

DOCUMENTATION WORKSHOP-2 PROJECT OBJECT-ORIENTED PROGRAMMING: NOTION (NOTES APP)

Juan Carlos Córdoba Asprilla – 20242020047

Sebastián Camilo Sánchez Cárdenas- 20242020086

Marzo 01 de 2025



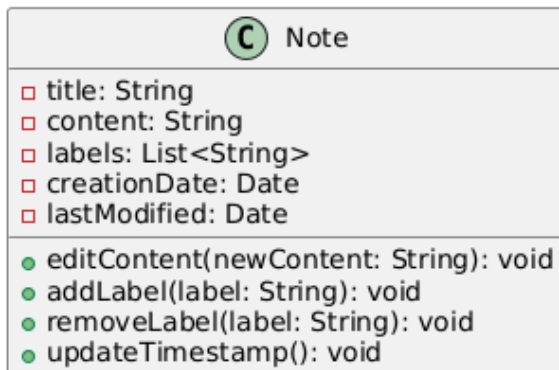
**UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS**

Object-oriented Programming

Ing. Carlos Andrés Sierra

CONCEPTUAL DESIGN UPDATES:

1. Addition of the Note Class



Reason:

Although the Notes class referenced a list of Note objects (notesList: List<Note>), the Note class itself was not explicitly defined. It has been created to represent a single note, fulfilling key user stories and functional requirements related to note creation, editing, tagging, and tracking modification times.

Attributes:

- title: String
- content: String
- labels: List<String>
- creationDate: Date
- lastModified: Date

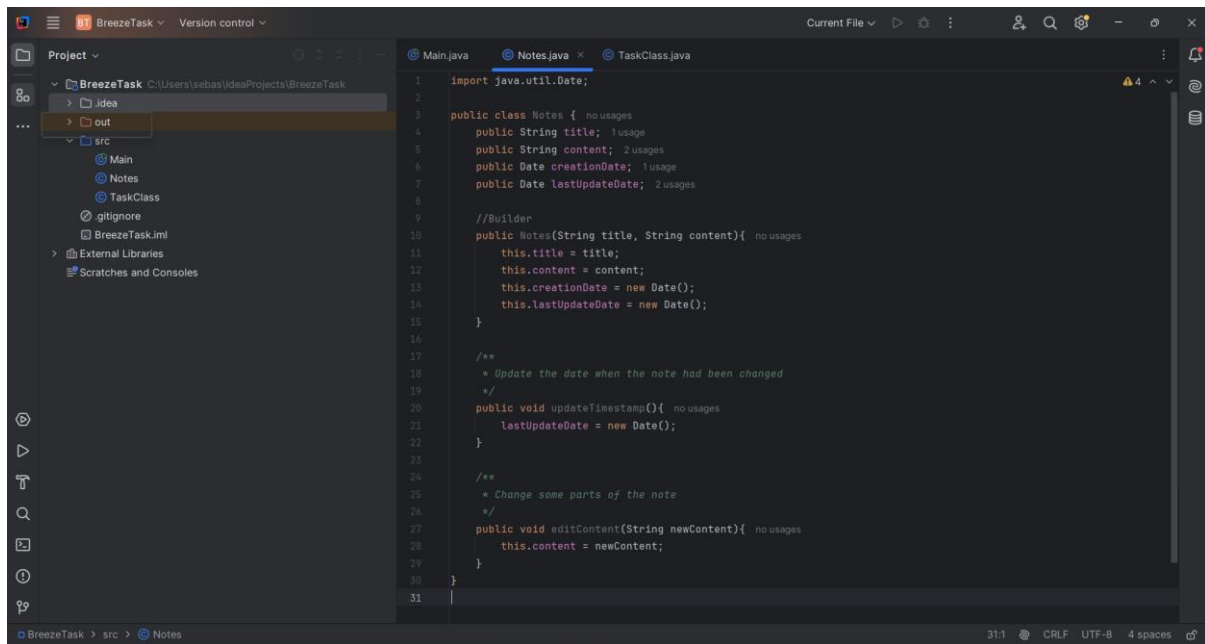
Methods:

- editContent(newContent: String): void
- addLabel(label: String): void
- removeLabel(label: String): void
- updateTimestamp(): void

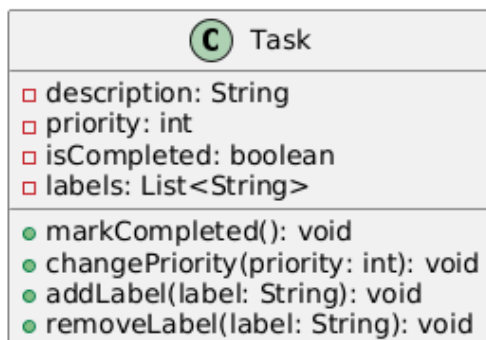
Relationships:

Notes --> Note

Note --> Storage



2. Addition of the Task Class



Reason:

The Tasks class was already listed as managing multiple tasks, but no definition existed for the individual Task entity. This class was added to model a single task with its priority, completion status, and labels—supporting user stories that involve goal tracking, task organization, and filtering.

Attributes:

- description: String
- priority: int (ejem., 1 = high, 2 = medium, 3 = low)
- isCompleted: boolean
- labels: List<String>

Methods:

- markCompleted(): void

- changePriority(priority: int): void
- addLabel(label: String): void
- removeLabel(label: String): void

Relationships:

Tasks --> Task

Task --> Storage

```

1  import java.util.Scanner;
2
3  public class TaskClass { no usages
4
5      public String title; 2 usages
6      public String description; 2 usages
7      public Integer priority; 2 usages
8      public boolean isCompleted; 3 usages
9
10     //Builder
11     public TaskClass(String title, String description){ no usages
12         this.title = title;
13         this.description = description;
14         this.priority = null;
15         this.isCompleted = false;
16     }
17
18     /**
19      * Ask the user the name of the task
20      *
21      */
22     public void askTitle() { no usages
23         System.out.print("What is the title of your task? ");
24         Scanner scanner = new Scanner(System.in);
25         title = scanner.nextLine();
26     }
27
28     /**
29      * Ask the user what is the purpose of its task
30      */
31     public void askDescription() { no usages
32         System.out.print("What is the description of your task? ");

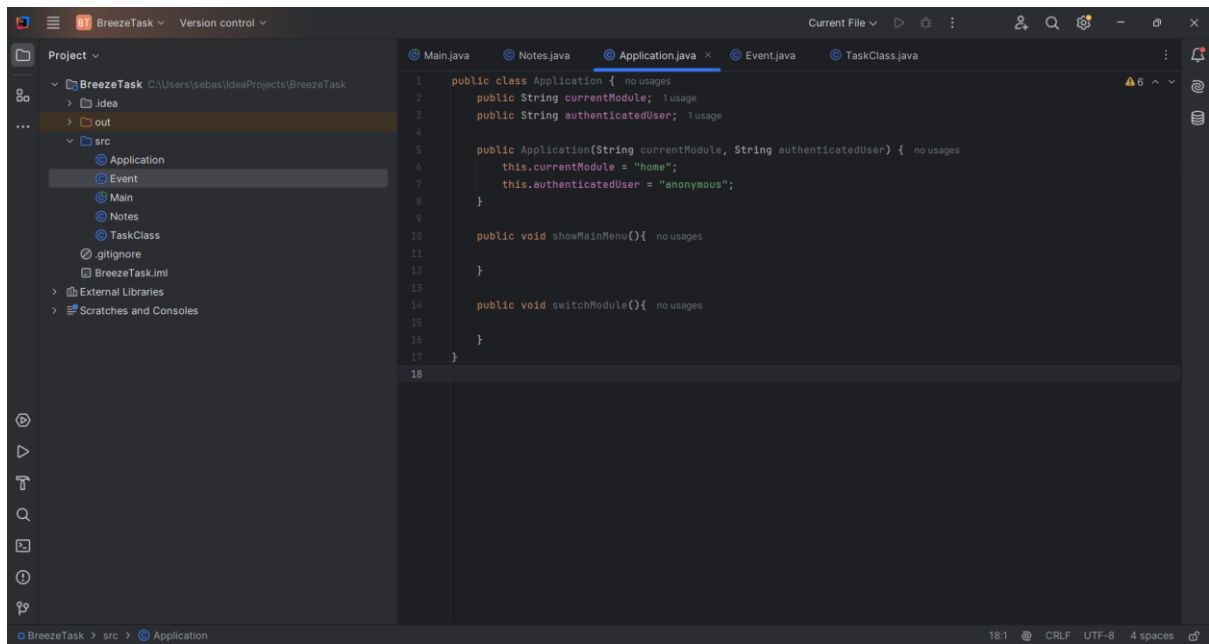
```

```

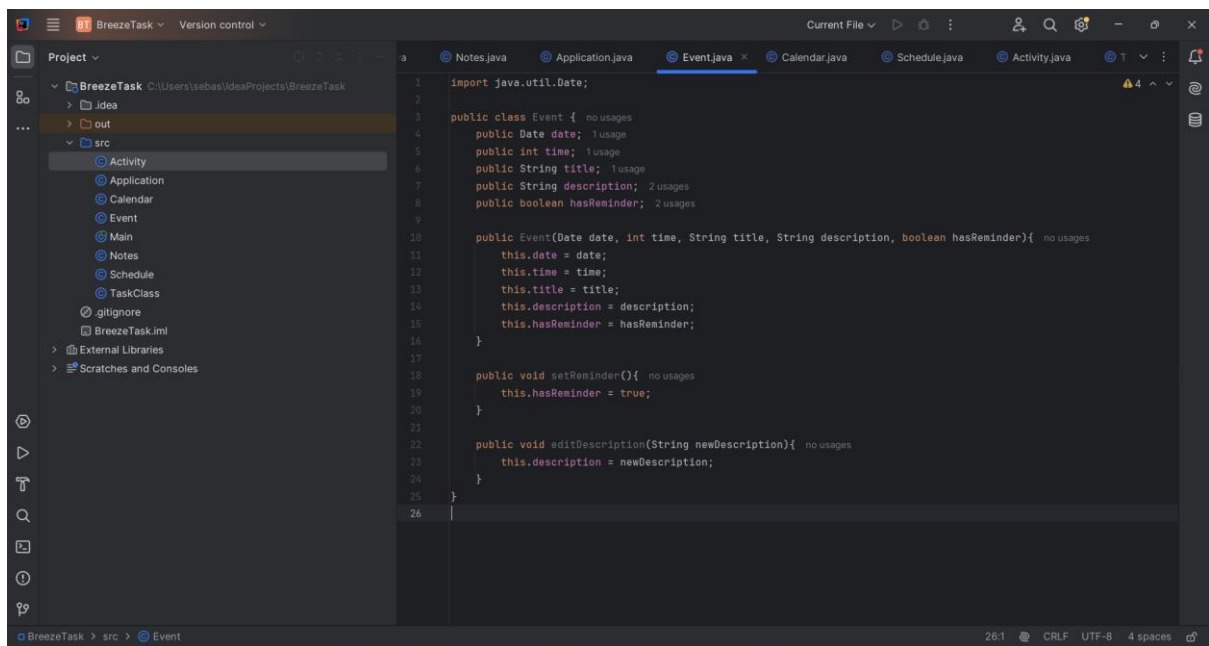
33         System.out.print("What is the description of your task? ");
34         Scanner scanner = new Scanner(System.in);
35         description = scanner.nextLine();
36     }
37
38     public void changedPriority() { no usages
39         System.out.print("Which is the priority of your task? ");
40         Scanner scanner = new Scanner(System.in);
41         priority = scanner.nextInt();
42     }
43
44     //Rename usages
45     public void markCompleted() {
46         System.out.print("Is your task completed? ");
47         Scanner scanner = new Scanner(System.in);
48         isCompleted = scanner.nextBoolean();
49         if (isCompleted == true) {
50             System.out.println("Task completed.");
51         } else {
52             System.out.println("Task not completed.");
53         }
54     }
55 }

```

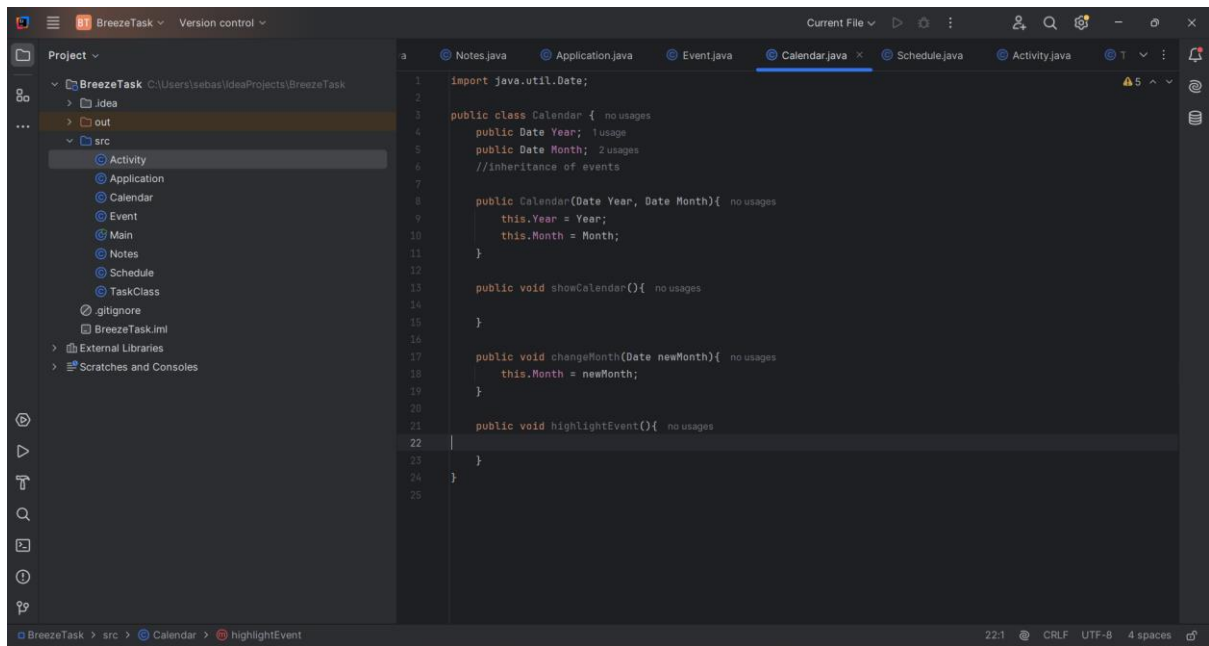
Application



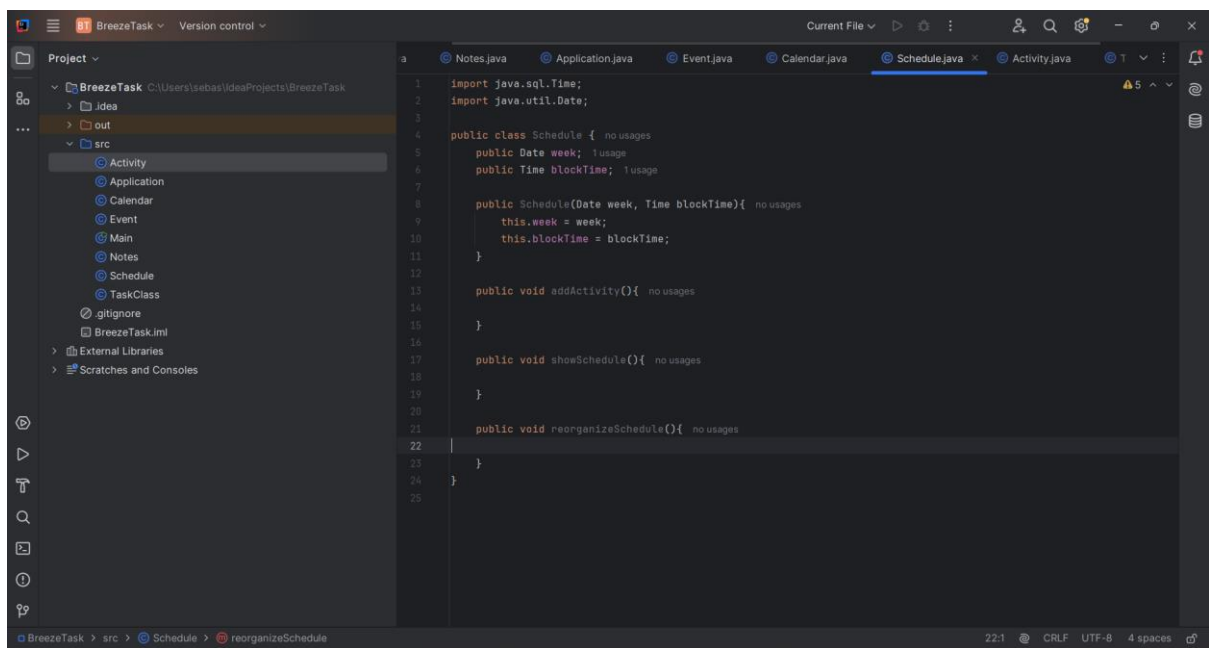
Event



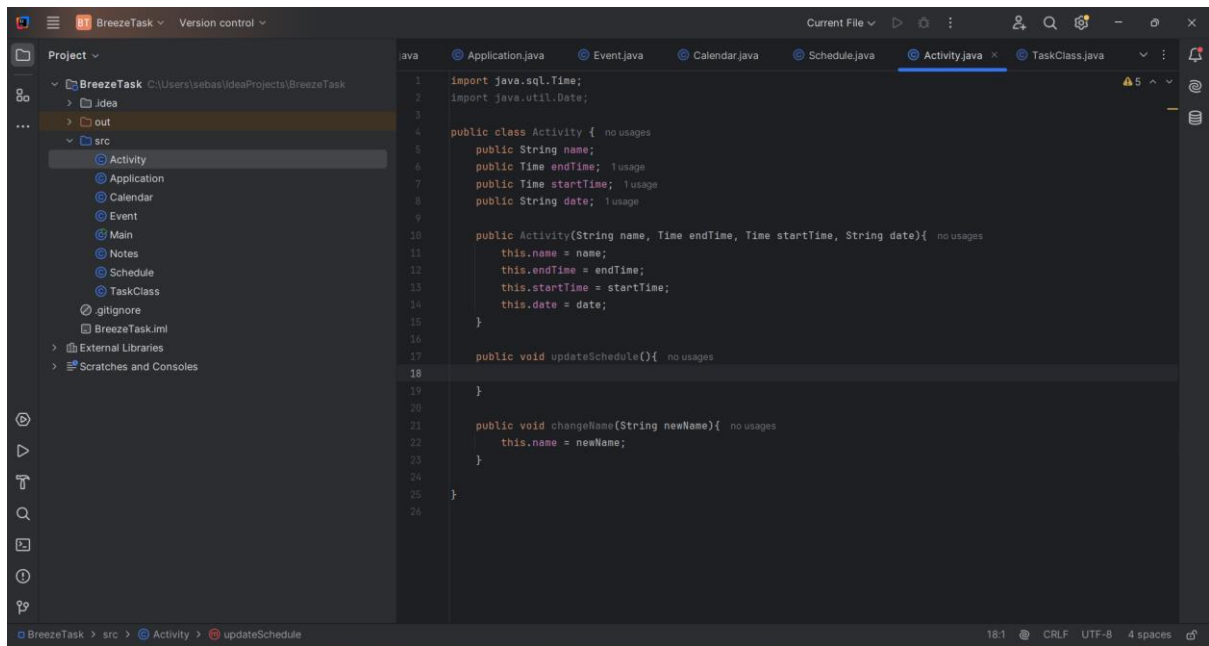
Calendar



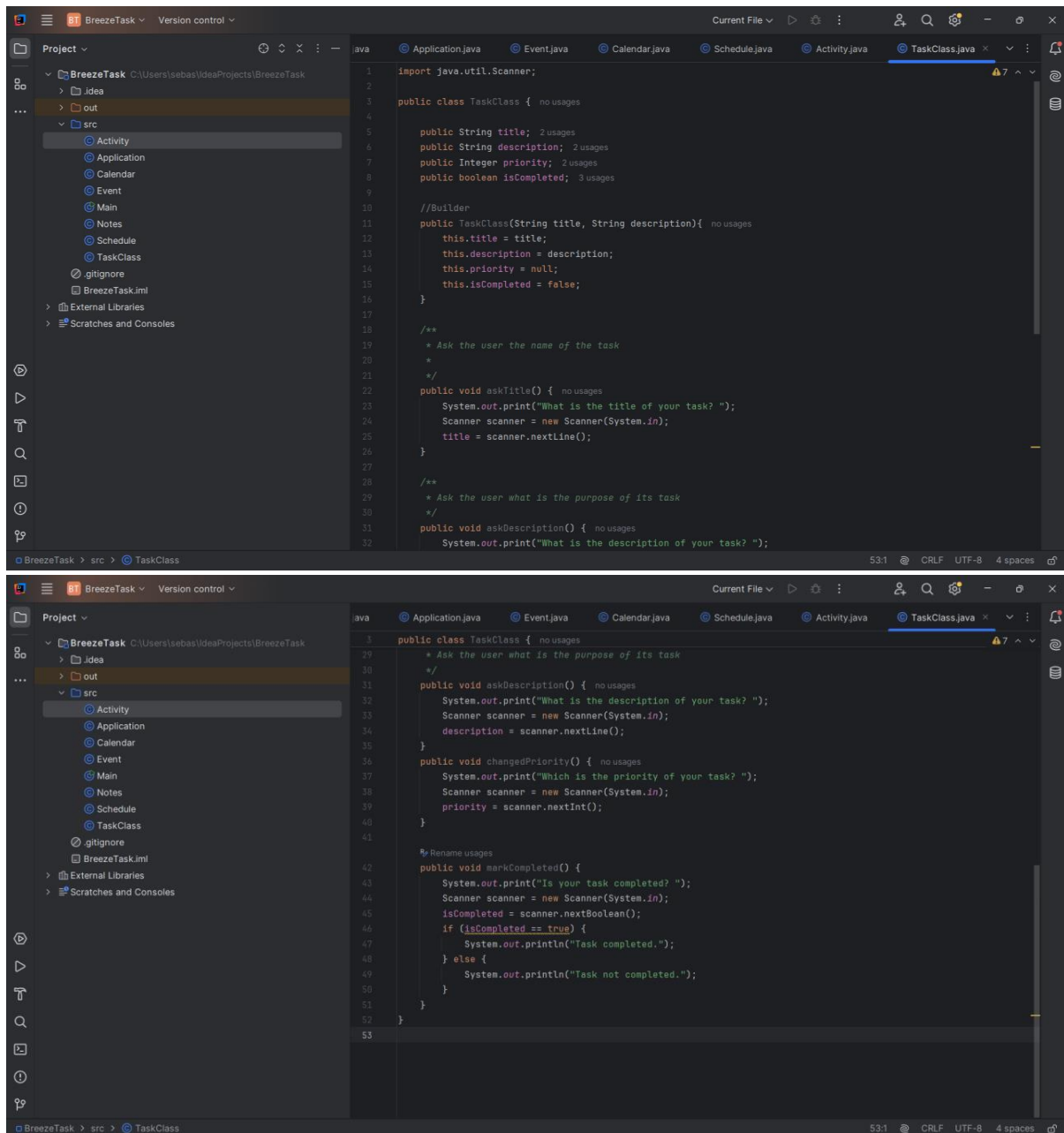
Schedule



Activity



Task



DIAGRAMS

[ACTIVITY DIAGRAM](#) - [UML CLASS DIAGRAM](#) - [SEQUENCE DIAGRAM](#)

Note: The diagrams shown below use classes with conceptual (yet to be defined) attributes and methods to establish their respective connections. In the future, the structure of these methods and the necessary attributes to include or exclude will be reviewed to ensure the application functions are as intended.

APPLICATION OF OOP CONCEPTS

At this stage of the application's development, the use of object-oriented programming principles is not yet fully defined or implemented, as many functionalities are focused on direct user interaction with specific modules.

Regarding **encapsulation**, protecting the user's information has been considered essential, especially since this is a local application — without online connectivity — meaning all data will be stored directly on the user's device. Therefore, appropriate access modifiers are planned to restrict direct access to attributes and ensure that any modifications are handled through well-defined public methods.

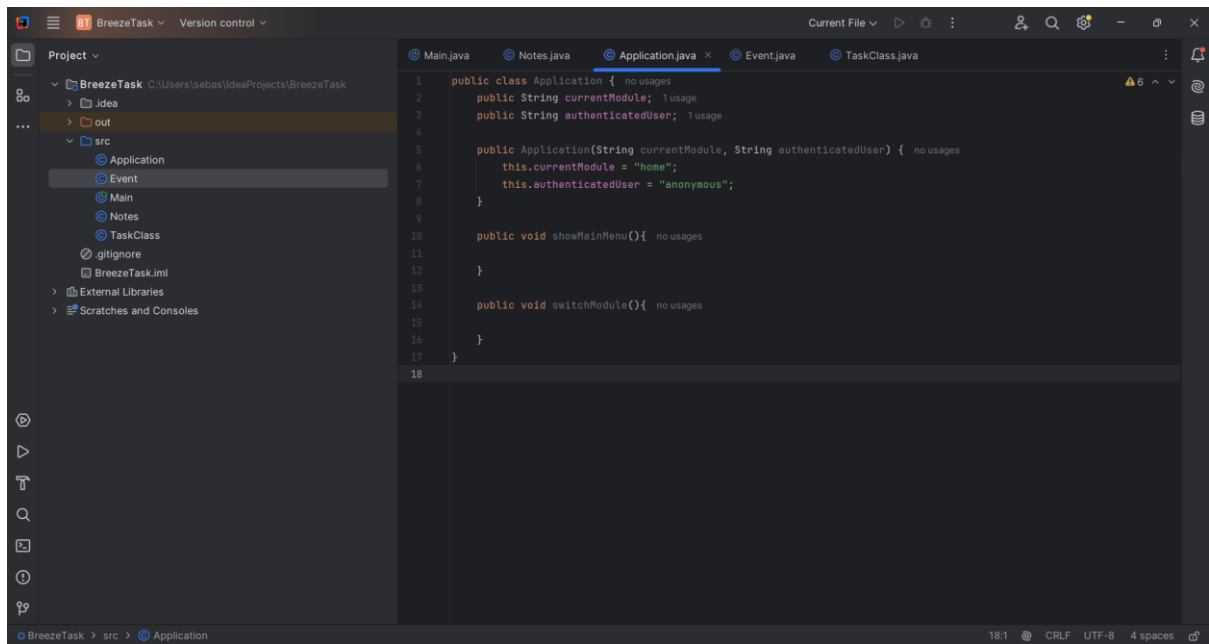
Concerning **inheritance**, the possibility of creating a base class called Module has been considered, from which components such as calendar, notes, tasks, and reminders could be derived. However, this decision is still under evaluation, as it must be determined whether it truly adds value or introduces unnecessary complexity into the design.

Lastly, regarding **polymorphism**, its implementation is not currently anticipated, as each class's responsibilities are clearly defined and do not require flexible or alternative behaviors. Each module operates independently and fulfills specific tasks, making the use of this concept currently unnecessary.

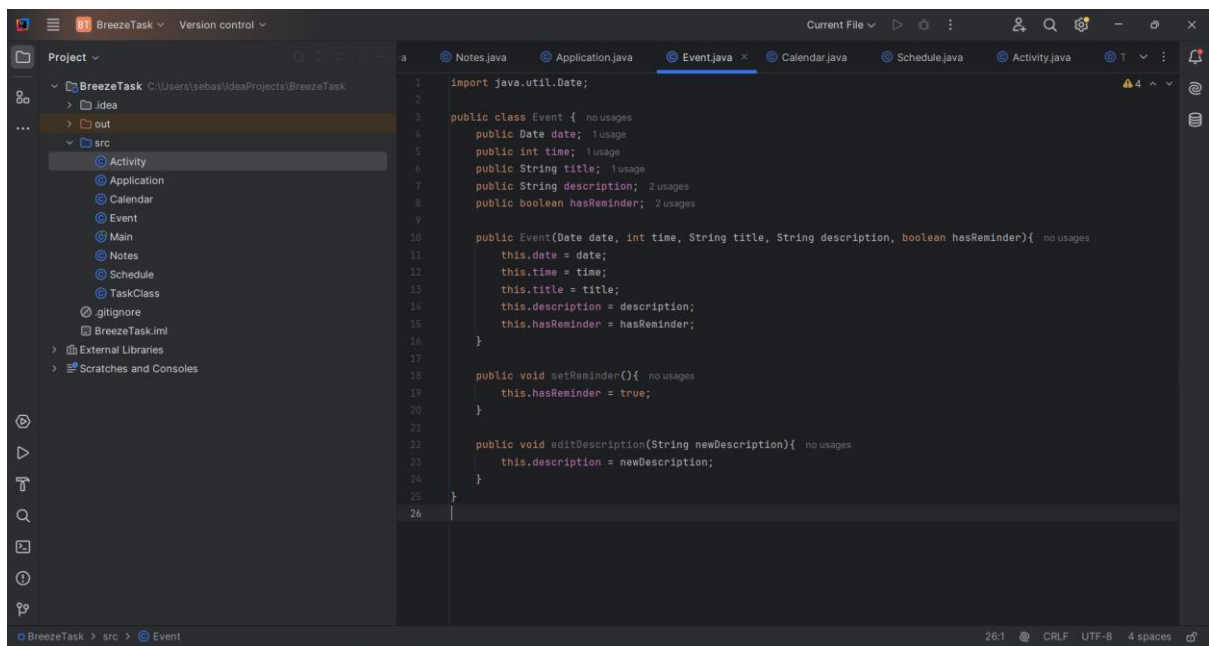
PACKAGE ESTRUCTURE

WORK IN PROGRESS

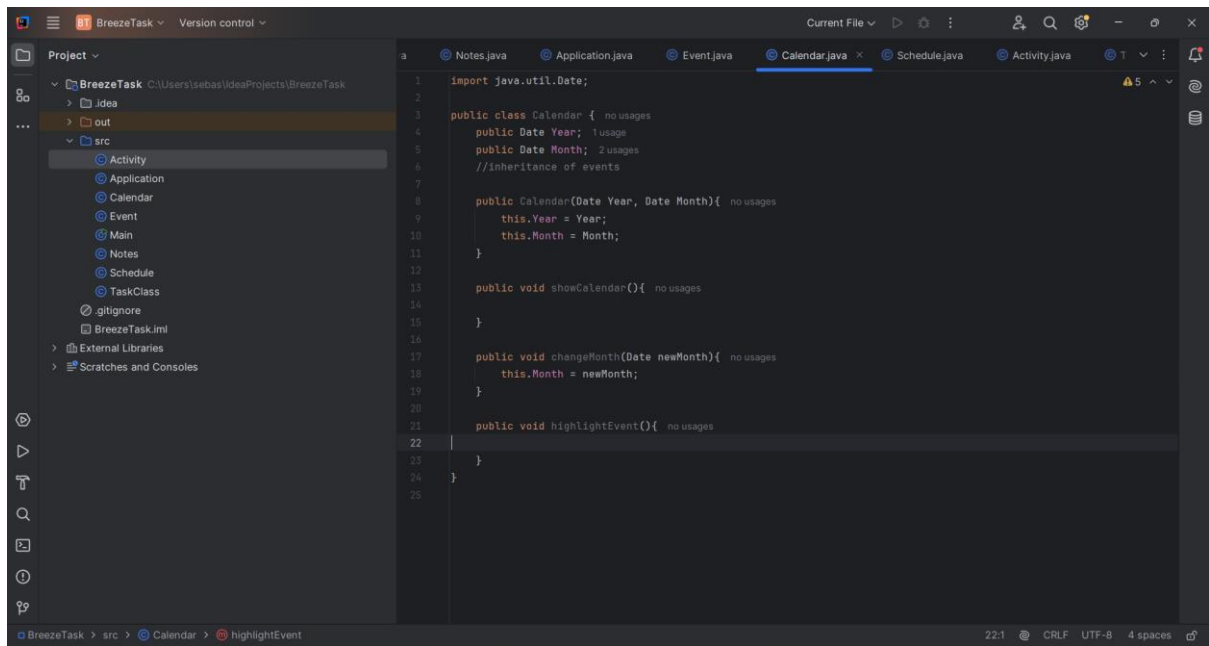
Application



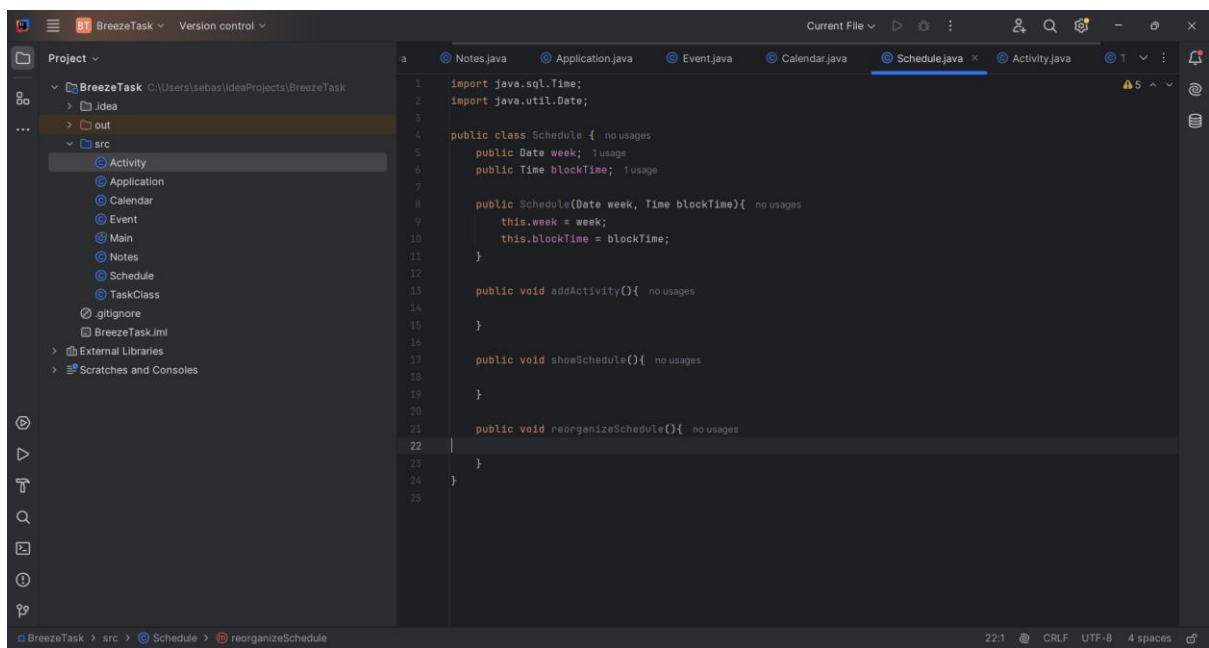
Event



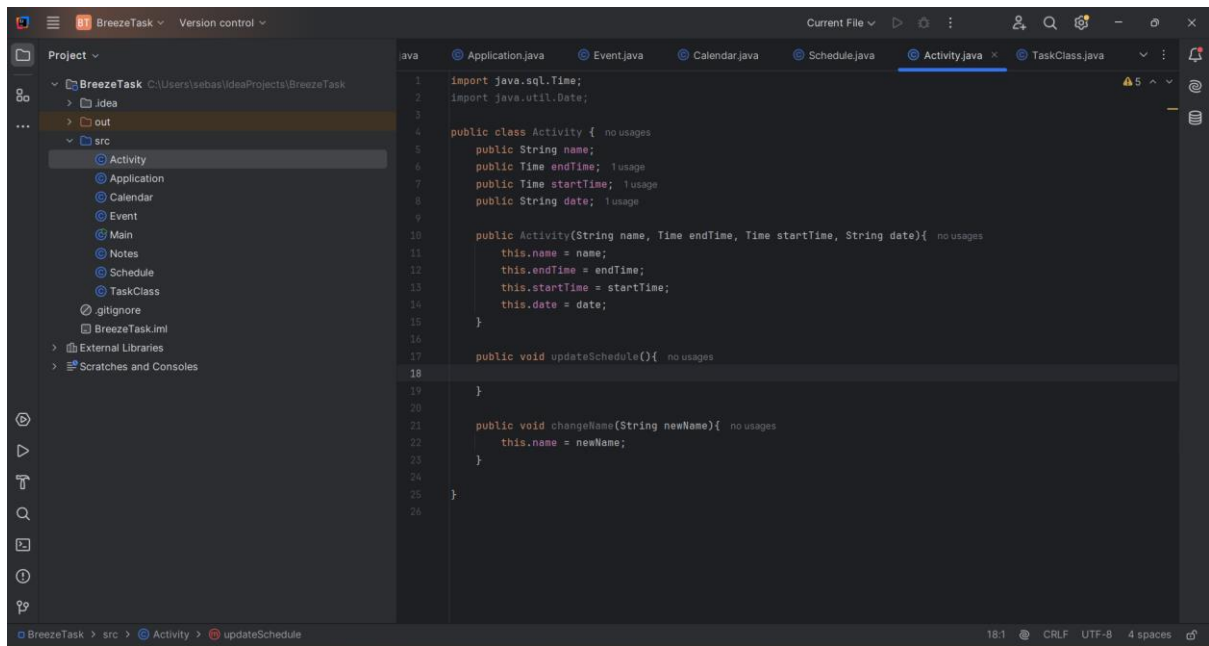
Calendar



Schedule



Activity



Task

