Ejercicio 1:

¿calcular cuántos años más podremos seguir utilizando esta forma de contar?

Tipo long: 64 bits (1 bit de signo y 63 bits de representación del número)

$$2^{63} milisegundos \times \frac{1\ año}{3{,}1536 \times 10^{10}\ milisegundos} = 292{,}471 \times 10^{6}\ años$$

Ejercicio 2:

¿Qué significa que el tiempo medido sea 0?

Que el tamaño del problema sea lo suficientemente pequeño y que el ordenador sea lo suficientemente rápido como para que al computar la operación a medir nos dé un tiempo inferior a 1 milisegundo.

¿A partir de qué tamaño de problema (n) se obtienen tiempos válidos >=50 msg?

Ejecutamos el programa Vector2.java con un tamaño de problema tal que cumpla lo preguntado.

Ejecutamos fuera del paquete p11 la orden: java -Djava.compiler=NONE p11.Vector2 n

Obtenemos los tiempos válidos a partir de un tamaño n = 5000000, obteniendo un tiempo de ejecución de 62 msg.

Ejercicio 3:

¿Qué pasa con el tiempo si el tamaño del problema se multiplica por 2?

Que también se multiplica por 2 al tratarse de un algoritmo con complejidad lineal (recordemos que el algoritmo de complejidad lineal del que se habla es el del método suma() de Vector1)

¿Qué pasa con el tiempo si el tamaño del problema se multiplica por otro k que no sea 2? (Pruebe, por ejemplo, para k=3 y k=4 y compruebe los tiempos obtenidos.)

Al ser un algoritmo de complejidad lineal, si se multiplica el tamaño del problema por una constante “k” el tiempo de ejecución se multiplicará por dicha constante “k”.

Modificamos el código fuente de Vector4 para que en cada iteración del bucle principal el tamaño del problema se multiplique por la constante k=3 y luego k=4.

Nos metemos dentro del paquete p11 y ejecutamos la orden: javac Vector4.java
Ahora tenemos el proyecto Vector4.java compilado y generado como un .class

Ejecutamos fuera del paquete p11 la orden: java -Djava.compiler=NONE p11.Vector4 n

- Para k=3:

Repeticiones = 1000; => Medidas en microsegundos (las recogidas en la tabla han sido convertidas a milisegundos)

Tamaño	Tiempo (msg)
10000	0,094
30000	0,250
90000	0,750
270000	2,250
810000	6,766
2430000	20,379
7290000	61,354
21870000	184,770
6561000	578,523

- Para k=4:

Repeticiones = 1000; => Medidas en microsegundos (las recogidas en la tabla han sido convertidas a milisegundos)

Tamaño	Tiempo (msg)
10000	0,094
40000	0,344
160000	1,328
640000	5,344
2560000	21,455
10240000	85,809
40960000	345,291

¿Razone si los tiempos obtenidos son los que se esperaban de la complejidad lineal O(n) de la operación suma?

Los tiempos obtenidos en la operación suma son los que se esperaban porque al multiplicar el tamaño del problema en una constante “k”, los tiempos de ejecución fueron multiplicados por dicha constante “k”, es decir, se demuestra que la operación suma es de complejidad lineal.

A partir de lo visto en Vector4.java midiendo los tiempos de suma, realice las tres siguientes Clases: Vector5.java para medir los tiempos del máximo, Vector6.java para medir los tiempos de coincidencias1 y Vector7.java para medir los tiempos de coincidencias2.
Con los tiempos obtenidos (en milisegundos) de las clases anteriores, rellene las dos tablas:

TABLA 1 (tiempos en milisegundos y SIN_OPTIMIZACIÓN)

Pondremos "FdT" para tiempos superiores al minuto

n	t suma	t maximo
10000	0,078	0,093
20000	0,172	0,187
40000	0,343	0,375
80000	0,656	0,781
160000	1,344	1,625
320000	2,672	3,079
640000	5,36	6,079
1280000	10,689	12,268
2560000	21,348	24,52
5120000	43,176	49,211
10240000	85,855	98,503
20480000	173,205	197,309
40960000	354,839	393,16
81920000	700,678	786,409

TABLA 2 (tiempos en milisegundos y SIN_OPTIMIZACIÓN)

Pondremos "FdT" para tiempos superiores al minuto

n	t coincidencias1	t coincidencias2
10000	672	0,125
20000	2672	0,219
40000	10658	0,453
80000	42713	0,891
160000	FdT	1,766
320000	FdT	3,532
640000	FdT	7,064
1280000	FdT	14,173
2560000	FdT	28,567
5120000	FdT	57,1
10240000	FdT	114,25
20480000	FdT	228,512
40960000	FdT	457,093
81920000	FdT	913,771

Procesador:	Intel(R) Core(TM) i7-8750H CPU @ 2.20GHz
Memoria RAM:	12 GB

Una vez rellenadas ambas tablas, concluya si los tiempos obtenidos cumplen con lo esperado, dada la complejidad temporal computacional de las diferentes operaciones.

Los tiempos obtenidos cumplen con lo esperado aunque ciertos tiempos no demuestren con total precisión la afirmación dicha porque en la medición de dichos tiempos se perdieron decimales.
Quitando ese detalle, los tiempos obtenidos cumplen con lo esperado, para Vector6 la complejidad es n^2, luego es el programa que más tarda en terminar su ejecución y es el que, para un tamaño n que se duplica en cada iteración y la complejidad cuadrática mencionada, los tiempos en cada iteración son multiplicados en un factor de 4.

Para Vector4, Vector5 y Vector7 la complejidad es lineal, luego, si en cada iteración el tamaño n se duplica y la complejidad de dichos programas es lineal, en cada iteración el tiempo es multiplicado por un factor de 2.