**USER INTERFACE**

The **user interface**, in the industrial design field of human–machine interaction, is the space where interaction between humans and machines occurs. The goal of this interaction is effective operation and control of the machine on the user's end, and feedback from the machine, which aids the operator in making operational decisions.

In computing, **graphical user interface** (**GUI)**, sometimes pronounced "gooey" or "gwee") is a type of user interface that allows users to interact with electronic devices through graphical icons and visual indicators such as secondary notation, as opposed to text-based interfaces, typed command labels or text navigation. GUIs were introduced in reaction to the perceived steep learning curve of command-line interfaces (CLI), which require commands to be typed on the keyboard.

**Design**

**GUI** of Integrated text editor (ITE) is designed using various swing containers and components of java.

GUI of ITE contains a text area, various buttons, text fields, menu bar, tool bar, icons. Along with this, the GUI also pops up various pop menus and dialogs on user actions.

1. **Text area**: Allows user to enter text.
2. **Menu bar:** For save and edit options.
3. **Tool bar:** For cut, copy and paste options.
4. **Find text field**: For dynamically find a character or a word in the text.
5. **Options button**: For various editor features.
6. **Advance button**: For replacing a word or a character with another.

**Text area**

Designed using JTextArea of swing package. A JTextArea is a multi-line area that displays plain text. It is intended to be a lightweight component that provides source compatibility with the java.awt.TextArea class where it can reasonably do so.

**public class JTextArea extends JComponent**

This component has capabilities not found in the java.awt.TextArea class. The superclass should be consulted for additional capabilities. The java.awt.TextArea internally handles scrolling. JTextArea is different in that it doesn't manage scrolling, but implements the swing Scrollable interface. This allows it to be placed inside a JScrollPane if scrolling behavior is desired, and used directly if scrolling is not desired. The text entered by the user is obtained using appropriate listeners such as **KeyListener**.

Every swing component is added to a frame (JFrame), a Frame is a top-level window with a title and a border. The size of the frame includes any area designated for the border. A frame, implemented as an instance of the JFrame class, is a window that has decorations such as a border, a title, and supports button components that close or iconify the window. Applications with a GUI usually include at least one frame.

Features incorporated in text area of ITE;

* Word wrapping at the end of the line.
* Automatic scroll.
* Automatic conversion of first letter of a sentence to capital letter.

**Menu Bar**

Designed using JMenuBar of swing package. JMenu objects are added to the menu bar to construct a menu. When the user selects a JMenu object, its associated JPopupMenu is displayed, allowing the user to select one of the JMenuItems on it.

**public class JMenuBar extends JComponent**

Menu bar of ITE has following features;

* **File**: For opening a new text file (new), saving the current file (save), saving the current file with a different name (save as...).
* **Edit:** For editing the current text file like cut (Cut), copy (Copy) and paste (Paste) options.

The option selected from the menu is recognized using **ItemListener.** Options (save, save as, open ) selected in the menu causes appropriate dialogs to pop up at the centre of the screen using **showXXXdialog ()** function of **JFileChooser.**

**JFileChooser:** FileChooser provides a simple mechanism for the user to choose a file. Provides a dialog to navigate a file system. File choosers provide a GUI for navigating the file system, and then either choosing a file or directory from a list, or entering the name of a file or directory.

**public class JFileChooser extends JComponent implements Acessible**

**Tool Bar**

Designed using **JToolBar** of swing package. **JToolBar** provides a component that is useful for displaying commonly used Actions or controls. A**JToolBar** is a container that groups several components — usually buttons with icons — into a row or column. Often, tool bars provide easy access to functionality that is also in menus.

**public class JToolBar extends JComponent**

Tool bar of ITE has following features;

* **New:** For opening new text file.
* **Open:** For an existing opening a text file.
* **Save:** For saving the current file.
* **Cut:** A button for removing the text from text area.
* **Copy:** A button for copying the specified text.
* **Paste:** A button for pasting the copied text into text area.

These features are associated with appropriate icons in the tool bar.

**Find text field**

A text field is designed using JTextField of swing package. TextField is a lightweight component that allows the editing of a single line of text. JTextField is intended to be source-compatible with java.awt.TextField where it is reasonable to do so. This component has capabilities not found in the java.awt.TextField class. The superclass should be consulted for additional capabilities.

**public class JTextField extends JComponent**

The text entered in the field is obtained using **KeyListener**. The specified word is **found** in the text area and **highlighted** dynamically.

Also, the **Find** text field has a focus when user presses **Ctrl + f.**

**Options**

Option is a button designed using **JButton** of swing package. **JButton** is implementation of a "push" button. Buttons can be configured, and to some degree controlled, by Actions. Using an Action with a button has many benefits beyond directly configuring a button.

A JButton generates an action event, which can be listened by **ActionListener.**

Further, **Option** button raises a dialog which provides user to select various features;

* **Auto complete:** Completes long words automatically.
* **Spell check:** Checks for spelling mistakes and corrects it.
* **Context recognizer:** Recognizes the context of what user is trying to type.
* **Underline:** Underlines spelling errors.

A dialog is designed using **JDialog** of swing package, the main class for creating a dialog window. You can use this class to create a custom dialog, or invoke the many class methods in **JOptionPane** to create a variety of standard dialogs. **JDialog** by default is associated with buttons which when hit causes the dialog to disappear.

The dialog has various swing components for different features like **radio buttons, check boxes, text fields, labels and separators.**

* **Radio Button:** JRadioButton,
* **Check box:** JCheckBox, for editor features.

* **Separators:** JSeparator, for implementing divide lines.
* **Label:** JLabel, for static texts like headings.
* **Text field:** JTextField

These components in the dialog are set to certain default values initially. Dialog by default is associated with two buttons (OK and Cancel),

* **Cancel**: Restores the values of dialog components to default values.
* **Ok**: Sets the values according to user’s choice.

**Advance**

A button for providing a dialog, which allows user to enter a word and its substitute. Every occurrence of the specified word is replaced with the substitute.

Button is designed using **JButton**, which generates an action event. Action listener recognizes the event (button pressed) and raises a dialog box at the centre of the screen.

The dialog contains following components:

* **Find:** A text field (**JTextField**), which accepts user input (word or character) for finding the word or character in the text area. The text entered in obtained using key listener dynamically. Every occurrence of the specified word or character is highlighted dynamically.
* **Replace:** A text field (**JTextField**), which accepts the substitute. The word or character specified in the find field is replaced with the substitute. The text entered in obtained using key listener dynamically. Every occurrence of the word or character is replaced with the substitute.
* **Replace** (replace) **button:** A button, which initiates the replacing operation.

This dialog is associated with an” Ok “button which when hit causes the dialog to disappear.