Alejandro Ramirez & Thomas Mosychuk, Jr.

CSC260

Final Project Draft

**Habitz First Draft**

**Main**

|  |
| --- |
| import java.util.LinkedList;  import java.util.Scanner;  public class Main {  public static void main(String[] args) {  LinkedList<Habit> habits = new LinkedList<>();  // Thread for running the GUI  Thread guiThread = new Thread(() -> {  HabitsGUI gui = new HabitsGUI(habits);  gui.setVisible(true);  });  guiThread.start();  Scanner input = new Scanner(System.in);  int selection = -1;  while (selection != 0) {  System.out.println("Please enter a menu option.\n 1. Create a habit \n 2. Print all the habits in the list \n 3. Complete a habit.\n 0. Exit");  selection = input.nextInt();  input.nextLine(); // Consume newline character after reading int  if (selection == 1) {  createHabit(habits);  } else if (selection == 2) {  printHabits(habits);  } else if (selection == 3) {  completeHabit(habits);  }  }  }  public static void createHabit(LinkedList<Habit> habits)  {  Scanner input = new Scanner(System.in);  String name;  String description;  System.out.println("Please enter the name of the habit you'd like to create.");  name = input.nextLine();  System.out.println("Please enter a brief description of your habit.");  description = input.nextLine();  //creates the habit given the user inputs, and then adds it to the linked list provided.  Habit currHabit = new Habit(name, description);  habits.add(currHabit);  }  public static void printHabits(LinkedList<Habit> habits)  {  for(Habit e : habits){  System.out.print("\n" + (habits.indexOf(e)+ 1) + ". "+ e.habitName + "; Completion status: ");  if(e.completed){  System.out.print("Completed.\n");  }  else{  System.out.print("Not completed.\n");  }  }  }  public static void completeHabit(LinkedList<Habit> habits) {  Scanner input = new Scanner(System.in);  int exit = 0;  while (exit != 1) {  System.out.println("Please enter the name or priority of your habit");  String stringSearch = input.nextLine();  boolean found = false;  for (Habit e : habits) {  if (e.habitName.equals(stringSearch)) {  e.complete();  found = true;  exit = 1;  break; // Exit loop if habit is found and completed  }  }  if (!found) {  System.out.println("Could not find the habit you were looking for. Please try again.");  }  }  }  } |

**Habit**

|  |
| --- |
| public class Habit {  String habitName;  String description;  boolean completed = false;  //above are the qualities that the habit should have so far:  // {name of habit, description of habit, whether it's been done}  public Habit(String habitName, String description)  {  this.habitName = habitName;  this.description = description;  }  public String getHabitName(){  return habitName;  }  public boolean isCompleted()  {  return completed;  }  public void setCompleted(boolean completed)  {  this.completed = completed;  }  public String getDescription(){  return description;  }  public void complete(){  completed = true;  }  } |

**GUI Interface**

|  |
| --- |
| import javax.swing.\*;  import java.awt.\*;  import java.awt.event.ActionEvent;  import java.awt.event.ActionListener;  import java.util.LinkedList;  public class HabitsGUI extends JFrame {  private LinkedList<Habit> habitsList;  private JPanel habitsPanel; // Panel to hold habit entries  public HabitsGUI(LinkedList<Habit> habits) {  this.habitsList = habits;  setTitle("Habits App");  setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);  setSize(400, 300);  setLocationRelativeTo(null);  JPanel mainPanel = new JPanel();  mainPanel.setLayout(new BoxLayout(mainPanel, BoxLayout.Y\_AXIS));  // 'Add Habit' Button  JButton addButton = new JButton("Add Habit");  addButton.setAlignmentX(Component.CENTER\_ALIGNMENT);  addButton.addActionListener(e -> {  String name = JOptionPane.showInputDialog("Enter habit name:");  String description = JOptionPane.showInputDialog("Enter habit description:");  if (name != null && description != null && !name.isEmpty() && !description.isEmpty()) {  habitsList.add(new Habit(name, description));  addHabitToPanel(habitsList.getLast());  }  });  // 'Uncheck All' Checkbox  JButton uncheckAllButton = new JButton("Reset all Habitz");  uncheckAllButton.setAlignmentX(Component.CENTER\_ALIGNMENT);  uncheckAllButton.addActionListener(e -> {  for (Habit habit : habitsList) {  habit.setCompleted(false);  }  refreshHabitsPanel(); // Refresh the panel to reflect changes  });  mainPanel.add(Box.createVerticalGlue());  mainPanel.add(addButton);  mainPanel.add(uncheckAllButton);  mainPanel.add(Box.createVerticalGlue());  habitsPanel = new JPanel();  habitsPanel.setLayout(new BoxLayout(habitsPanel, BoxLayout.Y\_AXIS));  JScrollPane scrollPane = new JScrollPane(habitsPanel);  scrollPane.setVerticalScrollBarPolicy(JScrollPane.VERTICAL\_SCROLLBAR\_ALWAYS);  scrollPane.setPreferredSize(new Dimension(350, 250));  mainPanel.add(scrollPane);  add(mainPanel);  setVisible(true);  }  private void addHabitToPanel(Habit habit) {  JPanel habitPanel = new JPanel();  habitPanel.setLayout(new FlowLayout(FlowLayout.LEFT));  JLabel nameLabel = new JLabel(habit.getHabitName() + " - " + habit.getDescription());  JCheckBox checkBox = new JCheckBox();  checkBox.setSelected(habit.isCompleted());  checkBox.addActionListener(e -> habit.setCompleted(checkBox.isSelected()));  habitPanel.add(checkBox);  habitPanel.add(nameLabel);  habitsPanel.add(habitPanel);  habitsPanel.revalidate();  habitsPanel.repaint();  }  private void refreshHabitsPanel() {  habitsPanel.removeAll();  for (Habit habit : habitsList) {  addHabitToPanel(habit);  }  habitsPanel.revalidate();  habitsPanel.repaint();  }  public static void main(String[] args) {  SwingUtilities.invokeLater(() -> {  LinkedList<Habit> habits = new LinkedList<>();  new HabitsGUI(habits);  });  }  } |

**UML**

|  |
| --- |
| * **Class Name:** Habit * **Attributes:**   + **-habitName: String**   + **-description: String**   + **-completed: boolean = false** * **Methods:**   + **+Habit(habitName: String, description: String)**   + **+getHabitName(): String**   + **+getDescription(): String**   + **+isCompleted(): boolean**   + **+setCompleted(completed: boolean): void**   + **+complete(): void** |