PD1

Ejercicio 1

1) ¿cómo se genera un vector monótonamente ascendente

Un vector monótonamente ascendente se genera llenando un arreglo donde cada elemento está ordenado de mayor a menor.

```
public int[] generarDatosAscendentes(int tamanio) {
    int[] vector = new int[tamanio];
    for (int i = 0; i < tamanio; i++) {
        vector[i] = i; // o cualquier función creciente como vector[i] = i * incremento
    }
    return vector;
}
```

2) ¿cómo se genera un vector monótonamente descendente?

Un vector monótonamente descendente se genera rellenando un arreglo donde cada elemento es menor o igual al anterior. Podría implementarse así:

```
public int[] generarDatosDescendentes(int tamanio) {
    int[] vector = new int[tamanio];
    for (int i = 0; i < tamanio; i++) {
        vector[i] = tamanio - i - 1; // decreciendo desde tamanio - 1
    }
    return vector;
}
```

3) ¿cómo se genera un vector con valores aleatorios? ¿pueden existir claves repetidas? ¿cuál es el orden del tiempo de ejecución de este método?

Se podría generar valores aleatorios utilizando la clase Random.
Pueden existir claves repetidas porque los números aleatorios no son únicos, puede salir otro igual que el anterior.
El tiempo de ejecución de este método es O(n), donde n es el tamaño del vector. Esto se debe a que se genera un número aleatorio para cada posición del vector.

```
public int[] generarDatosAleatorios(int tamanio) {
    Random random = new Random();
```

```
    int[] vector = new int[tamanio];
    for (int i = 0; i < tamanio; i++) {
        vector[i] = random.nextInt(); // Rango ajustable con nextInt(max)
    }
    return vector;
}
```

4) ¿cuántos elementos contiene el vector de datos generado?¿cómo se puede modificar esta clase para que la cantidad de elementos del vector sea parametrizable?

En el diseño actual, la cantidad de elementos depende de tamanioMax que es 32000

```
public int[] generarDatosAscendentes(int tamanio) { ... }
public int[] generarDatosDescendentes(int tamanio) { ... }
public int[] generarDatosAleatorios(int tamanio) { ... }
```

ejemplo

```
        public int[] generarDatosDescendentes() {
                return generarDatosDescendentes(TAMANIO_MAX);
        }

        public int[] generarDatosDescendentes(int tamaño) {
                int[] copiaDescendente = new int[tamaño];
                for (int i = 0; i < tamaño; i++) {
                        copiaDescendente[i] = tamaño - i;
                }
                return copiaDescendente;
        }
```

5) ¿cómo podemos verificar que un conjunto está ordenado? ¿cuál sería el orden del tiempo de ejecución de un algoritmo que lo haga?

```
public boolean estaOrdenado(int[] vector) {
    for (int i = 0; i < vector.length - 1; i++) {
        if (vector[i] > vector[i + 1]) {
            return false; // No está ordenado
        }
    }
    return true; // Está ordenado
}
```

El tiempo de ejecución es O(n) en el mejor y el peor de los casos, donde n es el tamaño del vector.