Workshop 2

Hector Fabian Cabrera Vargas

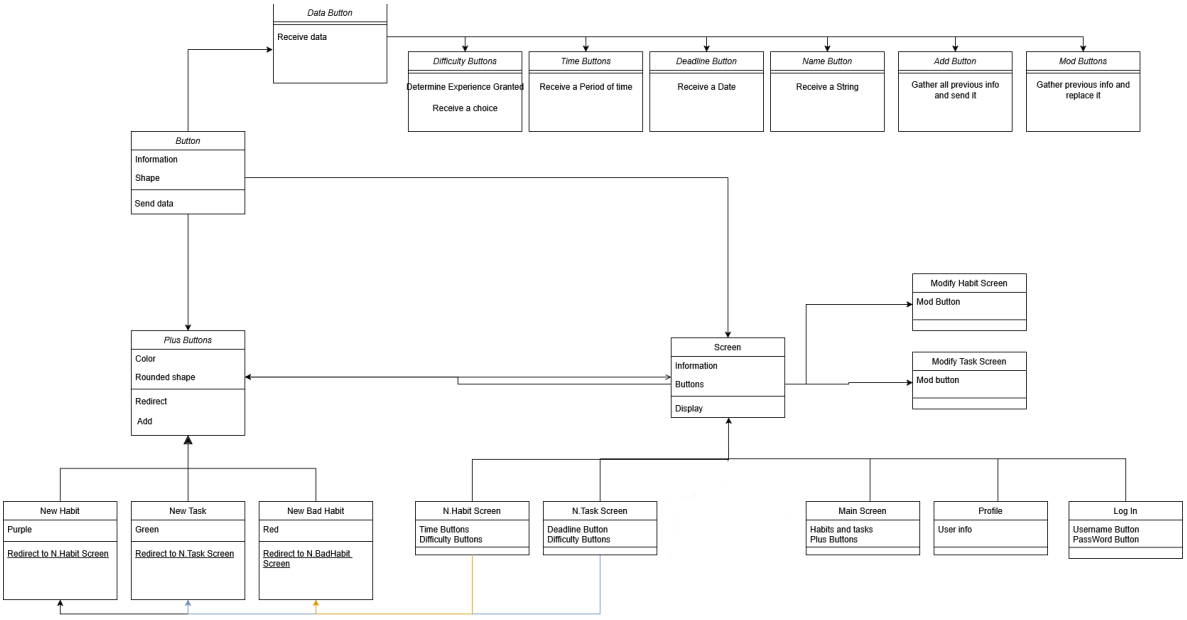
2020020

Juan Esteban Ávila Bautista

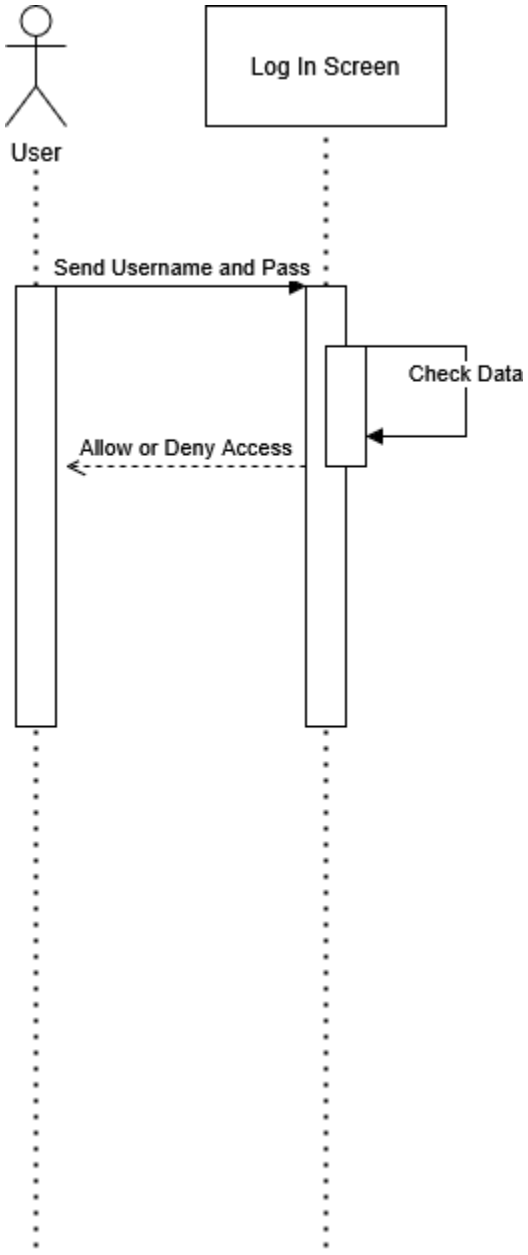
2020030

Conceptual Design: Habitra

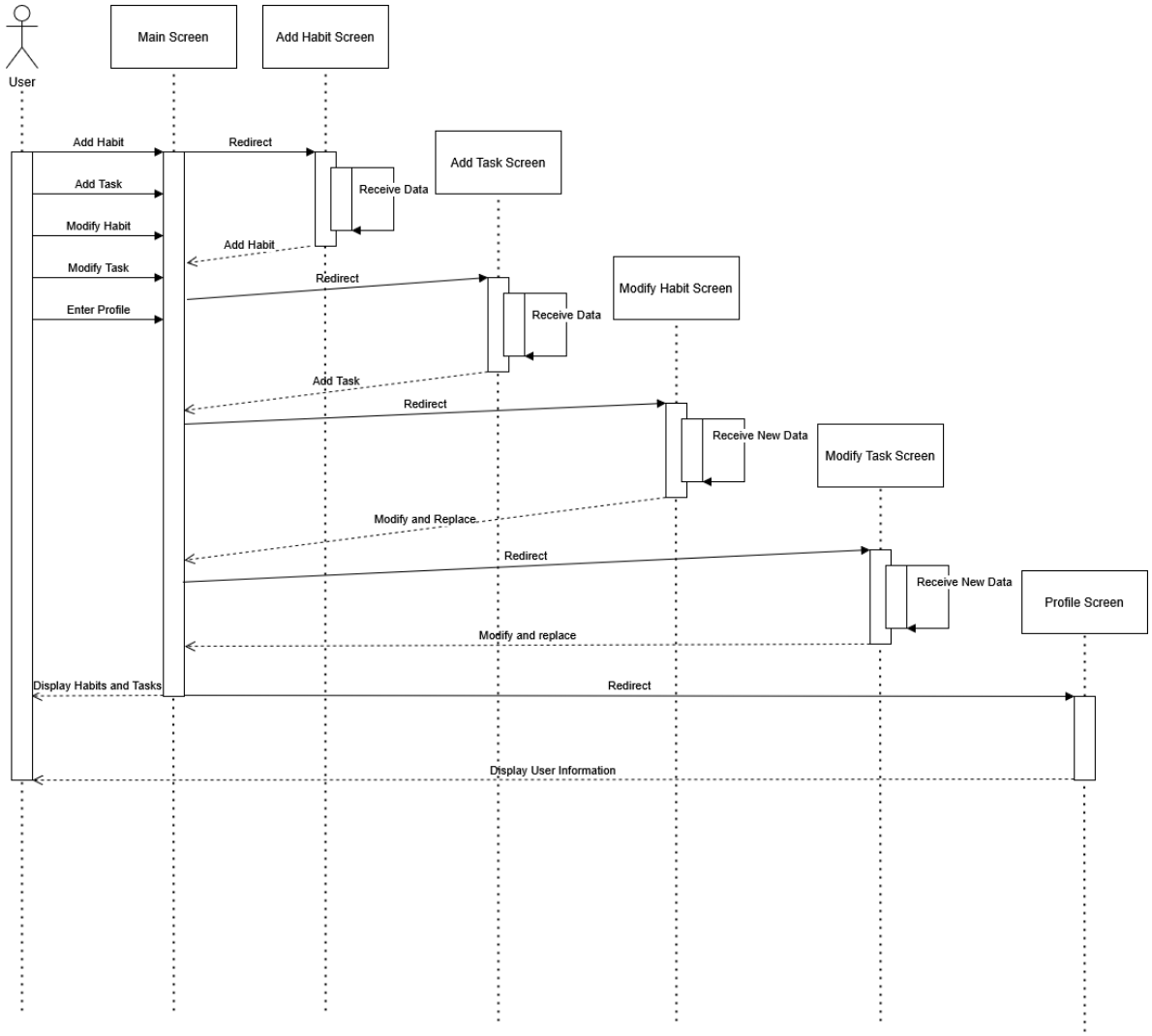
UML Diagrams:



Class Diagram for the Habitra app. Which shows their relations and attributes and methods.



Sequence Diagram that shows the process for User Log In.



Sequence Diagram for the main processes in the app.

Implementation of OOP concepts:

1)Encapsulation:

Encapsulation will be achieved by keeping user data private: Username and password will be kept private. To allow access to the app, a set method for both username and password will check if they are correct. If both are correct, the user will have access to the application.

2)Inheritance:

Inheritance will be achieved by the different screens and buttons the application has, since most of them work in almost identical manners. However, some of them have unique characteristics.

3)Polymorphism:

Polymorphism will be mainly achieved by overriding, since child classes will implement some methods and attributes differently, or have no need to use them at all. Such as buttons, that, despite having the same mother class, work differently, since not all of them receive data or redirect to other screens.

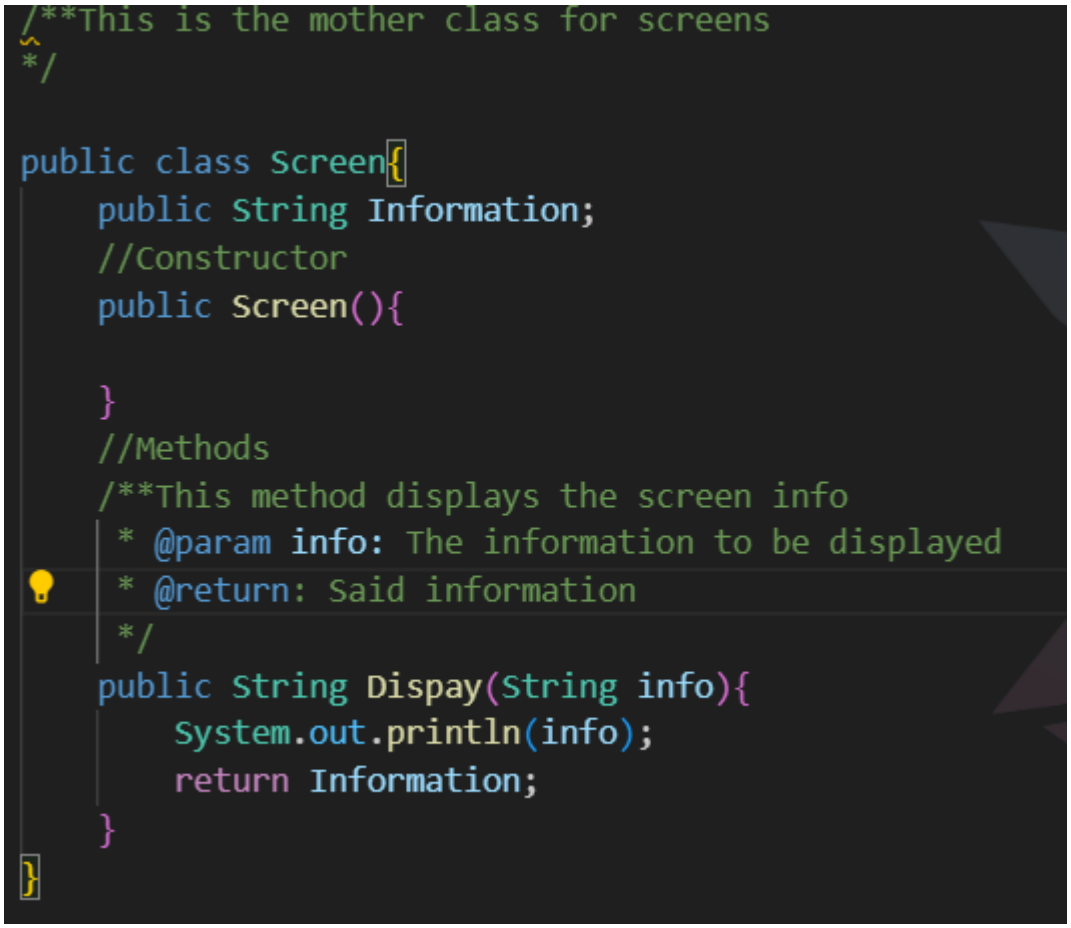
Overcharging is also present, with methods that have the same name, but different parameters.

Code Placeholders:

1) Main Classes

```java
/**This is the mother class for all buttons
 */
public class Button{
    private String S_Info;
    private int I_Info;
    //Constructor
    public Button(){

    }
    //Methods
    /**This method sends the associated information of the button
     * @param ButtonInfo: The assocaited information
     * @return: The information to be read, a string
    */
    public String Send(String ButtonInfo){
        return S_Info;
    }
    /**This method sends the associated information of the button
     * @param ButtonInfo: The assocaited information
     * @return: The information to be read, an interger
    */
    public Interger Send(int ButtonInfo){
        return I_Info;
    }
}
```

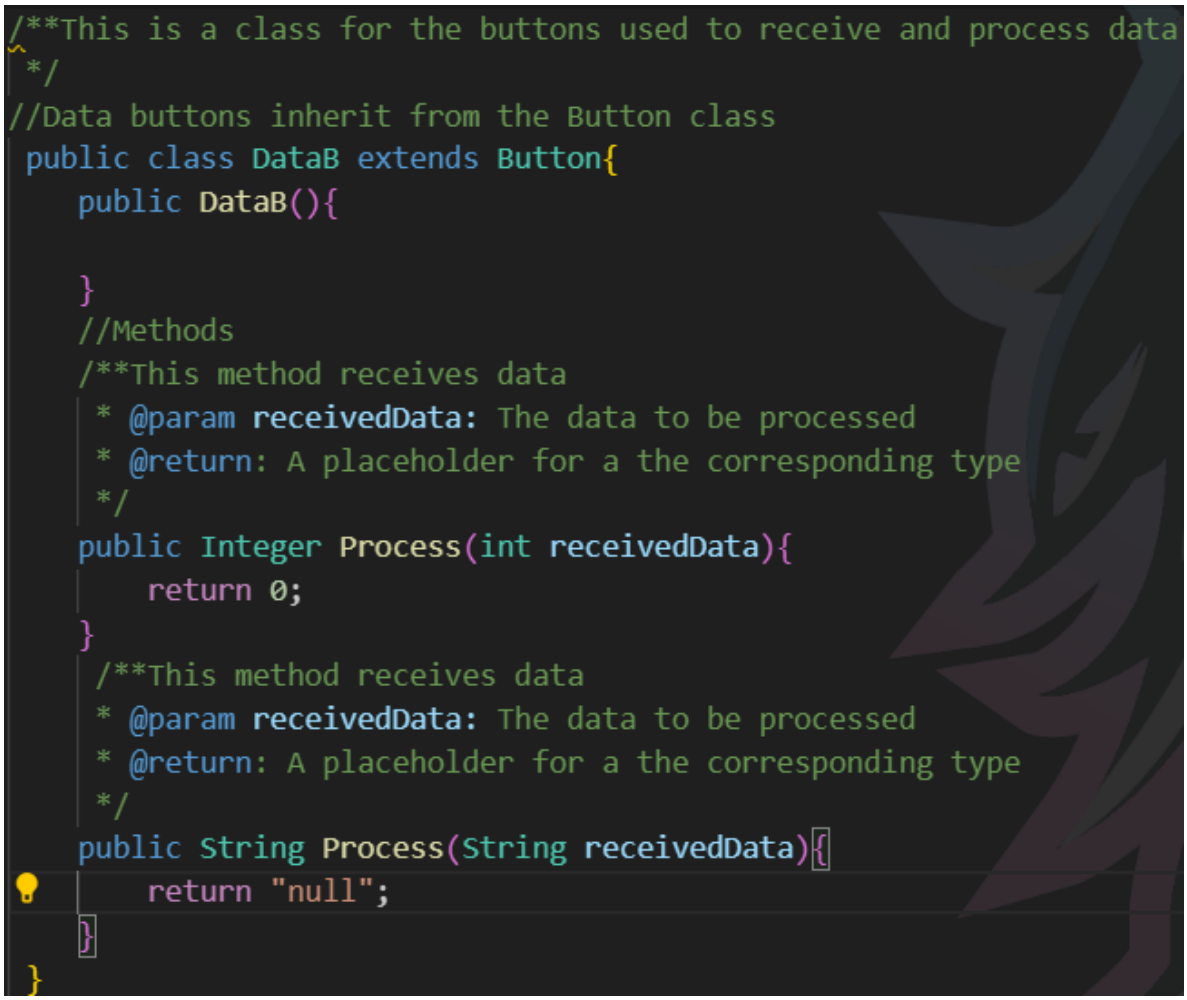
Placeholder for the Button Class



```java
/**This is the mother class for screens
*/

public class Screen{
    public String Information;
    //Constructor
    public Screen(){

    }
    //Methods
    /**This method displays the screen info
     * @param info: The information to be displayed
     * @return: Said information
     */
    public String Dispay(String info){
        System.out.println(info);
        return Information;
    }
}
```

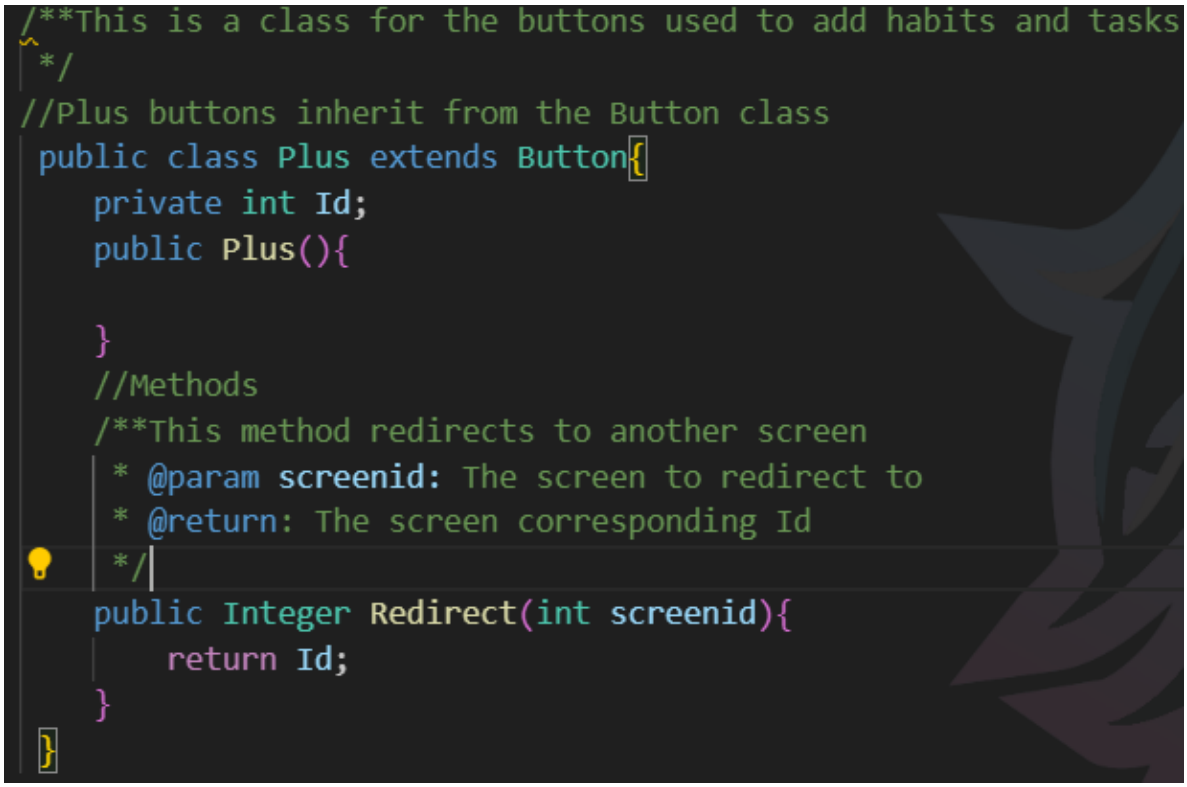
Placeholder for the Screen Class

2) Main Sub-Classes



```java
/**This is a class for the buttons used to receive and process data
 */
//Data buttons inherit from the Button class
public class DataB extends Button{
    public DataB(){

    }
    //Methods
    /**This method receives data
     * @param receivedData: The data to be processed
     * @return: A placeholder for a the corresponding type
     */
    public Integer Process(int receivedData){
        return 0;
    }
     /**This method receives data
     * @param receivedData: The data to be processed
     * @return: A placeholder for a the corresponding type
     */
    public String Process(String receivedData){
        return "null";
    }
}
```

Placeholder for the Data Button Child Class



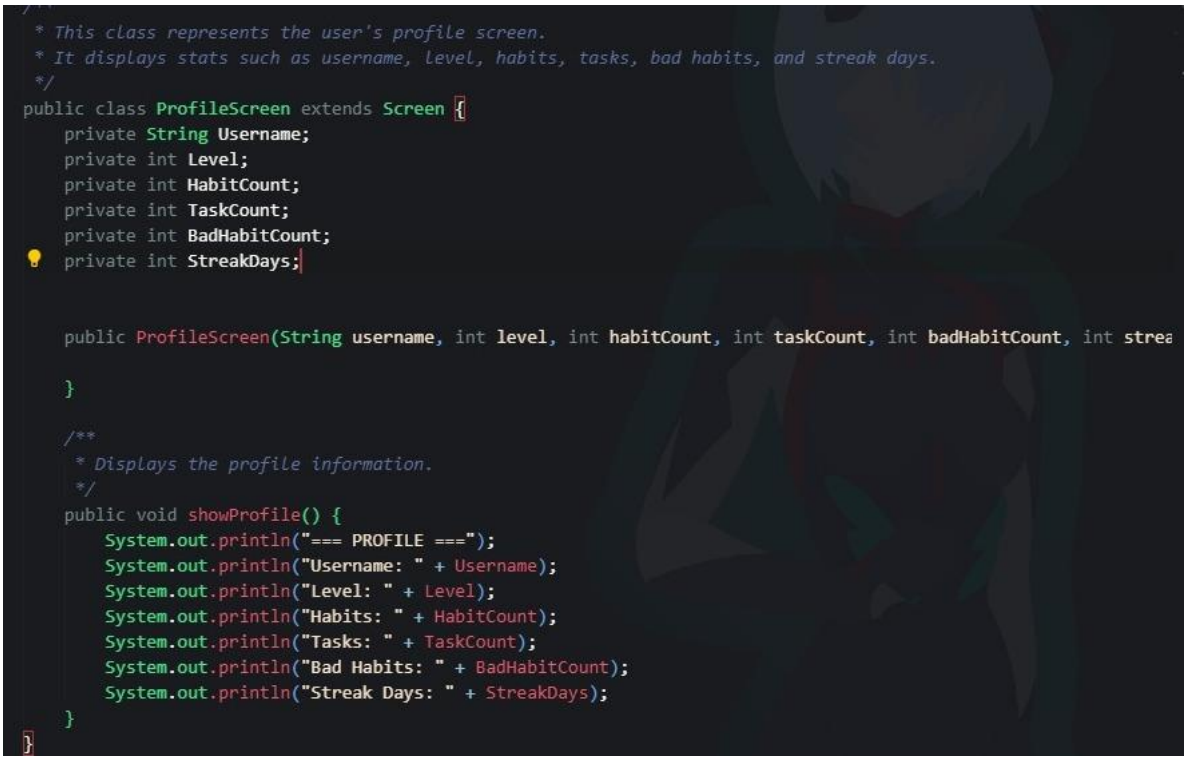
```
/**This is a class for the buttons used to add habits and tasks
 */
//Plus buttons inherit from the Button class
public class Plus extends Button{
    private int Id;
    public Plus(){

    }
    //Methods
    /**This method redirects to another screen
     * @param screenid: The screen to redirect to
     * @return: The screen corresponding Id
     */
    public Integer Redirect(int screenid){
        return Id;
    }
}
```

Placeholder for the Plus Button Child Class

```java
/*
 * This class represents a registration screen where the user
 * is prompted to enter a username and password.
 */
public class RegisterScreen extends Screen {
    private String username;
    private String password;

    public RegisterScreen() {
    }

    /**
     * Sets the username.
     * @param username The username to be set.
     */
    public void setUsername(String username) {
        this.username = username;
    }

    /**
     * Sets the password.
     * @param password The password to be set.
     */
    public void setPassword(String password) {
        this.password = password;
    }

    /**
     * Returns the entered username.
     * @return The username.
     */
    public String getUsername() {
        return username;
    }
```

```java
    /**
     * Returns the entered password.
     * @return The password.
     */
    public String getPassword() {
        return password;
    }

    /**
     * Displays a registration message asking the username.
     */
    public void ShowFormUsername() {
        System.out.println("U.NAME:");
    }

    /**
     * Displays a registration message asking the password.
     */
    public void ShowFormPassword() {
        System.out.println("PASS:");
    }
}
```

Placeholder for the Register screen child class

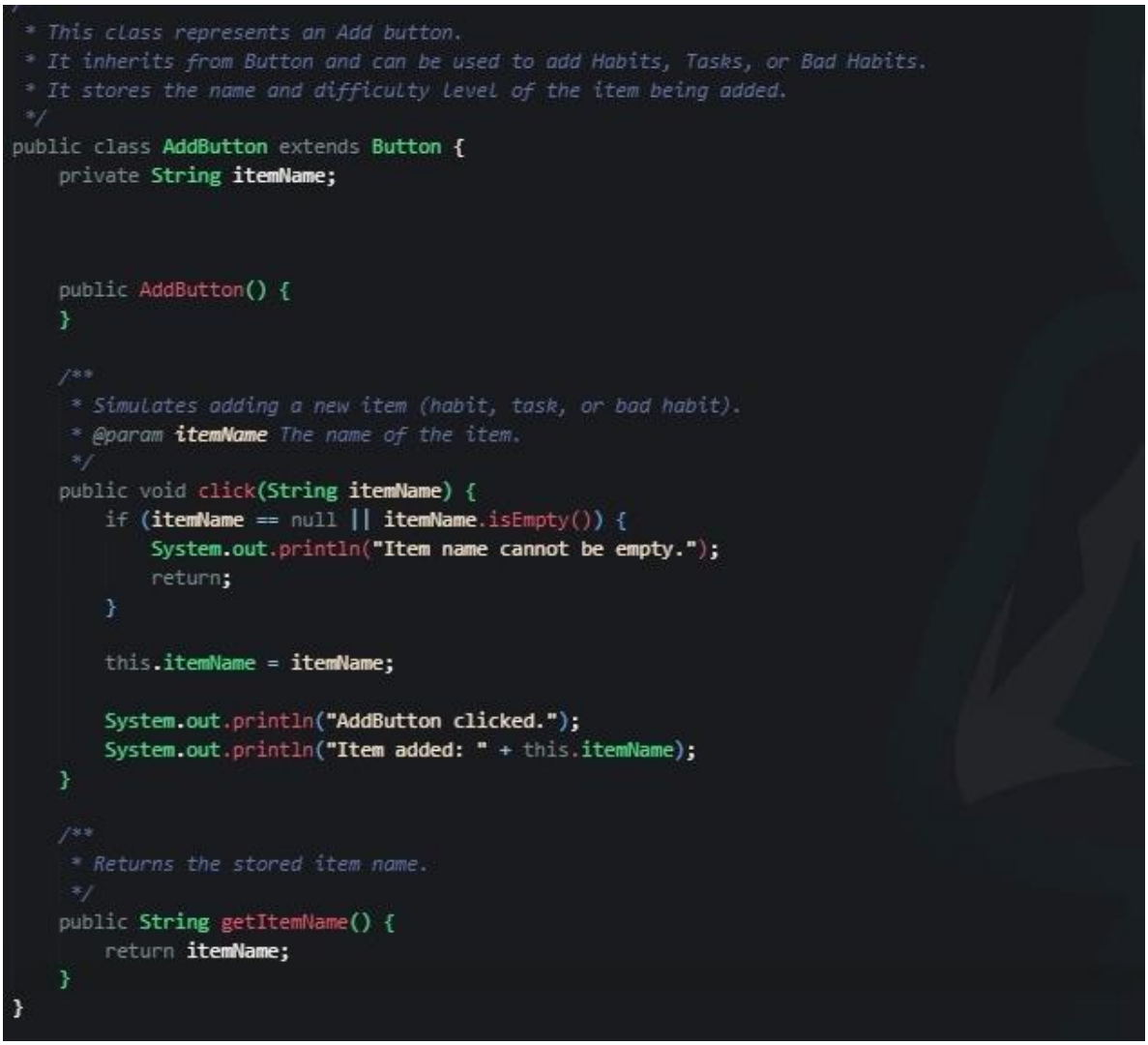


```java
/**
 * This class represents the user's profile screen.
 * It displays stats such as username, level, habits, tasks, bad habits, and streak days.
 */
public class ProfileScreen extends Screen {
    private String Username;
    private int Level;
    private int HabitCount;
    private int TaskCount;
    private int BadHabitCount;
    private int StreakDays;


    public ProfileScreen(String username, int level, int habitCount, int taskCount, int badHabitCount, int strea

    }

    /**
     * Displays the profile information.
     */
    public void showProfile() {
        System.out.println("=== PROFILE ===");
        System.out.println("Username: " + Username);
        System.out.println("Level: " + Level);
        System.out.println("Habits: " + HabitCount);
        System.out.println("Tasks: " + TaskCount);
        System.out.println("Bad Habits: " + BadHabitCount);
        System.out.println("Streak Days: " + StreakDays);
    }
}
```

Placeholder for the Profile screen child class

```java
/**
 * This class represents a login button.
 * It inherits from the Button class and simulates saving user credentials.
 */
public class LogInButton extends Button {
    private String Username;
    private String Password;

    /**
     * Constructor
     */
    public LogInButton() {
    }

    /**
     * Makes a login action by storing the username and password.
     * @param username The username to store.
     * @param password The password to store.
     */
    public void click(String username, String password) {
        this.Username = username;
        this.Password = password;
        System.out.println("Username stored: " + Username);
        System.out.println("Login simulated successfully.");
    }

    /**
     * Gets the stored username.
     * @return The stored username.
     */
    public String getStoredUsername() {
        return Username;
    }

    /**
     * Gets the stored password.
     * @return The stored password.
     */
    public String getStoredPassword() {
        return Password;
    }
}
```

Placeholder for the log in button child class



```java
/**
 * This class represents an Add button.
 * It inherits from Button and can be used to add Habits, Tasks, or Bad Habits.
 * It stores the name and difficulty level of the item being added.
 */
public class AddButton extends Button {
    private String itemName;



    public AddButton() {
    }

    /**
     * Simulates adding a new item (habit, task, or bad habit).
     * @param itemName The name of the item.
     */
    public void click(String itemName) {
        if (itemName == null || itemName.isEmpty()) {
            System.out.println("Item name cannot be empty.");
            return;
        }

        this.itemName = itemName;

        System.out.println("AddButton clicked.");
        System.out.println("Item added: " + this.itemName);
    }

    /**
     * Returns the stored item name.
     */
    public String getItemName() {
        return itemName;
    }
}
```

Placeholder for the add button child class

Workshop 1 reconsiderations:

Added a new child class for Button: The Return Button

16)

Return button
Responsibilities: • Return user to the previous screen
Collaborators: • New habit screen • New task screen • Name Button • User • Modify Habit Screen • Modify Task Screen

Please refer to the Change History section of the workshop for details.

Change History

Modification	Reason	By	Date