Universidad San Carlos de Guatemala
Estructura de datos
Ing. Oliver Sirra
David Enrique Lux Barrera 201931344

Manual de Tecnico:

Introducción:

El siguiente programa demuestra la funcionalidad de los conocimientos aplicados acerca de los temas vistos en clase, sobre un programa de carrera de caballos en donde el usuario puede ingresar las apuestas de las personas y muestra como estas son procesadas por varios metodos y generar los resultados por medio de las tablas correspondientes donde indica quienes son los ganadores o quienes estuvieron mas cerca de acertar su predicción.

Estructura:

Se puede ver que la complegidad del metodo creado para **agregar las apuestas** esta determinada por un O(1), ya que esta para cumpliir con una de las apuestas que se desarrolla devera de pasar por el arreglo creado que almacena las 10 posiciones de los caballos, ademas de los datos de los apostantes.

```
Apuesta apuesta = new Apuesta();
 int[] posicion = new int[10];
 double pasos = 0;
 try {
    long startTime = System.currentTimeMillis();
    int comas[] = new int[11];
    int x = 0;
    for (int i = 0; i < linea.length() - 1; i++) {
       pasos++;
       int j = i + 1;
       if (",".equals(linea.substring(i, j))) {
          pasos++;
          comas[x] = i;
          x++;
       }
    }
String nombre = linea.substring(1, comas[0] - 1);
    double monto = Double.parseDouble(linea.substring(comas[0] + 2, comas[1] - 1));
    posicion[0] = Integer.parseInt(linea.substring(comas[1] + 2, comas[2] - 1));
    posicion[1] = Integer.parseInt(linea.substring(comas[2] + 2, comas[3] - 1));
    posicion[2] = Integer.parseInt(linea.substring(comas[3] + 2, comas[4] - 1));
    posicion[3] = Integer.parseInt(linea.substring(comas[4] + 2, comas[5] - 1));
    posicion[4] = Integer.parseInt(linea.substring(comas[5] + 2, comas[6] - 1));
    posicion[5] = Integer.parseInt(linea.substring(comas[6] + 2, comas[7] - 1));
    posicion[6] = Integer.parseInt(linea.substring(comas[7] + 2, comas[8] - 1));
    posicion[7] = Integer.parseInt(linea.substring(comas[8] + 2, comas[9] - 1));
```

```
    posicion[8] = Integer.parseInt(linea.substring(comas[9] + 2, comas[10] - 1));
    posicion[9] = Integer.parseInt(linea.substring(comas[10] + 2, linea.length() - 1));

    apuesta.setNombre(nombre);
    apuesta.setMonto(monto);
    apuesta.setOrden(lugares);
    apuesta.setValidacion(true);
    pasos += 10;
    long endTime = System.currentTimeMillis();
    reporte.setTiempoIngreso(((endTime - startTime)/1000));
    reporte.setPasosIngreso(pasos);
} catch (Exception e) {
    nuevaApuesta.setError("Error");
    nuevaApuesta.setValidacion(false);
}
apuestas[numero] = apuesta;
```

P**ara la creacion de la verificación de las apuestas** se implemento un O(n) en la estructura utilizando de forma continua dos ciclos de repeticion que en este caso es el for donde n es el numero de apuestas que se verifican dentro del cuadro de texto.

```
 areaResultadosApuestas.setText(null);
double pasos = 0;
long startTime = System.currentTimeMillis();
for (int i = 0; i < 10; i++) {
    pasos++;
    for (int j = 0; j < apuestas.length; j++) {
        pasos++;
        if (apuestas[j].isValidacion() == true) {
            pasos++;
            int[] orden = apuestas[j].getOrden();
            if (resultados[i] == orden[i]) {
                pasos++;
                if (i == 0) {
                    apuestas[j].aumentarPunteo(10);
                } else if (i == 1) {
                    apuestas[j].aumentarPunteo(9);
                } else if (i == 2) {
                    apuestas[j].aumentarPunteo(8);
                } else if (i == 3) {
                    apuestas[j].aumentarPunteo(7);
                } else if (i == 4) {
                    apuestas[j].aumentarPunteo(6);
                } else if (i == 5) {
                    apuestas[j].aumentarPunteo(5);
                } else if (i == 6) {
                    apuestas[j].aumentarPunteo(4);
                } else if (i == 7) {
```

```
                apuestas[j].aumentarPunteo(3);
            } else if (i == 8) {
                apuestas[j].aumentarPunteo(2);
            } else if (i == 9) {
                apuestas[j].aumentarPunteo(1);
            }
        }
    }
}
```

Calculo de resultados al finalizar la carrera de igual manera se utilizo un O(n)

```
long startTime = System.currentTimeMillis();
double pasos = 0;
areaResultadosApuestas.setText(null);
int posicion;
Apuesta menor = new Apuesta();
Apuesta auxiliar = new Apuesta();

for (int i = 0; i < apuestas.length; i++) {
    pasos++;
    menor = apuestas[i];
    posicion = i;
    for (int j = i + 1; j < apuestas.length; j++) {
        pasos++;
        if (apuestas[j].getPunteo() < menor.getPunteo()) {
            menor = apuestas[j];
            posicion = j;
            pasos++;
        }
    }
    if (posicion != i) {
        pasos++;
        auxiliar = apuestas[i];
        apuestas[i] = apuestas[posicion];
        apuestas[posicion] = auxiliar;
    }
}
long endTime = System.currentTimeMillis();
reporte.setTiempoOrdenamientoP(((endTime - startTime) / 1000));
reporte.setPasosOrdenamientoP(pasos);
```

Ordenamiento de los resultados se untilizo un O(n2) el cual esta usa un ordenamiento tipo burbuja, ademas de que maneja dos ciclos for y es un ordenamiento por selección y es mas efectivo al ordenar cantidades.

Capturador de espacios utiliza una ocurrencia de tipo O(n), ya que como se puede apreciar es una secuencia que necesita capturar n linas para poder determinar la posicion de los caballos en las apuestas, al igual que el obtener las lineas de las posiciones.

```
public int numLineas() {
        int linea = 0;
        String texto = jTextArea1.getText();
        for (int i = 0; i < texto.length() - 1; i++) {
            if (texto.substring(i, i + 1).equals("\n")) {
                linea++;
            }
        }
        linea++;
        return linea;
    }

        public String[] ObtLineas(int numeroLineas, String texto) {
        String[] lineas = new String[numeroLineas];
        int lineasContadas = 0;
        int ultima = 0;
        for (int i = 0; i < texto.length() - 1; i++) {
            if (texto.substring(i, i + 1).equals("\n")) {
                lineas[lineasContadas] = texto.substring(ultima, i);
                lineasContadas++;
                ultima = i + 1;

            }
```