slip23

```
import java.io.File;

public class s23q1 {

    public static void main(String[] args) {

        String filePath = "file4.txt";


        File file = new File(filePath);


        if (file.exists()) {

            if (file.isHidden()) {
                System.out.println("The file is hidden.");
            } else {
                System.out.println("The file is not hidden.");
                System.out.println("File path: " + file.getAbsolutePath());
            }
        } else {
            System.out.println("The specified file does not exist.");
        }
    }
}
```

```
import javax.swing.*;

import java.awt.*;

import java.awt.event.*;


public class FrameDesign extends JFrame implements ActionListener {

    JMenuBar menuBar;

    JMenu fileMenu, editMenu, searchMenu;

    JMenuItem undoItem, redoItem, cutItem, copyItem, pasteItem;


    public FrameDesign() {



        setTitle("Frame Design");

        setSize(400, 300);

        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        setLocationRelativeTo(null);


        menuBar = new JMenuBar();

        setJMenuBar(menuBar);


        fileMenu = new JMenu("File");

        menuBar.add(fileMenu);


        editMenu = new JMenu("Edit");

        menuBar.add(editMenu);
```

```
searchMenu = new JMenu("Search");

menuBar.add(searchMenu);


undoItem = new JMenuItem("Undo");

undoItem.addActionListener(this);

redoItem = new JMenuItem("Redo");

redoItem.addActionListener(this);

cutItem = new JMenuItem("Cut");

cutItem.addActionListener(this);

copyItem = new JMenuItem("Copy");

copyItem.addActionListener(this);

pasteItem = new JMenuItem("Paste");

pasteItem.addActionListener(this);


editMenu.add(undoItem);

editMenu.add(redoItem);

editMenu.add(cutItem);

editMenu.add(copyItem);

editMenu.add(pasteItem);


setLayout(new BorderLayout());


JPanel contentPane = new JPanel();
```

```
    contentPane.setBackground(Color.WHITE);
    add(contentPane, BorderLayout.CENTER);

    setVisible(true);
}


public void actionPerformed(ActionEvent e) {
    if (e.getSource() == undoItem) {

    } else if (e.getSource() == redoItem) {
      } else if (e.getSource() == cutItem) {

    } else if (e.getSource() == copyItem) {

    } else if (e.getSource() == pasteItem) {

    }
}

public static void main(String[] args) {
    SwingUtilities.invokeLater(new Runnable() {
        @Override
        public void run() {
            new FrameDesign();
        }
        });
    }
}
```

```
import tkinter as tk
from tkinter import font

class FontStyleChanger:
    def __init__(self, master):
        self.master = master
        self.master.title("Label Font Style Changer")

        # Label to display text
        self.label = tk.Label(master, text="Sample Text", font=("Arial", 12))
        self.label.pack(pady=10)

        # Entry for font name
        self.font_name_label = tk.Label(master, text="Font Name:")
        self.font_name_label.pack()
        self.font_name_entry = tk.Entry(master)
        self.font_name_entry.pack(pady=5)

        # Entry for font size
        self.font_size_label = tk.Label(master, text="Font Size:")
        self.font_size_label.pack()
        self.font_size_entry = tk.Entry(master)
        self.font_size_entry.pack(pady=5)

        # Checkbutton for bold style
        self.bold_var = tk.BooleanVar()
        self.bold_check = tk.Checkbutton(master, text="Bold", variable=self.bold_var)
        self.bold_check.pack(pady=5)

        # Button to apply changes
        self.apply_button = tk.Button(master, text="Apply Font Style",
command=self.apply_font_style)
        self.apply_button.pack(pady=20)

    def apply_font_style(self):
        """Apply the selected font style to the label."""
        font_name = self.font_name_entry.get() or "Arial"  # Default to Arial if empty
        try:
            font_size = int(self.font_size_entry.get())
        except ValueError:
            font_size = 12  # Default size if input is invalid

        # Determine if bold should be applied
        font_weight = 'bold' if self.bold_var.get() else 'normal'

        # Set the new font style to the label
        self.label.config(font=(font_name, font_size, font_weight))
```

```
# Create the main window
if __name__ == "__main__":
    root = tk.Tk()
    app = FontStyleChanger(root)
    root.mainloop()
```

```
import math

class Circle:
    def __init__(self, radius):

        self.radius = radius

    def __add__(self, other):

        if isinstance(other, Circle):
            return Circle(self.radius + other.radius)
        return NotImplemented

    def area(self):

        return math.pi * (self.radius ** 2)

    def __str__(self):
        return f"Circle with radius: {self.radius}"


circle1 = Circle(5)

circle2 = Circle(3)

print(circle1)

print(circle2)

circle3 = circle1 + circle2

print(circle3)

print(f"Area of the new circle: {circle3.area():.2f}")
```