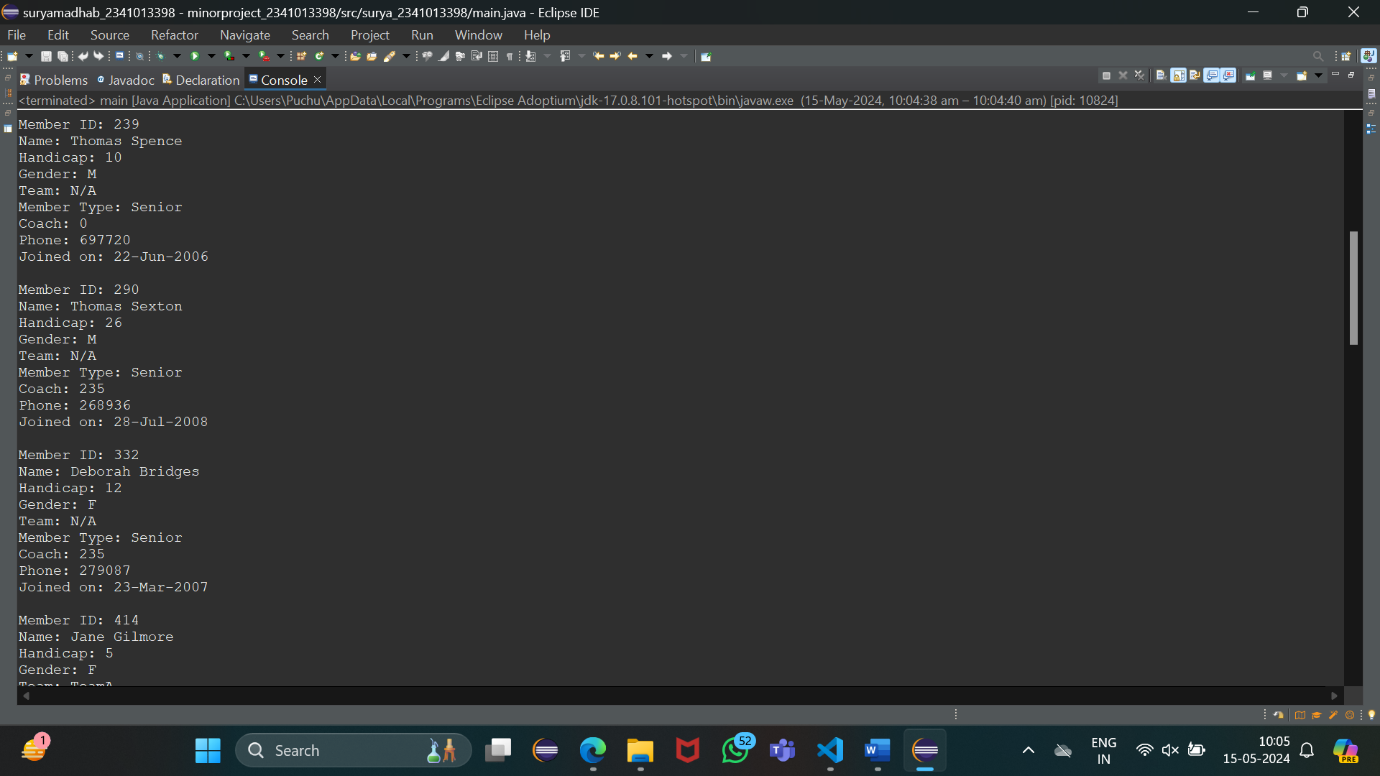
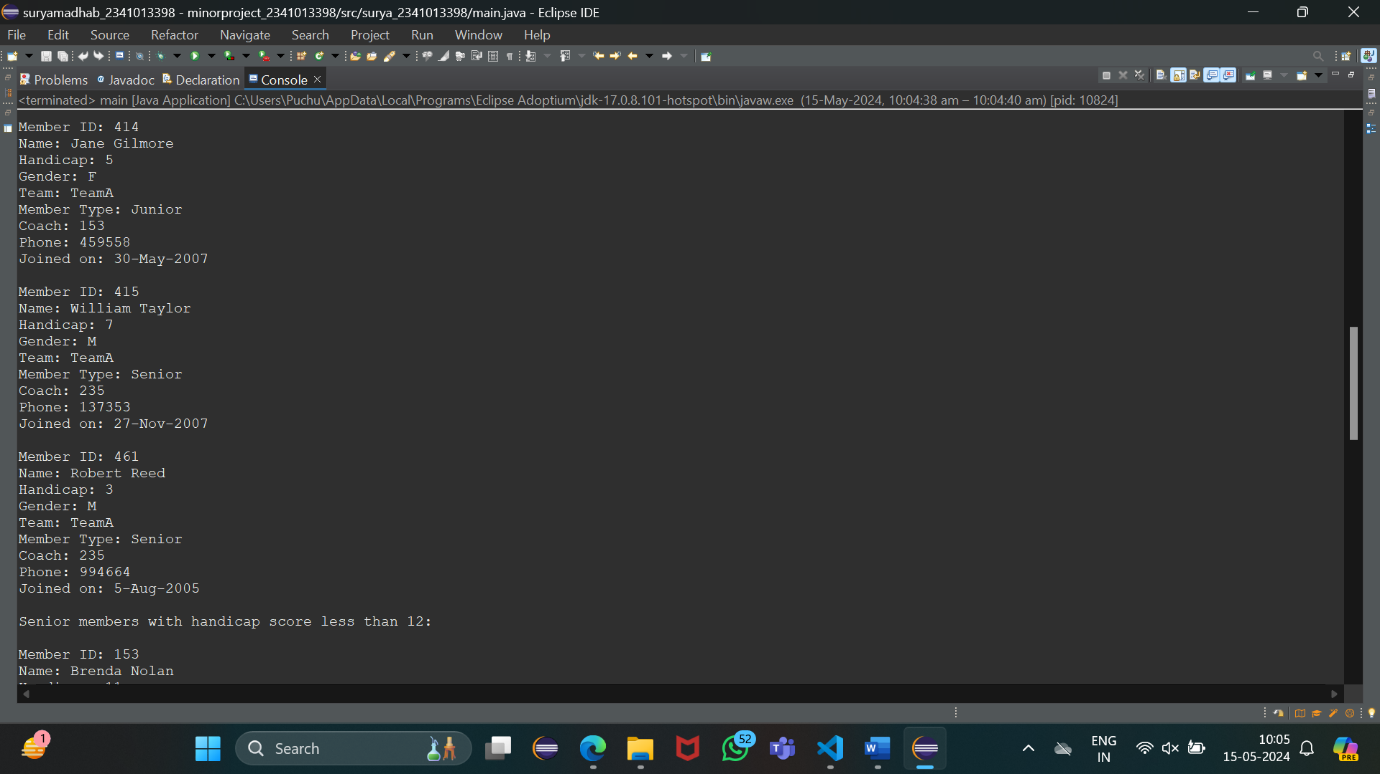
It displays members who joined before a certain date (07-Apr-09).

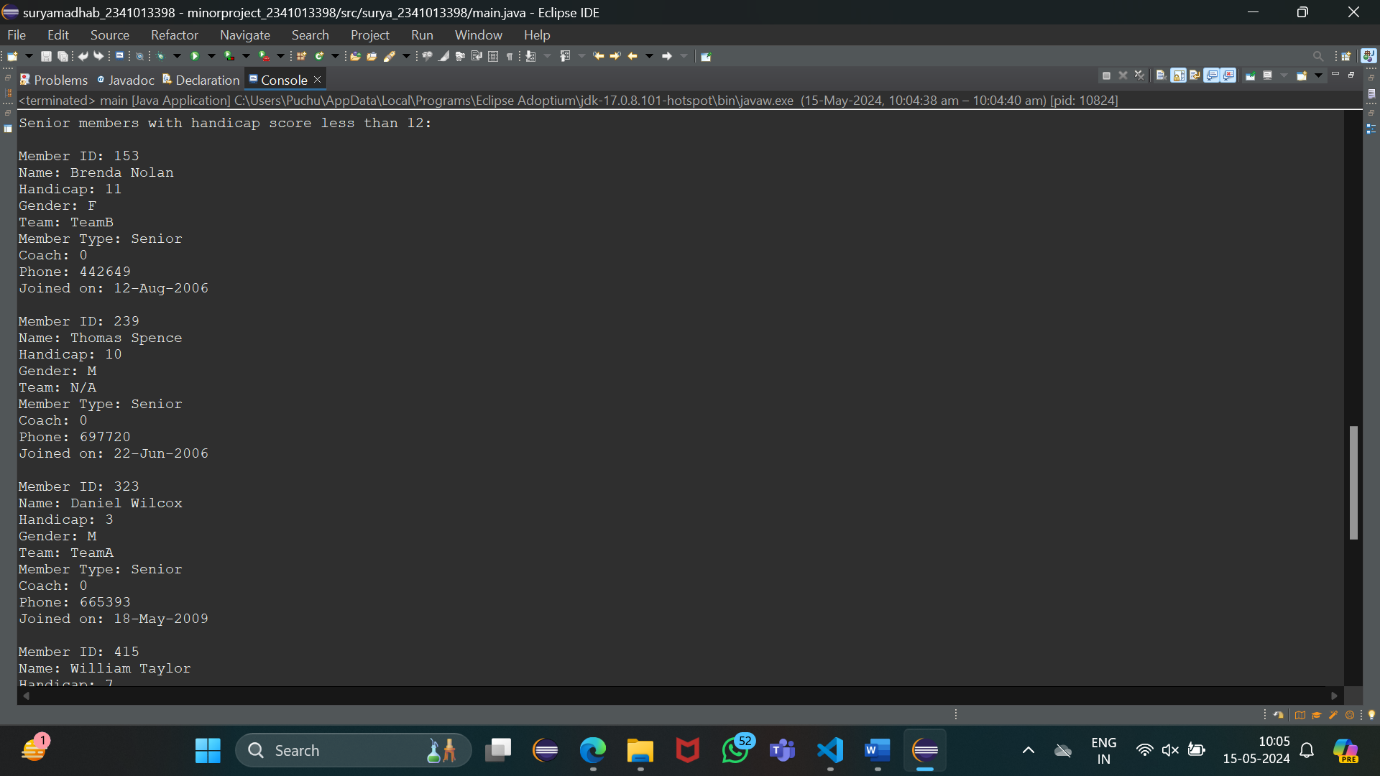
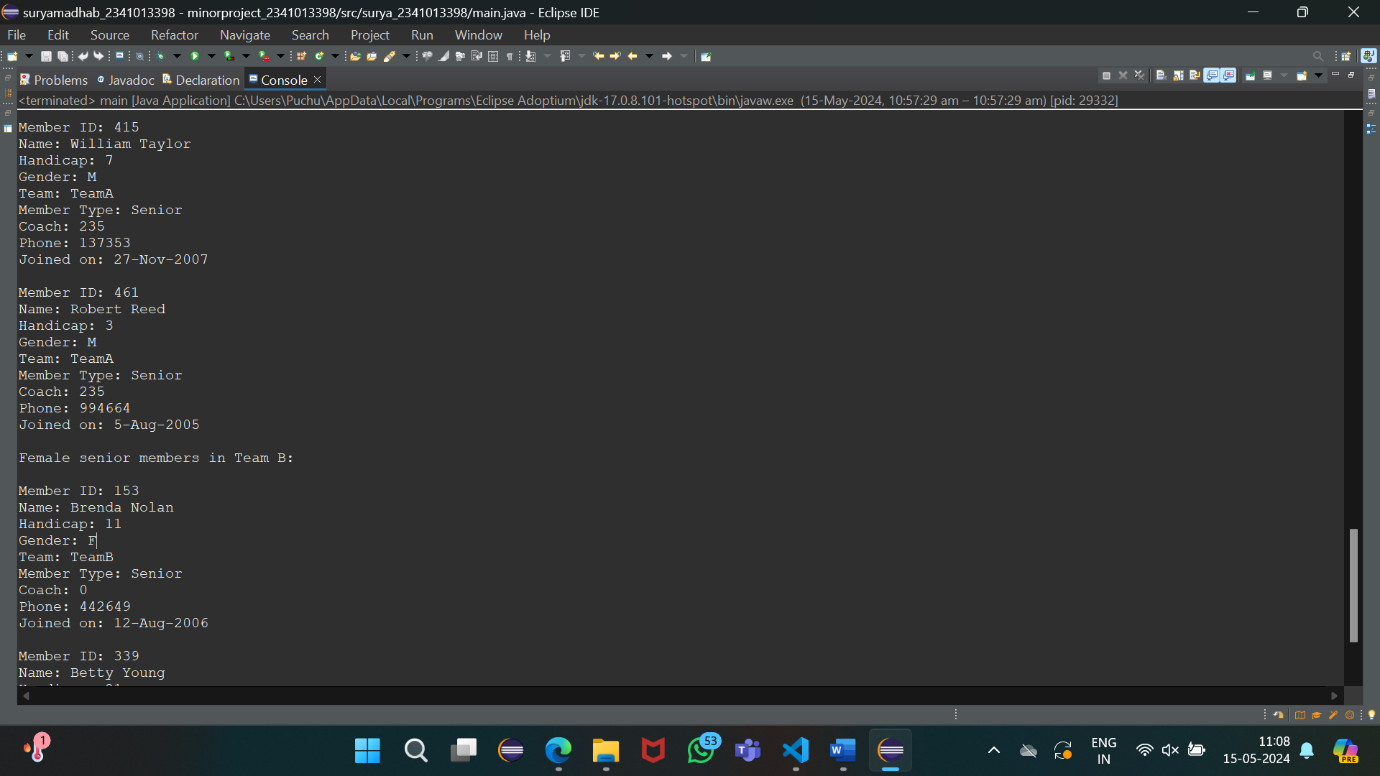
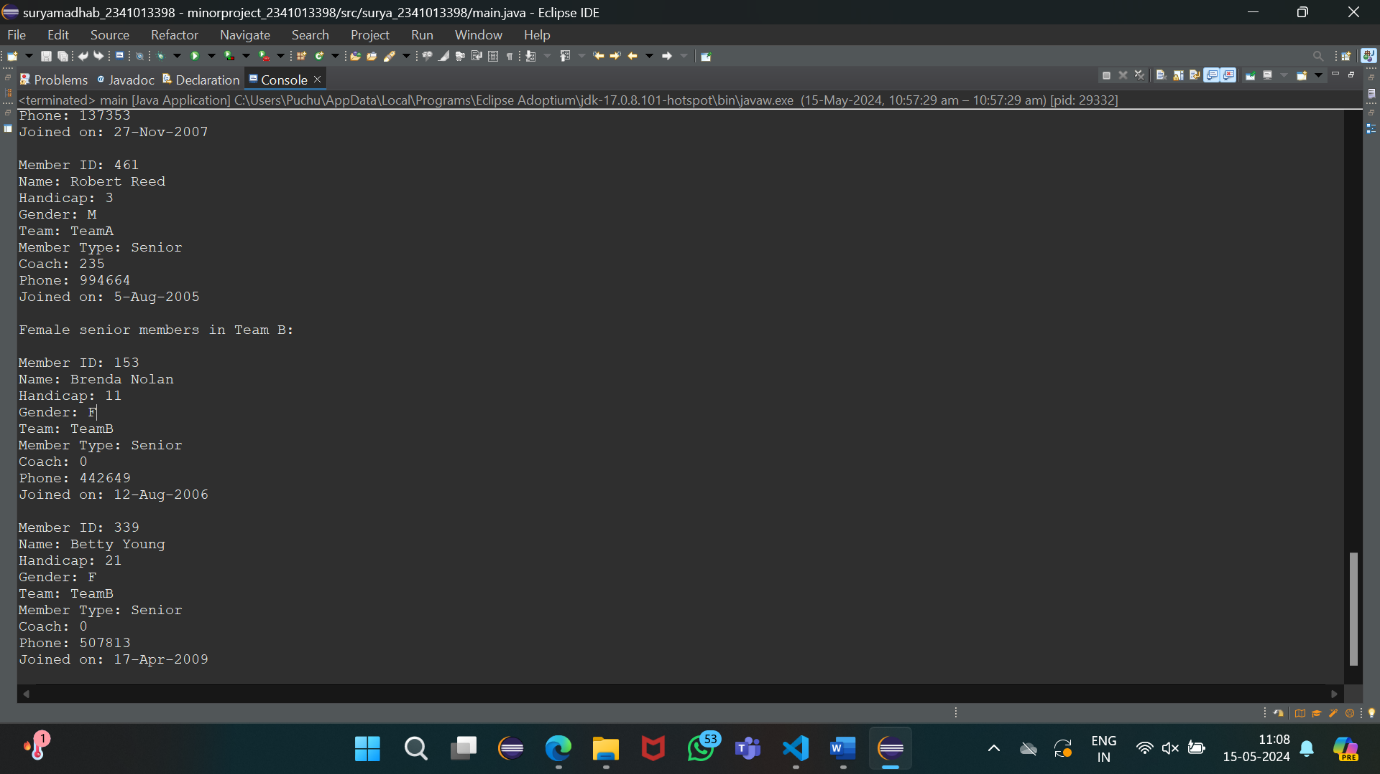
It shows senior members with a handicap score less than 12.

It lists female senior members who belong to "TeamB".



The code demonstrates the use of arrays, object-oriented programming concepts, conditional statements, and iteration to manage and filter member data based on specific criteria.

**Output:-**



**Members with join date earlier than 07-Apr-09:**

This section lists members who joined before April 7, 2009, along with their details such as Member ID, Name, Handicap, Gender, Team, Member Type, Coach, Phone, and Join Date.

**Senior members with handicap score less than 12:**

Here, senior members (presumably those over a certain age or experience level) with a handicap score less than 12 are listed. Again, details like Member ID, Name, Handicap, Gender, Team, Member Type, Coach, Phone, and Join Date are provided.

**Female senior members in Team B:**

This section specifically focuses on female senior members who are part of Team B. It provides similar details as in the previous sections but only for female senior members in this particular team.