JULY 7, 2023

# PORTFOLIO OF EVIDENCE

PROG 5121

ST10380788
[COMPANY NAME]
[Company address]

# Task 1 - Registration and Login Feature:

## LOGIN CLASS:

```
import javax.swing.JOptionPane;
public class Login
{

    // Method to check if username meets the requirement
    public boolean checkUserName(String username)
    {
        if (username == null)
        {
            // User cancelled the input
            return false;
        }
        if (username.length() == 5 && username.contains("_"))
        {
            return true;
        } else {
            return false;
        }
    }

    // Method to check if password meets the requirement
    public boolean checkPasswordComplexity(String password)
    {
        if (password == null)
        {
            // User cancelled the input
            return false;
        }
        boolean hasNumber = false;
```

```
    boolean hasCapital = false;
    boolean hasSpecial = false;
    for (int i = 0; i < password.length(); i++)
    {
        char c = password.charAt(i);
        if (Character.isDigit(c))
        {
            hasNumber = true;
        } else if (Character.isUpperCase(c))
        {
            hasCapital = true;
        } else if (!Character.isLetterOrDigit(c))
        {
            hasSpecial = true;
        }
    }
    if (password.length() >= 8 && hasNumber && hasCapital && hasSpecial)
    {
        return true;
    } else
    {
        return false;
    }
}

// Method to register user and return appropriate message
public String registerUser(String username, String password)
{
    if (!checkUserName(username))
    {
```

```
            return "Error: Username is not correctly formatted. Please ensure that your username contains an " +
                    "underscore and is no more than 5 characters in length.";
        }
        if (!checkPasswordComplexity(password))
        {
            return "Error: Password is not correctly formatted. Please ensure that the password contains at least 8 " +
                    "characters, a capital letter, a number, and a special character.";
        }
        return "Success: You have been registered.";
    }

    // Method to login user and return appropriate message
     public boolean loginUser()
     {
         boolean username = true;
         boolean password = true;
        String inputUsername = JOptionPane.showInputDialog("Enter username:");
        String inputPassword = JOptionPane.showInputDialog("Enter password:");

        if (inputUsername.equals(username) && inputPassword.equals(password))
        {
            return true;
        } else
        {
            return false;
        }
    }
     // Method to return appropriate message for login status
     public String returnLoginStatus()
     {
```

```
        boolean isSuccess = loginUser();
        String firstName = "";
        String lastName = "";
        String login = "";
        if (isSuccess)
        {
            return "Welcome " + firstName + " " + lastName + " it is great to see you again.";
        } else
        {
            return "Username or password incorrect, please try again.";
        }

    }
}import javax.swing.JOptionPane;
public class Login
{

    // Method to check if username meets the requirement
    public boolean checkUserName(String username)
    {
        if (username == null)
        {
            // User cancelled the input
            return false;
        }
        if (username.length() == 5 && username.contains("_"))
        {
            return true;
        } else {
            return false;
        }
```

```
}

// Method to check if password meets the requirement
public boolean checkPasswordComplexity(String password)
{
    if (password == null)
    {
        // User cancelled the input
        return false;
    }
    boolean hasNumber = false;
    boolean hasCapital = false;
    boolean hasSpecial = false;
    for (int i = 0; i < password.length(); i++)
    {
        char c = password.charAt(i);
        if (Character.isDigit(c))
        {
            hasNumber = true;
        } else if (Character.isUpperCase(c))
        {
            hasCapital = true;
        } else if (!Character.isLetterOrDigit(c))
        {
            hasSpecial = true;
        }
    }
    if (password.length() >= 8 && hasNumber && hasCapital && hasSpecial)
    {
        return true;
    } else
```

```
        {
            return false;
        }
    }

    // Method to register user and return appropriate message
    public String registerUser(String username, String password)
    {
        if (!checkUserName(username))
        {
            return "Error: Username is not correctly formatted. Please ensure that your username contains an " +
                    "underscore and is no more than 5 characters in length.";
        }
        if (!checkPasswordComplexity(password))
        {
            return "Error: Password is not correctly formatted. Please ensure that the password contains at least 8 " +
                    "characters, a capital letter, a number, and a special character.";
        }
        return "Success: You have been registered.";
    }

    // Method to login user and return appropriate message
     public boolean loginUser()
     {
         boolean username = true;
         boolean password = true;
        String inputUsername = JOptionPane.showInputDialog("Enter username:");
        String inputPassword = JOptionPane.showInputDialog("Enter password:");

        if (inputUsername.equals(username) && inputPassword.equals(password))
```

```
        {
            return true;
        } else
        {
            return false;
        }
    }
    // Method to return appropriate message for login status
    public String returnLoginStatus()
    {
        boolean isSuccess = loginUser();
        String firstName = "";
        String lastName = "";
        String login = "";
        if (isSuccess)
        {
            return "Welcome " + firstName + " " + lastName + " it is great to see you again.";
        } else
        {
            return "Username or password incorrect, please try again.";
        }

    }
}
```

## AUTHENTICATION CLASS:

```
import javax.swing.JOptionPane;
public class UserAuthentication
{
    String name;
    String surname;
```

```
String username;
String password;

public boolean checkUserName(String user_name)
{
    boolean valid = false;
    boolean underScore =  false;
    //Validate Username
    for (int i = 0; i < user_name.length(); i++)
    {
        if (user_name.charAt(i) == '_')
        {
            underScore = true;
        }
        if(underScore && i <= 4)
        {
            valid = true;
        }
    }return valid;
}

//Password Validation
public boolean checkPasswordComplexity(String pass_word)
{

    boolean length8 = false;
    boolean specialCharacter = false;
    boolean number = false;
    boolean capitalLetter = false;
    boolean valid = false;
```

```
if(pass_word.length() >= 8)
{
    length8 = true;
}

for(int i = 0; i < pass_word.length()-1; i++)
{
    if(pass_word.charAt(i) == '@' || pass_word.charAt(i) == '#' ||
            pass_word.charAt(i) == '&' || pass_word.charAt(i) == '%' ||
            pass_word.charAt(i) == '$' || pass_word.charAt(i) == '*' ||
            pass_word.charAt(i) == '!' || pass_word.charAt(i) == '^'){
        specialCharacter = true;
    }
}

for(int i = 0; i < pass_word.length()-1; i++)
{
    if(pass_word.charAt(i) >= 0 || pass_word.charAt(i) <= 9){
        number = true;
    }
}

for(int i = 0; i < pass_word.length()-1; i++){
    if(pass_word.charAt(i) >= 'A' && pass_word.charAt(i) <= 'Z'){
        capitalLetter = true;
    }
}

if(length8 && specialCharacter && number && capitalLetter){
    valid = true;
}return valid;
```

```
    }

    //User registeration
    public String registerUser(String n_ame, String sur_name, String user_name, String
pass_word) {
        String message = "";
        int valid = 0;

        if (checkUserName(user_name)) {
            JOptionPane.showMessageDialog(null,"correct username");
            valid++;
        }
        else{
            JOptionPane.showMessageDialog
                    (null,"incorrect username");
        }

        if (checkPasswordComplexity(pass_word)) {
            JOptionPane.showMessageDialog(null,"correct password");
            valid++;
        }
        else{
            JOptionPane.showMessageDialog
                    (null,"incorrect password");
        }

        if (valid == 2) {
            message = "registeration succesful";
            name = n_ame;
            surname = sur_name;
            username = user_name;
```

```
            password = pass_word;
        }
        else {
            message = "registration unsuccesful";
        }
        return message;


    }


    //User login
    public boolean loginUser(String n_ame, String sur_name, String user_name, String
pass_word){
        boolean valid = false;
        if (name== n_ame && surname == sur_name && username == user_name &&
password == pass_word){
            valid = true;
        }
        return valid;
    }


    //User login status
    public String returnLoginStatus(String n_ame, String sur_name, String user_name, String
pass_word){
        String message = null;
        if (loginUser(n_ame, sur_name, user_name, pass_word)){
            message = "login successful!";
        }
        else{
            message = "login unsuccessful";
        }
        return message;
    }
```

```
public static void main(String[] args)
{
    Login login = new Login();

    // example usage of methods
    String usernameInput = "";
    String passwordInput = "";
    boolean validUsername = false;
    boolean validPassword = false;

    while (!validUsername)
    {
        usernameInput = JOptionPane.showInputDialog("Enter your username (5 characters
long and contains an underscore):");
        validUsername = login.checkUserName(usernameInput);
        if (!validUsername)
        {
            JOptionPane.showMessageDialog(null, "Username is not correctly formatted.");
        }
    }

    while (!validPassword)
    {
        passwordInput = JOptionPane.showInputDialog("Enter your password (at least 8
characters long, containing a capital letter, a number, and a special character):");
        validPassword = login.checkPasswordComplexity(passwordInput);
        if (!validPassword)
        {
            JOptionPane.showMessageDialog(null, "Password is not correctly formatted.");
        }
    }
```

```
    }
}
```

## MAIN DRIVER CLASS:

```
import javax.swing.JOptionPane;

public class MainDriver
{
  public static void main(String[] args)
    {
        String username = null;
        String password = null;
        String firstName = null;
        String lastName = null;

        // create account
        boolean accountCreated = false;
        while (!accountCreated)
        {
            username = JOptionPane.showInputDialog("Enter your username:");
            if (checkUsername(username))
            {
                password = JOptionPane.showInputDialog("Enter your password:");
                if (checkPassword(password))
                {
                    firstName = JOptionPane.showInputDialog("Enter your first name:");
                    lastName = JOptionPane.showInputDialog("Enter your last name:");
                    JOptionPane.showMessageDialog(null, "Account created successfully. You can
now log in using your username and password.");
                    accountCreated = true;
                } else
```

```
            {
                JOptionPane.showMessageDialog(null, "Password is not correctly formatted,
please ensure that the password contains at least 8 characters, a capital letter, a number and
a special character.");
            }
        } 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.");
        }
    }

    // login
    boolean loggedIn = false;
    while (!loggedIn)
    {
        String enteredUsername = JOptionPane.showInputDialog("Enter your username:");
        String enteredPassword = JOptionPane.showInputDialog("Enter your password:");
        if (enteredUsername.equals(username) && enteredPassword.equals(password))
        {
            JOptionPane.showMessageDialog(null, "Login successful. Welcome " + firstName +
" " + lastName + ".");
            loggedIn = true;
        } else
        {
            JOptionPane.showMessageDialog(null, "Login failed. Please check your username
and password and try again.");
        }
    }
  }

  public static boolean checkUsername(String username)
```

```
{
    return (username.length() <= 5 && username.contains("_"));
}

public static boolean checkPassword(String password)
{
    boolean hasLength = password.length() >= 8;
    boolean hasUpperCase = !password.equals(password.toLowerCase());
    boolean hasDigit = password.matches(".*\\d.*");
    boolean hasSpecialChar = !password.matches("[a-zA-Z0-9 ]*");
    return hasLength && hasUpperCase && hasDigit && hasSpecialChar;
}
{
    Login login = new Login();

    // example usage of methods
    String usernameInput = "";
    String passwordInput = "";
    boolean validUsername = false;
    boolean validPassword = false;

    while (!validUsername)
    {
        usernameInput = JOptionPane.showInputDialog("Enter your username (5 characters
long and contains an underscore):");
        validUsername = login.checkUserName(usernameInput);
        if (!validUsername)
        {
            JOptionPane.showMessageDialog(null, "Username is not correctly formatted.");
        }
    }
```

```
        while (!validPassword)
        {
            passwordInput = JOptionPane.showInputDialog("Enter your password (at least 8
characters long, containing a capital letter, a number, and a special character):");
            validPassword = login.checkPasswordComplexity(passwordInput);
            if (!validPassword)
            {
                JOptionPane.showMessageDialog(null, "Password is not correctly formatted.");
            }
        }

        String registrationMessage = login.registerUser(usernameInput, passwordInput);
        JOptionPane.showMessageDialog(null, registrationMessage);

        String loginUsername = JOptionPane.showInputDialog("Enter your username:");
        String loginPassword = JOptionPane.showInputDialog("Enter your password:");
    }
}
```

# Task 2 – Adding Tasks Features:

## EASY KANBAN CLASS:

```
import javax.swing.JOptionPane;
public class EasyKanban
{
    public static void main(String [] args)
    {
        //login feature
        String username = " ";
        String password = " ";
        while(true)
        {
            username = JOptionPane.showInputDialog("Enter your username:");
            if (username != null && username.matches("^[A-Za-z0-9_]{1,5}$"))
            {
                break;
            }
            else
            {
                JOptionPane.showMessageDialog(null, "Incorrect username, please try again");
            }
        }
        while(true)
        {
            password = JOptionPane.showInputDialog("Enter your password:");
            if (password != null && password.matches("^(?=.*[A-Z])(?=.*[0-
9])(?=.*[!@#$%^&*()_+\\[\\]{};':\"\\\\|,.<>\\/?]).{8,}$"))
            {
                break;
            }
            else
```

```
        {
            JOptionPane.showMessageDialog(null, "Incorrect password, please try again");
        }
    }
    JOptionPane.showMessageDialog(null, "Welcome to Easy Kanban");

    //display menu options
    int option = 0;
    while (option != 3)
    {
        option = Integer.parseInt(JOptionPane.showInputDialog("Choose an option below:\n1. Add tasks\n2. Show report\n3. Quit"));
        switch (option)
        {
            case 1:
                //get number of tasks
                int numTasks = Integer.parseInt(JOptionPane.showInputDialog("Enter number of tasks:"));

                //create an array of task objects
                Task[] tasks = new Task[numTasks];

                //loop for task details
                for (int i = 0; i < numTasks; i++)
                {
                    //task name
                    String taskName = JOptionPane.showInputDialog("Enter task name:");

                    //task number
                    int taskNumber = i;

                    //task description
```

```
String taskDescription = " ";
while (true)
{
    taskDescription = JOptionPane.showInputDialog("Enter task description:");
    if (taskDescription != null && taskDescription.length() <= 50)
    {
        JOptionPane.showMessageDialog(null, "Task successfully captured");
        break;
    }
    else
    {
        JOptionPane.showMessageDialog(null, "Please enter a task description of
50 characters or less");
    }
}

//task developer details
String developerDetails = JOptionPane.showInputDialog("Enter task
developers first and last name:");

//task duration
double taskDuration =
Double.parseDouble(JOptionPane.showInputDialog("Enter task duration in hours:"));

//task ID
String taskID = taskName.substring(0, 2).toUpperCase() + ":" + taskNumber +
":" +
developerDetails.substring(developerDetails.length() - 3).toUpperCase();

//task status
String [] taskStatusOptions = {"ToDo", "Doing", "Done"};
int taskStatus = JOptionPane.showOptionDialog(null,"Select task status:",
"Task Status",
```

```
                JOptionPane.DEFAULT_OPTION, JOptionPane.PLAIN_MESSAGE, null,
taskStatusOptions, taskStatusOptions[0]);


                //create task object and add to array
                tasks[i] = new Task(taskName, taskNumber, taskDescription, developerDetails,
taskDuration, taskID, taskStatus);


                //display task details
                JOptionPane.showMessageDialog(null, "Task Status: " +
tasks[i].getStatusString() +
                "\nDeveloper Details" + tasks[i].getDeveloperDetails() + "\nTask Number: " +
tasks[i].getTaskNumber() +
                "\nTask Name: " + tasks[i].getTaskName() + "\nTask Description: " +
tasks[i].getTaskDescription() +
                "\nTaskID: " + tasks[i].getTaskID() + "\nDuration: " + tasks[i].getTaskDuration() +
" hours");
            }
            //total number of hours
            double totalHours = 0;
            for (int i = 0; i < numTasks; i++)
            {
                totalHours += tasks[i].getTaskDuration();
            }
            JOptionPane.showMessageDialog(null, "Total number of hours " + totalHours);
            break;
        case 2:
            JOptionPane.showMessageDialog(null, "Coming Soon");
            break;
        case 3:
            JOptionPane.showMessageDialog(null, "Thanks for using Easy Kanban");
            System.exit(0);
            break;
        default:
```

```
                JOptionPane.showMessageDialog(null, "Invalid option, please try again");
                break;
            }
        }
    }
}
```

## TASK CLASS:

```
public class Task
{
    private String taskName;
    private int taskNumber;
    private String taskDescription;
    private String developerDetails;
    private double taskDuration;
    private String taskID;
    private int taskStatus;

    public Task(String taskName, int taskNumber, String taskDescription, String
developerDetails, double taskDuration,
    String taskID, int taskStatus)
    {
      this.taskName = taskName;
      this.taskNumber = taskNumber;
      this.taskDescription = taskDescription;
      this.developerDetails = developerDetails;
      this.taskDuration = taskDuration;
      this.taskID = taskID;
      this.taskStatus = taskStatus;
    }
```

```
public String getTaskName()
{
    return taskName;
}

public int getTaskNumber()
{
    return taskNumber;
}

public String getTaskDescription()
{
    return taskDescription;
}

public String getDeveloperDetails()
{
    return developerDetails;
}

public double getTaskDuration()
{
    return taskDuration;
}

public String getTaskID()
{
    return taskID;
}

public int getTaskStatus()
```

```
{
    return taskStatus;
}

public String getStatusString()
{
    switch (taskStatus)
    {
        case 0:
            return "To Do";
        case 1:
            return "Doing";
        case 2:
            return "Done";
        default:
            return " ";
    }
}

public boolean checkTaskDescription()
{
    return taskDescription != null && taskDescription.length() <= 50;
}

public String createTaskID()
{
    return taskName.substring(0, 2).toUpperCase() + ":" + taskNumber + ":" +
    developerDetails.substring(developerDetails.length() - 3).toUpperCase();
}

public String printTaskDetails()
```

```
    {
        return "Task Status: " + getStatusString() + "\nDeveloper Details: " + developerDetails +
"\nTask Number: " +
        taskNumber + "\nTask Name: " + taskName + "\nTask Description: " + taskDescription +
"\nTask ID: " + taskID +
        "\nDuration: " + taskDuration + " hours";
    }

    public static int returnTotalHours(Task[] tasks)
    {
        int totalHours = 0;
        for (Task task : tasks)
        {
            totalHours += task.getTaskDuration();
        }
        return totalHours;
    }
}
```

# Task 3 - Store Data and Display Task Report

## ARRAY CLASS:

```
import javax.swing.JOptionPane;

import java.util.Arrays;

public class Array

{

   public static void main(String [] args)

   {

      String[] developer;

      String[] taskName;

      int[] taskID;

      int[] taskDuration;

      String[] taskStatus;


      //1. Populate arrays with test data given

      developer = new String[]{"Mike Smith", "Edward Harrison", "Samantha Paulson",
"Glenda Oberholzer"};

      taskName = new String[]{"Create Login", "Create Add Features", "Create Reports", "Add
Arrays"};

      taskID = new int[]{1, 2, 3, 4};

      taskDuration = new int[]{5, 8, 2, 11};

      taskStatus = new String[]{"To Do", "Doing", "Done", "To Do"};


      //2a. Display the Developer, Task Names and Task Duration for all tasks with the status of
"Done"

      for (int i = 0; i < taskStatus.length; i++)

      {

         if (taskStatus[i].equals("Done"))

         {
```

```
            JOptionPane.showMessageDialog(null, "Developer: " + developer[i] + "\n" + "Task
Name: " + taskName[i] +
            "\n"+ "Task Duration: " + taskDuration[i]);
        }
    }
    //2b. Display the Developer and Duration of the class with the longest duration
    int maxDuration = 0;
    int maxDurationIndex = 0;
    for (int i = 0; i < taskDuration.length; i++)
    {
        if (taskDuration[i] > maxDuration)
        {
            maxDurationIndex = i;
            maxDuration = taskDuration[i];
        }
    }
    JOptionPane.showMessageDialog(null,"Developer with longest duration: " +
developer[maxDurationIndex] + "\n" +
    "Task Duration: " + taskDuration[maxDurationIndex]);
    //2c. Search for a task with a Task Name and display the Task Name, Developer and
Task Status
    String searchTaskName = JOptionPane.showInputDialog("Search Task Name: ");
    for (int i = 0; i < taskName.length; i++)
    {
        if (taskName[i].equals(searchTaskName))
        {
            JOptionPane.showMessageDialog(null, "Task Name: " + taskName[i] + "\n" +
"Developer: " + developer[i] +
            "\n" +"Task Status: " + taskStatus[i]);
        }
    }
    //2d. Search for all tasks assigned to a developer and display the Task Name and Task
Status
```

```
        String searchDeveloper = JOptionPane.showInputDialog("Search Task Developer Name: ");
        for (int i = 0; i < developer.length; i++)
        {
            if (developer[i].equals(searchDeveloper))
            {
                JOptionPane.showMessageDialog(null, "Task Name: " + taskName[i] + "\n" + "Task Status: " + taskStatus[i]);
            }
        }
        //2e. Delete a task using the Task Name
        String deleteTaskName = JOptionPane.showInputDialog("Enter the Task Name you wish to delete: ");
        int deleteIndex = -1;
        for (int i = 0; i < taskName.length; i++)
        {
            if (taskName[i].equals(deleteTaskName))
            {
                deleteIndex = i;
            }
        }
        if (deleteIndex != -1)
        {
            for (int i = deleteIndex; i < taskName.length - 1; i++)
            {
                taskName[i] = taskName[i + 1];
                developer[i] = developer[i + 1];
                taskDuration[i] = taskDuration[i + 1];
                taskStatus[i] = taskStatus[i + 1];
            }
            taskName = Arrays.copyOf(taskName, taskName.length - 1);
            developer = Arrays.copyOf(developer, developer.length - 1);
```

```
            taskDuration = Arrays.copyOf(taskDuration, taskDuration.length - 1);
            taskStatus = Arrays.copyOf(taskStatus, taskStatus.length - 1);


            JOptionPane.showMessageDialog(null, "Task " + deleteTaskName + " deleted
successfully");
        }
        else
        {
            JOptionPane.showMessageDialog(null, "Task not found");
        }
        //2f. Display a report that lists the full details of all captured tasks
        for (int i = 0; i < taskID.length; i++)
        {
            JOptionPane.showMessageDialog(null, "------------ Task Report: ------------\n" + "Task
ID: " + taskID[i] + "\n"
            + "Developer: " + developer[i] + "\n" + "Task Name: " + taskName[i] + "\n" + "Task
Duration: " + taskID[i] + "\n"
            + "Task Status: " + taskStatus[i] +"\n"+ "-----------------------------------------------\n");
        }
    }
}
```