# Main Page of Code

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Main.java to edit this template
 */
package javaapp2;

import javax.swing.ButtonGroup;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JRadioButton;
import javax.swing.UIManager;
import java.awt.Color;

public class JavaApp2 {

    /**
     * @param args the command line arguments
     */
```

```
    public static void main(String[] args) {
        try {
            UIManager.put("OptionPane.background", Color.PINK);
            UIManager.put("Panel.background", Color.PINK);

            JRadioButton loginRadioButton = new JRadioButton("Login");
            JRadioButton registerRadioButton = new JRadioButton("Register");

            ButtonGroup buttonGroup = new ButtonGroup();
            buttonGroup.add(loginRadioButton);
            buttonGroup.add(registerRadioButton);

            JPanel panel = new JPanel();
            panel.setBackground(Color.PINK);
            panel.add(loginRadioButton);
            panel.add(registerRadioButton);

            JOptionPane.showMessageDialog(null, panel, "Welcome to EasyKanban",
JOptionPane.PLAIN_MESSAGE);

            Login login = new Login();
```

```
            if (loginRadioButton.isSelected()) {
                login(login);
            } else if (registerRadioButton.isSelected()) {
                register(login);
            }

            TaskManagementSystem taskManagementSystem = new TaskManagementSystem();
            taskManagementSystem.start();
        } catch (Exception e) {
            JOptionPane.showMessageDialog(null, "An error occurred.", "Error",
JOptionPane.ERROR_MESSAGE);
        }
    }

    public static void register(Login login) {
        try {
            login.setFirstName();
            login.setLastName();

            while (true) {
                String registerUser = login.registerUser();
```

```
            if (registerUser.equals("\nThe username and password meet the requirements, and
the user has been registered successfully.")) {

                break;

            }



            JOptionPane.showMessageDialog(null, "Please try again.", "Registration Failed",
JOptionPane.ERROR_MESSAGE);

        }

    } catch (Exception e) {

        throw e;

    }

  }



  public static void login(Login login) {

    boolean loginSuccessful = false;

    JOptionPane.showMessageDialog(null, "\nPlease enter your login details.");



    while (true) {

        try {

            loginSuccessful = login.getLoginDetails();
```

```
            if (loginSuccessful) {
                return;
            }
            JOptionPane.showMessageDialog(null, "Username or Password is incorrect. Please try again.", "Login Failed", JOptionPane.ERROR_MESSAGE);
        } catch (Exception e) {
            throw e;
        }
    }
  }
}
```

# Task Class

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package javaapp2;

import java.util.ArrayList;
import java.util.List;
import javax.swing.JOptionPane;
```

```
public class Task {

    private List<String> developer;

    private List<String> taskNames;

    private List<String> taskID;

    private List<String> status;

    private List<Integer> duration;


    public Task() {

        developer = new ArrayList<>();

        taskNames = new ArrayList<>();

        taskID = new ArrayList<>();

        status = new ArrayList<>();

        duration = new ArrayList<>();

    }

public void addTask(int count) {

        for (int i = 0; i < count; i++) {

            try {

                String developerName = JOptionPane.showInputDialog("Enter the developer name for
task " + (i + 1));

                String taskName = JOptionPane.showInputDialog("Enter the task name for task " + (i +
1));
```

```
            String taskID = JOptionPane.showInputDialog("Enter the task ID for task " + (i + 1));

            int taskDuration = Integer.parseInt(JOptionPane.showInputDialog("Enter the duration
for task " + (i + 1)));

            String taskStatus = JOptionPane.showInputDialog("Enter the status for task " + (i + 1) +
" (done, doing, to do)");



            developer.add(developerName);

            taskNames.add(taskName);

            this.taskID.add(taskID);

            duration.add(taskDuration);

            status.add(taskStatus);

        } catch (NumberFormatException e) {

            JOptionPane.showMessageDialog(null, "Invalid duration entered. Please enter a valid
number.");

            i--; // Decrement i to repeat the current iteration and prompt for input again

        }

    }



    JOptionPane.showMessageDialog(null, "Tasks added successfully!");

  }



  public int returnTotalHours() {
```

```
    int total = 0;
    for (int dur : duration) {
        total += dur;
    }
    return total;
}

public String searchTask(String taskName) {
    StringBuilder searchResult = new StringBuilder("Search Result:\n\n");
    boolean found = false;
    for (int i = 0; i < taskNames.size(); i++) {
        if (taskNames.get(i).equalsIgnoreCase(taskName)) {
            searchResult.append("Task Name: ").append(taskNames.get(i)).append("\n");
            searchResult.append("Task ID: ").append(taskID.get(i)).append("\n");
            searchResult.append("Developer: ").append(developer.get(i)).append("\n");
            searchResult.append("Duration: ").append(duration.get(i)).append(" hours\n");
            searchResult.append("Status: ").append(status.get(i)).append("\n\n");
            found = true;
        }
    }
    if (!found) {
```

```
        searchResult.append("Task not found.");
    }
    return searchResult.toString();
}

public String deleteTask(String taskName) {
    StringBuilder deleteResult = new StringBuilder();
    boolean deleted = false;
    for (int i = 0; i < taskNames.size(); i++) {
        if (taskNames.get(i).equalsIgnoreCase(taskName)) {
            taskNames.remove(i);
            taskID.remove(i);
            developer.remove(i);
            status.remove(i);
            duration.remove(i);
            deleteResult.append("Task \"").append(taskName).append("\" deleted.");
            deleted = true;
            break;
        }
    }
    if (!deleted) {
```

```
            deleteResult.append("Task not found.");
        }
        return deleteResult.toString();
    }

    public String displayLongest() {
        int longestDuration = 0;
        int longestIndex = -1;
        for (int i = 0; i < duration.size(); i++) {
            if (duration.get(i) > longestDuration) {
                longestDuration = duration.get(i);
                longestIndex = i;
            }
        }
        if (longestIndex != -1) {
            StringBuilder longestResult = new StringBuilder("Developer with Longest
Duration:\n\n");
            longestResult.append("Developer:
").append(developer.get(longestIndex)).append("\n");
            longestResult.append("Task Name:
").append(taskNames.get(longestIndex)).append("\n");
            longestResult.append("Task ID: ").append(taskID.get(longestIndex)).append("\n");
```

```
            longestResult.append("Duration: ").append(duration.get(longestIndex)).append("
hours");

            return longestResult.toString();

        } else {

            return "No tasks found.";

        }

    }



    public String searchDeveloper(String devName) {

        StringBuilder searchResult = new StringBuilder("Search Result:\n\n");

        boolean found = false;

        for (int i = 0; i < developer.size(); i++) {

            if (developer.get(i).equalsIgnoreCase(devName)) {

                searchResult.append("Developer: ").append(developer.get(i)).append("\n");

                searchResult.append("Task Name: ").append(taskNames.get(i)).append("\n");

                searchResult.append("Task ID: ").append(taskID.get(i)).append("\n");

                searchResult.append("Duration: ").append(duration.get(i)).append(" hours\n");

                searchResult.append("Status: ").append(status.get(i)).append("\n\n");

                found = true;

            }

        }
```

```
    if (!found) {
        searchResult.append("Developer not found.");
    }
    return searchResult.toString();
}

public void updateTaskStatus(String taskName, String status) {
    for (int i = 0; i < taskNames.size(); i++) {
        if (taskNames.get(i).equalsIgnoreCase(taskName)) {
            this.status.set(i, status);
            break;
        }
    }
}
public List<String> getDeveloper() {
    return developer;
}

public List<String> getTaskNames() {
    return taskNames;
}
```

```
    public List<String> getTaskID() {
        return taskID;
    }

    public List<String> getStatus() {
        return status;
    }

    public List<Integer> getDuration() {
        return duration;
    }
}
```

# Task Management Code

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package javaapp2;
import javax.swing.JOptionPane;
```

```
public class TaskManagementSystem {
    private Task task;
    private int totalHours;

    public TaskManagementSystem() {
        task = new Task();
        totalHours = 0;
    }

    public void start() {
        try {
            while (true) {
                Object[] options = { "Add Task", "Show Report", "Search for Task", "Delete Task",
                        "Display Developer with Longest Duration", "Search Task Assigned to Developer",
                        "Mark Task as Done", "Quit" };

                Object choice = JOptionPane.showInputDialog(null, "Select an option:", "Task
Management System",
                        JOptionPane.PLAIN_MESSAGE, null, options, options[0]);
```

```
            if (choice == null || choice.equals("Quit")) {
                // User closed the dialog or selected "Quit", exit the loop
                break;
            }

            if (choice.equals("Add Task")) {
                try {
                    int countTask = Integer.parseInt(JOptionPane.showInputDialog("How many tasks
do you want to add?"));
                    task.addTask(countTask);
                    totalHours += task.returnTotalHours();
                } catch (NumberFormatException e) {
                    JOptionPane.showMessageDialog(null, "Invalid input. Please enter a valid
number.");
                }
            } else if (choice.equals("Show Report")) {
                ShowReport show = new ShowReport(task.getDeveloper(), task.getTaskNames(),
task.getTaskID(),
                        task.getStatus(), task.getDuration());
                show.showReport();
            } else if (choice.equals("Search for Task")) {
                String taskName = JOptionPane.showInputDialog("Please enter the task name you
want to search:");
```

```
            String searchResult = task.searchTask(taskName);
            JOptionPane.showMessageDialog(null, searchResult);
        } else if (choice.equals("Delete Task")) {
            String taskDel = JOptionPane.showInputDialog(null, "Which task do you wish to
delete:");
            String deleteResult = task.deleteTask(taskDel);
            JOptionPane.showMessageDialog(null, deleteResult);
        } else if (choice.equals("Display Developer with Longest Duration")) {
            String longestResult = task.displayLongest();
            JOptionPane.showMessageDialog(null, longestResult);
        } else if (choice.equals("Search Task Assigned to Developer")) {
            String dev = JOptionPane.showInputDialog(null, "Please enter the developer
details:");
            String searchDevResult = task.searchDeveloper(dev);
            JOptionPane.showMessageDialog(null, searchDevResult);
        } else if (choice.equals("Mark Task as Done")) {
            String taskNameDone = JOptionPane.showInputDialog(null, "Which task do you
want to mark as Done?");
            task.updateTaskStatus(taskNameDone, "Done");
            JOptionPane.showMessageDialog(null, "Task \"" + taskNameDone + "\" marked as
Done.");
        }
    }
```

```
        } catch (Exception e) {

            JOptionPane.showMessageDialog(null, "An error occurred.", "Error",
JOptionPane.ERROR_MESSAGE);

        }

    }



    public static void main(String[] args) {

        TaskManagementSystem system = new TaskManagementSystem();

        system.start();

    }

}
```

## LOGIN Class

```
/*

 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change
this license

 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template

 */

package javaapp2;

import javax.swing.JOptionPane;

/**

 *

 * @author lab_services_student
```

```
*/
public class Login {
 private String username, password, name, surname;

  public String getUsername() {
     return this.username;
  }
  public String getPassword() {
     return this.password;
  }
  public String getName() {
     return this.name;
  }
  public String getSurname() {
     return this.surname;
  }
  public boolean setUsername(String username) {
     this.username = username;
     boolean usernameIsValid = checkUserName();

     if (usernameIsValid) {
```

```
        JOptionPane.showMessageDialog(null, "Username successfully captured.",
"Username", JOptionPane.INFORMATION_MESSAGE);

    } else {

        JOptionPane.showMessageDialog(null, "Username is not correctly formatted. Please
ensure that your username contains an underscore and is no more than 5 characters in
length.", "Invalid Username", JOptionPane.ERROR_MESSAGE);

        return false;

    }



    return true;

  }



  public boolean setPassword(String password) {

    this.password = password;

    boolean passwordIsValid = checkPasswordComplexity();



    if (passwordIsValid) {

        JOptionPane.showMessageDialog(null, "Password successfully captured.", "Password",
JOptionPane.INFORMATION_MESSAGE);

    } else {

        JOptionPane.showMessageDialog(null, "Password is not correctly formatted. Please
ensure that the password contains at least 8 characters.", "Invalid Password",
JOptionPane.ERROR_MESSAGE);

        return false;
```

```
        }


        return true;

    }


    public void setFirstName() {

        this.name = JOptionPane.showInputDialog(null, "Please enter your name:", "First Name",
JOptionPane.PLAIN_MESSAGE);

    }


    public void setLastName() {

        this.surname = JOptionPane.showInputDialog(null, "Please enter your surname:", "Last
Name", JOptionPane.PLAIN_MESSAGE);

    }


    public String registerUser() {

        String username = JOptionPane.showInputDialog(null, "Please enter your username:",
"Username", JOptionPane.PLAIN_MESSAGE);

        boolean isUsernameValid = setUsername(username);


        if (!isUsernameValid) {

            return "\nThe username is incorrectly formatted.";
```

```
        }


        String password = JOptionPane.showInputDialog(null, "Please enter your password:",
"Password", JOptionPane.PLAIN_MESSAGE);

        boolean isPasswordValid = setPassword(password);



        if (!isPasswordValid) {

            return "\nThe password does not meet the complexity requirements.";

        }

        return "\nThe username and password meet the requirements, and the user has been
registered successfully.";

    }

    public boolean checkUserName() {

        return this.username.contains("_") && this.username.length() <= 5;

    }

    public boolean checkPasswordComplexity() {

        return this.password.length() >= 8;

    }

    public boolean getLoginDetails() {

        String userName = JOptionPane.showInputDialog(null, "Username:", "Login",
JOptionPane.PLAIN_MESSAGE);

        String userPassword = JOptionPane.showInputDialog(null, "Password:", "Login",
JOptionPane.PLAIN_MESSAGE);
```

```
        boolean loginStatus = loginUser(userName, userPassword);

        if (loginStatus) {

            JOptionPane.showMessageDialog(null, "\nWelcome " + this.username + " " +
this.surname + ". It's great to see you again.", "Login Successful",
JOptionPane.INFORMATION_MESSAGE);

        } else {

            JOptionPane.showMessageDialog(null, "\nUsername or Password is incorrect. Please
try again.", "Login Failed", JOptionPane.ERROR_MESSAGE);

        }

        return loginStatus;

    }



    public boolean loginUser(String userName, String userPassword) {

        return this.username.equals(userName) && this.password.equals(userPassword);

    }

}
```

## Show Report Class

```
/*

 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change
this license

 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template

 */
```

```
package javaapp2;

import javax.swing.JOptionPane;

import java.util.List;

public class ShowReport {

    private List<String> developer;

    private List<String> taskNames;

    private List<String> taskID;

    private List<String> status;

    private List<Integer> duration;



    public ShowReport(List<String> developer, List<String> taskNames, List<String> taskID,
List<String> status, List<Integer> duration) {

        this.developer = developer;

        this.taskNames = taskNames;

        this.taskID = taskID;

        this.status = status;

        this.duration = duration;

    }



    public void showReport() {

        StringBuilder report = new StringBuilder("Task Report:\n\n");
```

```
    for (int i = 0; i < taskNames.size(); i++) {
        report.append("Task Name: ").append(taskNames.get(i)).append("\n");
        report.append("Task ID: ").append(taskID.get(i)).append("\n");
        report.append("Developer: ").append(developer.get(i)).append("\n");
        report.append("Duration: ").append(duration.get(i)).append(" hours\n");
        report.append("Status: ").append(status.get(i)).append("\n\n");
    }

    String[] options = { "All Tasks", "Done Tasks" };
    String choice = (String) JOptionPane.showInputDialog(null, "Which tasks would you like to
display?", "Task Report",
            JOptionPane.PLAIN_MESSAGE, null, options, options[0]);

    if (choice == null) {
        // User closed the dialog, return without displaying the report
        return;
    }

    if (choice.equals("Done Tasks")) {
        StringBuilder doneTasksReport = new StringBuilder("Done Tasks:\n\n");
        for (int i = 0; i < taskNames.size(); i++) {
```

```
            if (status.get(i).equalsIgnoreCase("done")) {
                doneTasksReport.append("Task Name: ").append(taskNames.get(i)).append("\n");
                doneTasksReport.append("Task ID: ").append(taskID.get(i)).append("\n");
                doneTasksReport.append("Developer: ").append(developer.get(i)).append("\n");
                doneTasksReport.append("Duration: ").append(duration.get(i)).append(" hours\n");
                doneTasksReport.append("Status: ").append(status.get(i)).append("\n\n");
            }
        }
        JOptionPane.showMessageDialog(null, doneTasksReport.toString());
    } else {
        JOptionPane.showMessageDialog(null, report.toString());
    }
  }
}
```