**Estándar de codificación para**

** Java*FX***

|  |  |
| --- | --- |
| **Propósito** | Guiar la implementación de programas en Java*FX* para estandarizar la manera de escribir el código para todo el equipo de desarrolladores. |
| **Comentarios** | * Deben contener solo información relevante para la lectura y entendimiento del programa * Mantener los comentarios actualizados conforme el programa va siendo modificado * No incluir caracteres especiales * Los comentarios siempre deben estar indentados al mismo nivel que el código que describen * Los comentarios multilínea deben escribirse como comentarios de bloque (/\* … \*/) * Los comentarios de una línea deben comentarse con // |
| **Ejemplo** | /\*  \* Nombre del archivo: Ejemplo.java  \* Autor: Raul Hernandez  \* Paquete: sgbp.modelo.pojo  \* Fecha de creación: 01/10/2023  \* Fecha de modificación: 08/10/2023  \* Descripción: clase para imprimir un texto  \*/  public class Ejemplo {  int numero1 = 5; // Número entero 1  int numero2 = 3; // Número entero 2  if (numero1 > numero2) {  // Comentario: Si numero1 es mayor que numero2  System.out.println("Número 1 es mayor.");  }  } |
| **Declaraciones** | En los archivos .java se deben seguir los siguientes criterios:   * Realizar solo una declaración por línea * Escribir las declaraciones al principio del bloque de código donde serán utilizadas * Inicializar las variables locales donde son declaradas, a menos que la solución del código lo requiera distinto |
| **Ejemplo de declaraciones** | public class Ejemplo {  int numero1 = 1;  int numero2 = 2;  System.out.println(numero1 + numero2);  if(numero1 = numero2) {  int num3 = 3;  System.out.println(num3);  }  } |
| **Líneas en blanco** | Se deben dejar dos saltos de línea en las siguientes circunstancias:   * Entre métodos * Entre las variables locales de un método y su primera sentencia * Antes de un comentario (excepto el encabezado) * Antes de estructura de control |
| **Ejemplo** | public class Ejemplo {  public void metodo1() {  int numero1 = 10;  System.out.println("Numero: " + numero);  }  public void metodo2() {  int numero2 = 10;  while (numero2 > 0) {  numero2 = numero2 - 1;  }  }  } |
| **Espacios en blanco** | Se debe dejar un espacio en blanco en las siguientes circunstancias:   * Una palabra reservada del lenguaje seguida por un paréntesis * Después de cada coma en las listas de argumentos * Antes y después de operadores binarios * Las expresiones dentro de la cláusula de control de una sentencia for * Antes de una llave de apertura |
| **Ejemplo** | public class Ejemplo {  int num1 = 5;  int num2 = 3;  if (num1 > num2) {  System.out.println("Número 1 es mayor.");  } else {  System.out.println("Número 2 es mayor.");  }  for (int i = 0; i < 5; i++) {  System.out.println("Iteración " + i);  }  } |
| **Indentación** | * Se deben usar 4 espacios como unidad de indentación * Se debe indentar el contenido dentro de cada nuevo bloque de código * Cuando existan sentencias anidadas se debe aumentar el nivel de indentación * Las expresiones separadas por su longitud en varías líneas, la nueva línea quedará al nivel de dos tabulaciones a la derecha respecto a donde comienza |
| **Ejemplo** | if (!nuevaCarrera.isEmpty()) {  if (!verificarExistenciaCarrera(nuevaCarrera) {  agregarCarrera(nuevaCarrera);  } else {  lblValidacion.setText("La carrera ya existe");  }  } else {  lblValidacion.setText("Debes agregar un valor…");  } |
| **Longitud de la línea** | Evitar la línea de más de 120 caracteres.  Cuando una expresión no entre en una sola línea, separarla de acuerdo con los siguientes criterios:   * Después de una coma * Antes de un operador * Después de un signo +.   Una expresión separada en varias líneas, será considerada como solo una línea de código para la aplicación de las reglas de líneas en blanco |
| **Ejemplo** | public void mostrarProductos() {  for (int i = 0; i < cont; i++) {  System.out.println("Descripcion de otras cosas: "  + productos[i].getDescripcion());  }  } |
| **Jerarquía** | Los modificadores tienen el siguiente orden de declaración para variables, constantes y métodos:   1. Static 2. Public 3. Private 4. Protected 5. Sin modificador (default)   **Variables y constantes**  Se declaran en el siguiente orden   1. Constantes 2. Variables @FXML 3. Resto de variables   **Métodos**  Los métodos se declaran después de constantes y variables en el siguiente orden   1. Método main (en las clases que lo incluyan) 2. Métodos @Override 3. Métodos @FXML 4. Resto de métodos |
| **Ejemplo** | public class MiClase extends Application {  private static final int ANCHO\_VENTANA = 800;  public static int variablePublica;  private int variablePrivada;  protected String variableProtegida;  int variableDefault;  @FXML  private Lavel lblNombre;    public static void main (String[] args){  launch(args);  }  @Override  public void init() throws Exception {  super.init();  }  @FXML  private void realizarAccion(){  //TODO  }    private void metodoPrivado() {  //TODO  }  } |
| **Sentencias** | * Siempre deben usar llaves de apertura y cierre * La llave de apertura se coloca en la misma línea que la estructura de control y se deja un espacio en blanco antes de ella * La llave de cierre se coloca en una línea separada y al mismo nivel de indentación que donde inicia la estructura de control * Cada case de la estructura switch debe llevar llaves de apertura y cierre * Cada vez que un case no incluya la sentencia break, incluir un comentario donde se encontraría normalmente |
| **Ejemplo** | **Sentencias if, if-else, if else-if else**  if (condicion) {  sentencias;  } else if (condicion) {  sentencias;  } else {  sentencias;  }  **Sentencias for**  for (inicializacion; condicion; actualizacion) {  sentencias;  }  **Sentencias while**  while (condicion) {  sentencias;  }  **Sentencias do-while**  do {  sentencias;  } while (condicion);  **Sentencias switch**  switch (condicion) {  case 1: {  sentencias;  //Este caso se propaga  }  case 2: {  sentencias;  break;  }  } |
| **Nombrado** | * Evitar acrónimos y abreviaturas (a menos que sean reconocidas) * Usar nombres descriptivos y con la menor ambigüedad posible   **Clases**   * Deben ser sustantivos en singular (salvo que se requiera algo diferente) * Evitar el uso de verbos * Seguir el estilo PascalCase   **Métodos**   * Deben ser verbos * Seguir el estilo camelCase * Los métodos de acceso llevan el prefijo get * Los métodos mutators llevan el prefijo set   **Variables**   * El nombre debe indicar la función que cumple la variable * Seguir el estilo camelCase * Evitar el nombrado de un solo carácter, a menos que sean variables para índices temporales   **Constantes**   * Seguir el estilo SCREAMING\_SNAKE \_CASE |
| **Ejemplo** | **Clases**  public class PastorAleman {  codigo;  }  **Métodos**  private void agregarCarrera (String carrera) {  codigo;  }  **Variables**  String carreraNueva = “Estadistica”;  **Constantes**  static final int ANCHURA\_MINIMA = 4; |
| **Nombrado de recursos JavaFX** | Cualquier componente no mencionado aquí, será agregado posteriormente   |  |  |  | | --- | --- | --- | | **Componente** | **Prefijo** | **Ejemplo** | | TextField | tf | tfCantidad | | Label | lbl | lblResultado | | Button | btn | btnCalcularIva | | ComboBox | cb | cbCarreras | | RadioButton | rbtn | rBtnAzul | | CheckBox | chb | chbJamon | | Pane | pane | paneFondo | | AnchorPane | apane | apaneVentana | | PasswordField | pf | pfLogin | | TextArea | ta | taDescripcion | | ImageView | iv | ivFoto | | DatePicker | dp | dpFechaNacimiento | | TableView | tv | tvPacientes | | TableColumn | col | colNombre | | VBox | vbox | vboxBitacorasEstudiante | |
| **Nombrado de archivos** | **FXML**  Deben llevar el prefijo FXML seguido por el nombre de la vista   * Usar el estilo PascalCase   **Controller**  Deben nombrarse igual que la vista.  **CSS**  Debe llevar el prefijo “escena”   * Usar el estilo camelCase   **Paquetes**   * Deben nombrarse en lowercase |
| **Ejemplo** | **FXML**  FXMLVistaPrincipal.fxml  **CSS**  escenaVistaPrincipal.css  **Controller**  FXMLVistaPrincipal.java  **Paquete**  sgbp.vista |
| **Lenguaje** | Todo nombrado será en español, a menos que el propio estándar indique otra cosa  Excepto:   * Si la palabra en español incluye ñ * Es un nombre propio * Si es parte de un patrón de diseño o arquitectura de software |
| **Ejemplo** | private void calcularRaiz();  private passwordField pfPassword();  FXMLRegistrarDefecto.fxml |