

Nombre:

75.07 / 95.02 - ALGORITMOS Y PROGRAMACIÓN III
SEGUNDO PARCIAL - Jueves 8/6/2023

.... / 3

1) Dado el siguiente programa:

```
1 package ejercicio1;
2 public class Main {
3     public static void main(String[] args) {
4         try {
5             aMethod();
6             System.out.println("D");
7         } catch (NumberFormatException ex) {
8             System.out.println("C");
9         } finally {
10            System.out.println("B");
11        }
12        System.out.println("A");
13    }
14
15    public static void aMethod() {
16        System.out.println("E" + 1 / 0);
17    }
}
```

- a) explique cuál sería la salida por pantalla al correr el programa original;
- b) explique cuál sería la salida por pantalla al correr el programa original, pero con la línea 15 borrada;
- c) explique cuál sería la salida por pantalla al correr el programa original, pero con la línea 7 cambiada por: } catch (Exception ex) {

2) Dado el siguiente programa:

.... / 2

```
1 package ejercicio2;
2 import java.io.*;
3
4 public class Main {
5
6     public static final java.util.Scanner teclado = new java.util.Scanner(System.in);
7     public static final java.io.PrintStream pantalla = new java.io.PrintStream(System.out);
8
9     public static void main(String[] args) throws FileNotFoundException, IOException {
10        pantalla.println("Ingrese el nombre de un archivo (si es necesario, con su ruta)");
11        pantalla.println("Por ejemplo: src/main/java/ejercicio2/Main.java");
12        pantalla.print("Archivo: ");
13        String nombreArchivo = teclado.nextLine();
14        FileReader fileReader = new FileReader(nombreArchivo);
15        LineNumberReader lineNumberReader = new LineNumberReader(fileReader);
16        String renglon;
17        while ((renglon = lineNumberReader.readLine()) != null) {
18            pantalla.println(lineNumberReader.getLineNumber() + "] " + renglon);
19        }
20    }
}
```

- a) indique por qué este programa, aunque es eficaz, no es eficiente. Explique cómo Ud. lo mejoraría;
- b) explique el funcionamiento del programa dando un breve ejemplo.

3) Dado el siguiente programa:

.... / 2

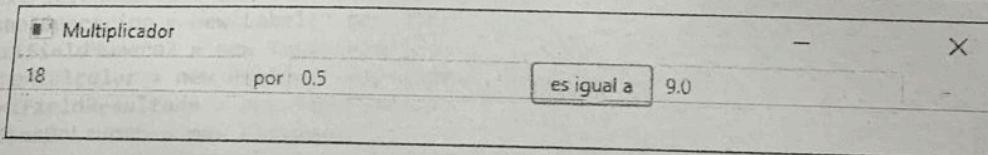
```
1 package ejercicio3;
2 import javafx.application.Application;
3 import javafx.stage.Stage;
4 public class App extends Application {
5     @Override
6     public void start(Stage stage) {
7         Vista vista = new Vista(stage);
8         Modelo modelo = new Modelo();
9         Controlador controlador = new Controlador(modelo, vista);
10        controlador.iniciar();
11    }
12    public static void main(String[] args) {
13        launch();
14    }
15 }
```

```
1 package ejercicio3;
2 import javafx.scene.Scene;
3 import javafx.scene.control.*;
4 import javafx.scene.layout.FlowPane;
5 import javafx.stage.Stage;
6 public class Vista {
7     private TextField textFieldNumero1;
8     private Label labelOperacion;
9     private TextField textFieldNumero2;
10    private Button botonCalcular;
11    private TextField textFieldResultado;
12    private FlowPane gestorDeLayout;
13    private Scene escenaPrincipal;
14    public Vista(Stage escenario) {
15        textFieldNumero1 = new TextField();
16        labelOperacion = new Label(" por ");
17        textFieldNumero2 = new TextField();
18        botonCalcular = new Button(" es igual a ");
19        textFieldResultado = new TextField();
20        gestorDeLayout = new FlowPane();
21        gestorDeLayout.getChildren().add(textFieldNumero1);
22        gestorDeLayout.getChildren().add(labelOperacion);
23        gestorDeLayout.getChildren().add(textFieldNumero2);
24        gestorDeLayout.getChildren().add(botonCalcular);
25        gestorDeLayout.getChildren().add(textFieldResultado);
26        escenaPrincipal = new Scene(gestorDeLayout, 600, 50);
27        escenario.setScene(escenaPrincipal);
28        escenario.setTitle("Multiplicador");
29        escenario.setResizable(false);
30        escenario.show();
31    }
32    public Button getBotonCalcular() {
33        return botonCalcular;
34    }
35    public TextField getTextFieldNumero1() {
36        return textFieldNumero1;
37    }
38    public TextField getTextFieldNumero2() {
39        return textFieldNumero2;
40    }
41    public TextField getTextFieldResultado() {
42        return textFieldResultado;
43    }
44 }
```

```

1 package ejercicio3;
2 public class Modelo {
3
4     public float operar(Vista vista) {
5         float n1 = Float.parseFloat(vista.getTextFieldNumero1().getText());
6         float n2 = Float.parseFloat(vista.getTextFieldNumero2().getText());
7         return n1 * n2;
8     }
9 }
10
11 package ejercicio3;
12 import javafx.event.ActionEvent;
13 import javafx.event.EventHandler;
14 public class Controlador {
15     private Modelo modelo;
16     private Vista vista;
17     public Controlador(Modelo modelo, Vista vista) {
18         this.modelo = modelo;
19         this.vista = vista;
20     }
21     public void iniciar() {
22         vista.getBotonCalcular().setOnAction(new EventHandler<ActionEvent>() {
23             @Override
24             public void handle(ActionEvent event) {
25                 float resu = modelo.operar(vista);
26                 vista.getTextFieldResultado().setText(Float.toString(resu));
27             }
28         });
29     }
30 }

```



- a) justifique por qué, aunque funciona, no se trata de una implementación estándar del patrón MVC;
 b) indique qué desventaja(s) ofrece esta versión frente a la implementación estándar del patrón MVC.

4) Dado el siguiente fragmento de código:

..... / 3

```

1 List<Integer> lista = Arrays.asList(-81, 64, 0, -4, 100, 49, -25, 9, 1);
2 System.out.println(lista.stream().filter(x -> x>=0).map(Math::sqrt).reduce(0.0, (x,y) -> x+y));

```

- a) utilizando términos y conceptos de la *Programación Funcional*, describa qué son y cómo funcionan `filter`, `map` y `reduce`; así como también `x -> x>=0`, `Math::sqrt` y `(x,y) -> x+y`;
 b) utilizando términos y conceptos de la *Programación Orientada a Objetos*, describa el funcionamiento del fragmento de código anterior e indique cuál sería su salida por pantalla;
 c) describa tres nuevos métodos `filtrar`, `mapear` y `reducir` (sus parámetros, tipos de retorno y lo que hacen, que NO necesariamente deben coincidir con los parámetros, tipos de retorno y lo que hacen `filter`, `map` y `reduce`) y sin utilizar la API Stream de Java >= 8, explique cómo usarlos juntos para obtener el mismo resultado que el producido por la línea 2 del fragmento de código anterior.