Nombre: [redacted] Padrón: [redacted] 10/10

75.07 / 95.02 - ALGORITMOS Y PROGRAMACIÓN III - SEGUNDO PARCIAL - Jueves 17/11/2022

2/2

1) Dado el siguiente programa:

```
1  package ejercicio1;
2  public class Main {
3    public static final java.util.Scanner teclado = new java.util.Scanner(System.in);
4    public static final java.io.PrintStream pantalla = new java.io.PrintStream(System.out);
5    public static void main(String[] args) {
6      try {
7        pantalla.print("Ingrese el dividendo (un entero): ");
8        int dividendo = Integer.parseInt(teclado.nextLine());
9        pantalla.print("Ingrese el divisor (un entero != 0): ");
10       int divisor = Integer.parseInt(teclado.nextLine());
11       pantalla.printf("Cociente: %d\nResto: %d\n", dividendo/divisor, dividendo%divisor);
12     } catch (Exception e) {
13       pantalla.printf("%s\n%s\n", e, "No se puede hacer la division entera si divisor == 0.");
14     } catch (NumberFormatException e) {
15       pantalla.printf("%s\n%s\n", e, "El dato ingresado no es un numero entero.");
16     }
17   }
18 }
```

B a) indique qué se le debería corregir para que compile y corra, justificando su respuesta;

B b) explique su funcionamiento dando suficientes ejemplos.

2/2

2) Dado el siguiente programa:

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

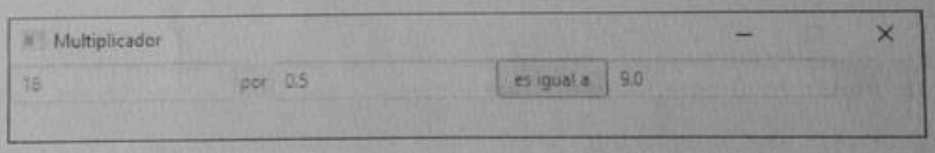
B a) identifique un patrón de diseño e indique dónde y por qué está siendo aplicado el mismo;

B b) explique el funcionamiento del programa dando un breve ejemplo.

3) Dado el siguiente programa para refactorizar de acuerdo al patrón MVC: 3../3

```
1  package algo3.parcial2_gui;
2  import javafx.application.Application;
3  import javafx.scene.Scene;
4  import javafx.scene.control.*;
5  import javafx.scene.layout.FlowPane;
6  import javafx.stage.Stage;
7  public class App extends Application {
8    @Override
9    public void start(Stage escenario) {
10     TextField numero1 = new TextField();
11     Label labelOperacion = new Label(" por ");
12     TextField numero2 = new TextField();
13     Button botonCalcular = new Button(" es igual a ");
14     TextField resultado = new TextField();
15     FlowPane gestorDeLayout = new FlowPane();
16     gestorDeLayout.getChildren().add(numero1);
17     gestorDeLayout.getChildren().add(labelOperacion);
18     gestorDeLayout.getChildren().add(numero2);
19     gestorDeLayout.getChildren().add(botonCalcular);
20     gestorDeLayout.getChildren().add(resultado);
21     Scene escenaPrincipal = new Scene(gestorDeLayout, 600, 50);
22     escenario.setScene(escenaPrincipal);
23     escenario.setTitle("Multiplicador");
24     escenario.setResizable(false);
25     escenario.show();
26     botonCalcular.setOnAction(event -> {
27       float n1 = Float.parseFloat(numero1.getText());
28       float n2 = Float.parseFloat(numero2.getText());
29       float resu = n1 * n2;
30       resultado.setText(Float.toString(resu));
31     });
32   }
33   public static void main(String[] args) {
34     launch();
35   }
36 }
```

VISTA



B a) indique qué parte del código* debería implementarse en el modelo;
B b) indique qué parte del código* debería implementarse en la vista;
B c) indique qué parte del código* debería implementarse en el controlador. (*) eventualmente adaptado

4) Dado el siguiente fragmento de código: 3.../3

```
1  List<Integer> lista = Arrays.asList(-81, 64, 0, -4, 100, 49, -25, 9, 1);
2  mostrar(calcularRaicesCuadradas(filtrar(ordenar(lista))));
```

B- a) describa qué son, cómo funcionan y cómo, desde el punto de vista de la programación funcional, están siendo utilizados en el código *mostrar*, *calcularRaicesCuadradas*, *filtrar* y *ordenar*;

B b) indique por qué la implementación dada, aunque es eficaz, no es eficiente; explique cómo la mejoraría (sin usar la API *Stream* de Java >= 8);

B- c) explique cómo expresar el algoritmo corregido, ahora con la API *Stream* de Java >= 8.

Salida:

```
0.0
1.0
3.0
7.0
8.0
10.0
```