TECNOLÓGICA DEL ORIENTE
INSTITUCIÓN DE EDUCACIÓN SUPERIOR

PROGRAMACIÓN ORIENTADA A OBJETOS

CÓDIGOS EDITABLES DE LOS EJEMPLOS DE POO

CÓDIGOS EDITABLES DE LOS EJEMPLOS DE POO

1. 1

```
// Excepción verificada: IOException
try {
    FileReader file = new FileReader("archivo.txt");
    // ...
} catch (IOException e) {
    System.out.println("Error al leer el archivo: " + e.getMessage());
}
// Excepción no verificada: NullPointerException
String str = null;
try {
    int length = str.length(); // Lanza NullPointerException
} catch (NullPointerException e) {
    System.out.println("Error: " + e.getMessage());
}
```

1. 2

```
Throwable
 |-- Exception
 |    |-- RuntimeException
 |    |    |-- NullPointerException
 |    |    |-- ArithmeticException
 |    |    |-- IndexOutOfBoundsException
 |    |         |-- ArrayIndexOutOfBoundsException
 |    |         |-- StringIndexOutOfBoundsException
 |    |-- IOException
 |    |    |-- FileNotFoundException
 |    |-- SQLException
 |    |-- ...
 |-- Error
      |-- OutOfMemoryError
      |-- StackOverflowError
      |-- VirtualMachineError
      |-- ...
```

1. 3

```
public void leerArchivo(String nombreArchivo) throws IOException {
    try {
        FileReader file = new FileReader(nombreArchivo);
        // ...
    } catch (FileNotFoundException e) {
        throw new IOException("Archivo no encontrado: " + nombreArchivo, e);
    } finally {
        // Código de limpieza o liberación de recursos
    }
}
```

1. 4

```
public class InsuficienteSaldoException extends Exception {
    private double saldoActual;
    private double saldoRequerido;
    public InsuficienteSaldoException(String message, double saldoActual, double
    saldoRequerido) {
        super(message);
        this.saldoActual = saldoActual;
        this.saldoRequerido = saldoRequerido;
    }
    public double getSaldoActual() {
        return saldoActual;
    }

    public double getSaldoRequerido() {
        return saldoRequerido;
    }
}
// Uso de la excepción personalizada
public void retirarDinero(double cantidad) throws InsuficienteSaldoException {
    if (saldoActual < cantidad) {
        throw new InsuficienteSaldoException("Saldo insuficiente", saldoActual, cantidad);
    }
    // ...
}
```

1.5

```
public void procesarArchivo(String nombreArchivo) {
    FileReader file = null;
    try {
        file = new FileReader(nombreArchivo);
        // Procesar el archivo
    } catch (FileNotFoundException e) {
        // Manejar el error de archivo no encontrado
        System.err.println("Error: Archivo no encontrado - " + e.getMessage());
    } catch (IOException e) {
        // Manejar otros errores de entrada/salida
        System.err.println("Error de entrada/salida - " + e.getMessage());
    } finally {
        if (file != null) {
            try {
                file.close();
            } catch (IOException e) {
                System.err.println("Error al cerrar el archivo - " + e.getMessage());
            }
        }
    }
}
```

2.1

java
```
JButton button = new JButton("Haz clic");
button.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        // Código para manejar el evento de clic del botón
        System.out.println("¡Se hizo clic en el botón!");
    }
});
```

2.2

java
```
JPanel panel = new JPanel();
panel.addMouseListener(new MouseAdapter() {
    public void mouseClicked(MouseEvent e) {
        // Código para manejar el evento de clic del mouse
        System.out.println("Clic del mouse en las coordenadas: " + e.getX() + ", " + e.getY());
    }
});
```

2.3

java
```
JButton button = new JButton("Haz clic");
button.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        // Código para manejar el evento de clic del botón
        System.out.println("¡Se hizo clic en el botón!");
    }
});
```
java
```
JTextField textField = new JTextField();
textField.addKeyListener(new KeyAdapter() {
    public void keyReleased(KeyEvent e) {
        // Código para manejar el evento de liberación de tecla en el campo de texto
        System.out.println("Tecla liberada: " + e.getKeyChar());
    }
});
JComboBox<String> comboBox = new JComboBox<>(new String[]{"Opción 1", "Opción 2",
"Opción 3"});
comboBox.addItemListener(new ItemListener() {
    public void itemStateChanged(ItemEvent e) {
        // Código para manejar el evento de cambio de estado del elemento seleccionado en
el combobox
        if (e.getStateChange() == ItemEvent.SELECTED) {
            System.out.println("Elemento seleccionado: " + e.getItem());
        }
    }
});
```

2.4

```
java
public class CalculatorApp extends JFrame implements ActionListener {
    private JTextField displayField;
    private JButton addButton, subtractButton, multiplyButton, divideButton, equalsButton;
    private double firstNumber, secondNumber, result;
    private String operator;

    public CalculatorApp() {
        // Inicialización de componentes de la GUI
        displayField = new JTextField();
        addButton = new JButton("+");
        subtractButton = new JButton("-");
        multiplyButton = new JButton("*");
        divideButton = new JButton("/");
        equalsButton = new JButton("=");

        // Asociar eventos a los botones
        addButton.addActionListener(this);
        subtractButton.addActionListener(this);
        multiplyButton.addActionListener(this);
        divideButton.addActionListener(this);
        equalsButton.addActionListener(this);

        // Configurar el diseño de la GUI
        // ...
    }
    public void actionPerformed(ActionEvent e) {
        if (e.getSource() == addButton) {
            operator = "+";
            firstNumber = Double.parseDouble(displayField.getText());
            displayField.setText("");
        } else if (e.getSource() == subtractButton) {
            // Lógica para el botón de restar
            // ...
        } else if (e.getSource() == multiplyButton) {
            // Lógica para el botón de multiplicar
            // ...
        } else if (e.getSource() == divideButton) {
            // Lógica para el botón de dividir
            // ...
        } else if (e.getSource() == equalsButton) {
            secondNumber = Double.parseDouble(displayField.getText());
            switch (operator) {
                case "+":
                    result = firstNumber + secondNumber;
                    break;
                case "-":
                    result = firstNumber - secondNumber;
```

```
                break;
            case "*":
                result = firstNumber * secondNumber;
                break;
            case "/":
                result = firstNumber / secondNumber;
                break;
        }
        displayField.setText(String.valueOf(result));
    }
}

public static void main(String[] args) {
    CalculatorApp calculator = new CalculatorApp();
    calculator.setVisible(true);
}
}
```

2.5

java

```
JButton button = new JButton("Haz clic");
button.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        // Código para manejar el evento de clic del botón
        System.out.println("¡Se hizo clic en el botón!");
    }
});
```

Ejemplo de implementación de la interfaz MouseListener:

java

```
JPanel panel = new JPanel();
panel.addMouseListener(new MouseAdapter() {
    public void mouseClicked(MouseEvent e) {
        // Código para manejar el evento de clic del mouse
        System.out.println("Clic del mouse en las coordenadas: " + e.getX() + ", " + e.getY());
    }
});
```

3.1

java

```
public void processData(String data) {
    try {
        System.out.println("Procesando datos: " + data); // Traza
        // Punto de ruptura establecido en la siguiente línea
        int result = Integer.parseInt(data);
        System.out.println("Resultado: " + result); // Traza
    } catch (NumberFormatException e) {
        System.out.println("Error de formato numérico: " + e.getMessage()); // Traza
    }
}
```

3.3

```
try {
    int result = divideNumbers(10, 0);
    System.out.println("Resultado: " + result);
} catch (Exception e) {
    System.out.println("Ocurrió un error.");
}

try {
    int result = divideNumbers(10, 0);
    System.out.println("Resultado: " + result);
} catch (ArithmeticException e) {
    System.out.println("Error: División por cero. " + e.getMessage());
}
```

3.4

```
import java.util.logging.*;

public class EjemploLogging {
    private static final Logger logger = Logger.getLogger(EjemploLogging.class.getName());

    public static void main(String[] args) {
        try {
            int result = divideNumbers(10, 0);
            logger.info("Resultado: " + result);
        } catch (ArithmeticException e) {
            logger.severe("Error de división por cero: " + e.getMessage());
        }
    }

    private static int divideNumbers(int a, int b) {
        return a / b;
    }
}
```

3.5

```
java
import org.junit.jupiter.api.Assertions;
import org.junit.jupiter.api.Test;

public class CalculadoraTest {

    @Test
    public void testDivisionPorCero() {
        Calculadora calculadora = new Calculadora();
        Assertions.assertThrows(ArithmeticException.class, () -> {
            calculadora.dividir(10, 0);
        });
```

```
    }

    @Test
    public void testDivisionValida() {
        Calculadora calculadora = new Calculadora();
        int resultado = calculadora.dividir(10, 2);
        Assertions.assertEquals(5, resultado);
    }
}
```