MANUAL TÉCNICO

Botón Generar: Al hacer clic en el botón “Generar” se limpiará el ArrayList y la Tabla que pudieran tener información antes gracias al siguiente código:

```
private void generarMouseClicked(java.awt.event.MouseEvent evt) {
    // TODO add your handling code here:
    arregloNumeros.clear();
    table.setModel(new DefaultTableModel(null, new String[]{"Posición", "Elemento"}));
```

Si el usuario no ingresa un valor para el tamaño del vector se activará un JOptionPane que le avisará de ello:

```
if (tamañoVector.getText().equals("")) {
    JOptionPane.showMessageDialog(null, "Señor usuario por favor ingrese un número antes de iniciar.");
} else {
```

Sino, la entrada se guarda en una variable tipo entero. Si en el ComboBox de preferencias el usuario escoje la opción “Aleatorio”, se creará un objeto de la clase Random para generar los elementos del vector. Éste código genera esos números aleatorios, los guarda dentro del ArrayList y muestra cada posición en una Tabla:

```
else {
  int numeroUsuario = Integer.parseInt(tamañoVector.getText());
  if (preferenciaCB.getSelectedItem().equals("Aleatorio")) {
      Random aleatorio = new Random();
      for (int i = 0; i < numeroUsuario; i++) {
          int numeroAleatorio = aleatorio.nextInt(101);
          int posicion = i + 1;
          DefaultTableModel model = (DefaultTableModel) table.getModel();
          model.addRow(new Object[]{posicion, numeroAleatorio});
          arregloNumeros.add(numeroAleatorio);
      }
  } else {
```

Este Else significa que el usuario tuvo que elegir la opción “Manual” en preferencias. De esta manera se muestran varios JOptionPane para pedir al usuario posición por posición del vector. Si el usuario ingresa algo diferente a un número entero se mostrará otro JOptionPane que le avise de esto para que el usuario pueda corregirlo.

```
else {
  String cadena = "";
  for (int i = 0; i < numeroUsuario; i++) {
      int numero = 0;
      while (true) {
          cadena = JOptionPane.showInputDialog("Digite el elemento en la posición " + (i + 1) + ":");
          try {
              numero = Integer.parseInt(cadena);
              break;
          } catch (NumberFormatException e) {
              JOptionPane.showMessageDialog(null, "Ingrese un número entero.");
          }
      }
```

La siguiente parte añade cada número ingresado por el usuario al ArrayList. Por cada número se pregunta si desea continuar llenando mediante un JOptionPane. Si la respuesta es “Sí” se vuelve a mostrar el JOptionPane para ingresar el siguiente número. Si la respuesta es “No” o se cierra directamente el JOptionPane, se llena el resto de posiciones del ArrayList con ceros (0) y se muestran todos los elementos en la Tabla.

```
int posicion = i + 1;
arregloNumeros.add(numero);
DefaultTableModel model = (DefaultTableModel) table.getModel();
model.addRow(new Object[]{posicion, numero});
int num = JOptionPane.showConfirmDialog(null, "¿Desea ingresar más elementos? Si su respuesta es NO"
        + "se llenarán los elementos restantes con 0.", "Ingresar", JOptionPane.YES_NO_OPTION,
        JOptionPane.QUESTION_MESSAGE);
if (num == JOptionPane.CLOSED_OPTION || num == JOptionPane.NO_OPTION) {
    for (int j = i + 1; j < numeroUsuario; j++) {
        int pos = j + 1;
        arregloNumeros.add(0);
        model.addRow(new Object[]{pos, 0});
    }
    break;
}
```

Botón Borrar: Al hacer clic en el botón borrar se limpiará el ArrayList, la Tabla y todos los TextFields que puedan contener algún valor en ellos:

```
private void borrarMouseClicked(java.awt.event.MouseEvent evt) {
    table.setModel(new DefaultTableModel(null, new String[]{"Posición", "Elemento"}));
    arregloNumeros.clear();
    tamañoVector.setText("");
    posicionTF1.setText("");
    posIntercambio1.setText("");
    posIntercambio2.setText("");
}
```

Botón Cambiar: Con esta parte del código se controla que el usuario no haya dejado en blanco el TextField para introducir el índice del vector, ni haya introducido un índice que no esté dentro del vector.

```
private void cambiarMouseClicked(java.awt.event.MouseEvent evt) {
    // TODO add your handling code here:
    if (posicionTF1.getText().equals("")) {
        JOptionPane.showMessageDialog(null, "Señor usuario por favor ingrese la posición del vector que desea cambiar.");
    } else {
        int posicion = Integer.parseInt(posicionTF1.getText());
        if (posicion > arregloNumeros.size() || posicion < 1) {
            JOptionPane.showMessageDialog(null, "Señor usuario por favor ingrese una posición que esté dentro del vector.");
        } else {
```

En caso de que el usuario haya elegido cambiar el elemento del vector de forma aleatoria, se implementa este código. Éste genera un número aleatorio y lo guarda en el ArrayList en el índice introducido por el usuario, además actualiza la Tabla.

```
if (preferenciaCambiarCB.getSelectedItem().equals("Aleatorio")) {
    Random aleatorio = new Random();
    int numeroAleatorio = aleatorio.nextInt(101);
    arregloNumeros.set((posicion - 1), numeroAleatorio);
    DefaultTableModel model = (DefaultTableModel) table.getModel();
    model.setValueAt(arregloNumeros.get(posicion - 1), (posicion - 1), 1);
} else {
```

Si el usuario no elige “Aleatorio” habrá elegido “Manual”. Se le pide al usuario el elemento que va a cambiar en dicha posición por medio de un JOptionPane y se controla que introduzca un número entero.

```
else {
  String cadena;
  int numero;
  while (true) {
      cadena = JOptionPane.showInputDialog("Digite el elemento en la posición " + posicion + ":");
      try {
          numero = Integer.parseInt(cadena);
          break;
      } catch (NumberFormatException e) {
          JOptionPane.showMessageDialog(null, "Ingrese un número entero.");
      }
  }
```

Luego se introduce el número del usuario en la posición que haya ingresado y se actualiza en la Tabla con el siguiente código:

```
arregloNumeros.set((posicion - 1), numero);
DefaultTableModel model = (DefaultTableModel) table.getModel();
model.setValueAt(arregloNumeros.get(posicion - 1), (posicion - 1), 1);
```

TextFields: El siguiente código se utiliza en los TextFields para hacer que el usuario sólo pueda ingresar números en los mismos:

```
private void tamañoVectorKeyTyped(java.awt.event.KeyEvent evt) {
    // TODO add your handling code here:
    char c = evt.getKeyChar();
    if (!(Character.isDigit(c) || c == KeyEvent.VK_BACK_SPACE || c == KeyEvent.VK_DELETE)) {
        evt.consume();
    }
}
```

Botón Intercambiar: Al clicar, se controla que el usuario sí haya introducido algún valor:

```
private void intercambiarMouseClicked(java.awt.event.MouseEvent evt) {
    // TODO add your handling code here:
    if (posIntercambio1.getText().equals("") || posIntercambio2.getText().equals("")){
        JOptionPane.showMessageDialog(null, "Señor usuario por favor ingrese las posiciones a intercambiar.");
    } else {
```

Si se ha introducido, se guardan las posiciones a intercambiar en dos variables enteras. Se controla si el usuario introduce posiciones que no estén dentro del vector con un JOptionPane que le avise de esto.

```
else {
  int pos1 = Integer.parseInt(posIntercambio1.getText()), pos2 = Integer.parseInt(posIntercambio2.getText());
  if (pos1 > arregloNumeros.size() || pos2 > arregloNumeros.size() || pos1 < 1 || pos2 < 1) {
      JOptionPane.showMessageDialog(null, "Señor usuario por favor ingrese posiciones que estén dentro del vector.");
  } else {
```

Si las posiciones introducidas están dentro del vector, se invoca el método **intercambioPosicion**, mandándole el ArrayList y las dos posiciones como parámetros. Por último, se muestran los valores intercambiados en la Tabla.

```
else {
  intercambioPosicion(arregloNumeros, (pos1 - 1), (pos2 - 1));
  DefaultTableModel model = (DefaultTableModel) table.getModel();
  model.setValueAt(arregloNumeros.get((pos1 - 1)), (pos1 - 1), 1);
  model.setValueAt(arregloNumeros.get((pos2 - 1)), (pos2 - 1), 1);
```

<u>Método intercambioPosicion:</u> Recibe un ArrayList y dos posiciones de este como parámetros. Se crea una variable entera para almacenar el dato en la primera posición introducida, se almacena el elemento de la segunda posición dada en la primera posición y por último se utiliza la variable para guardar el dato que había en la primera posición en la segunda posición introducida por el usuario.

```
private void intercambioPosicion(ArrayList<Integer> x, int pos1, int pos2) {
    int y;
    y = x.get(pos1);
    x.set(pos1, x.get(pos2));
    x.set(pos2, y);
}
```

<u>Botón Invertir Orden:</u> invoca el método **invertirOrden**.

```
private void invertirMouseClicked(java.awt.event.MouseEvent evt) {
    // TODO add your handling code here:
    invertirOrden(arregloNumeros);
}
```

Método invertirOrden: Recibe como único parámetro un ArrayList. Invierte el orden de los elementos del vector, cambiando el primero elemento con el último, el segundo con el penúltimo, y así sucesivamente. También actualiza la Tabla que se muestra en la Interfaz.

```
private void invertirOrden(ArrayList<Integer> x) {
    int j = (x.size() - 1);
    for (int i = 0; i < (x.size() / 2); i++) {
        intercambioPosicion(x, i, j);
        DefaultTableModel model = (DefaultTableModel) table.getModel();
        model.setValueAt(arregloNumeros.get(i), i, 1);
        model.setValueAt(arregloNumeros.get(j), j, 1);
        if (i == j || i + 1 == j) {
            break;
        }
        j--;
    }
}
```

Botón Ordenar: Invoca el método **ordenar**:

```
private void ordenarMouseClicked(java.awt.event.MouseEvent evt) {
    // TODO add your handling code here:
    ordenar(arregloNumeros);
}
```

Método ordenar: Recibe como parámetro un ArrayList. Ordena el ArrayList de forma ascendente, de menor a mayor valor, al mismo tiempo que va modificando la Tabla. Este método utiliza el método **buscarMenor**.

```
private void ordenar(ArrayList<Integer> x) {
    int m = x.size();
    for (int i = 0; i < (m - 1); i++) {
        int posMenor;
        posMenor = buscarMenor(x, i);
        intercambioPosicion(x, i, posMenor);
        DefaultTableModel model = (DefaultTableModel) table.getModel();
        model.setValueAt(arregloNumeros.get(i), i, 1);
        model.setValueAt(arregloNumeros.get(posMenor), posMenor, 1);
    }
}
```

Método buscarMenor: Recibe como parámetro un ArrayList y un entero como posición inicial de búsqueda. El método busca el elemento menor del vector y retorna su posición como un entero.

```
private int buscarMenor(ArrayList<Integer> x, int posInicial) {
    int m = x.size(), menor = x.get(posInicial), posMenor = posInicial;
    for (int i = (posInicial + 1); i < m; i++) {
        if (menor > x.get(i)) {
            menor = x.get(i);
            posMenor = i;
        }
    }
    return posMenor;
}
```