Nombre: _____

Ejercicio 1 (2 puntos (10 x 0.2):

Tanto los Set como los List permiten representar colecciones de elementos, pero ¿cuáles son las dos diferencias principales entre los Set y los List?

Si en un programa queremos definir un TreeSet<Alumno> habrá que tener en cuenta que

a) La clase Alumno tendrá que implementar Comparable, de lo contrario se producirá error

b) La clase Alumno tendrá que implementar Comparable y tendrá que tener reescrito el método hashCode(), de lo contrario se producirá error

c) Si la clase Alumno no implementa Comparable, no dará error, pero las búsquedas se realizarán más lentas.

d) Todas las colecciones necesitan que la clase miembro de la colección implemente equals(), hashCode() y compareTo()

Tenemos nombres almacenados en una colección y comprobamos si contiene determinado nombre usando el método contains. ¿Con cuál de las siguientes clases la búsqueda se hará aproximadamente en el mismo tiempo si la colección tiene 1000 nombres que si tiene 1000000 de nombres?

a) TreeSet<String>

b) HashSet <String>

c) ArrayList<String>

d) LinkedList<String>

Hemos definido un Map<Alumno, Profesor> tutores, para almacenar quién es el Profesor tutor de cada Alumno. Para poder trabajar con el Map de forma adecuada:

a) Alumno tiene que reescribir adecuadamente equals y hashCode.

b) Profesor tiene que reescribir adecuadamente equals y hashCode

c) Alumno y Profesor tienen que reescribir adecuadamente equals y HashCode

d) Dependiendo de la clase que usemos para el Map, Alumno tendrá que implementar/reescribir unos métodos u otros

Si c es una colección de 100000 Strings, con cuál de las siguientes definiciones de c sería más eficiente (tardaría menos) la operación c.remove(0)

a) ArrayList<String> c;

b) LinkedList<String> c;

c) TreSeet<String> c;

d) HashSet<String> c

Si c es una colección de 100000 Strings, con cuál de las siguientes definiciones de c sería más eficiente (tardaría menos) la operación c.get(25000)

a) ArrayList<String> c;

b) LinkedList<String> c;

c) TreSeet<String> c;

d) HashSet<String> c

Si queremos escribir un entero en un fichero binario usaremos un objeto:

DataOutputStream f = new DataOutputStrem (new FileOutputStream(“datos.bin”))

Explica brevemente por qué es necesario usar dos clases DataOutputStream y FileOutputStream y no basta con usar DataOutputStream

¿Qué hace la instrucción File f = new File("carta.txt")?

a) Crea el fichero carta.txt en la carpeta actual, solo si no existe ya en el disco

b) Crea el fichero carta.txt en la carpeta actual. Si el fichero ya existe lo reescribe, borrando y el contenido que tuviera previamente se pierde.

c) Crea un objeto que representa la ruta y nombre del fichero. El fichero podría no existir en disco.

d) Crea un objeto que representa la ruta y nombre del fichero. Si el fichero no existe en el disco, se producirá la excepción FileNotFoundException

¿Cuándo se trabaja con ficheros, qué diferencia hay entre utilizar un try-catch "normal" y un try-catch "with resources"?

¿Si f es un objeto de la clase File, que devuelve f.length?

a) Si f es un archivo, su tamaño en bytes. Si f es una carpeta, el tamaño de los archivos que contiene en bytes.

b) Si f es un archivo, su tamaño en bytes. Si f es una carpeta, un valor indeterminado, que no representa lo que ocupa la carpeta.

c) Si f es un archivo, su tamaño en bytes. Si es una carpeta, el número de archivos que la carpeta contiene.

d) Si f es de tipo File, no podrá ser una carpeta. Devuelve el tamaño en bytes del archivo.

Ejercicio 2 (3 puntos): El instituto participa en una carrera benéfica con el siguiente funcionamiento:

- A cada alumno/a lo patrocina un profesor/a.
- Un profesor/a puede patrocinar a varios alumnos/as
- El alumnado da vueltas al perímetro del centro. Por cada vuelta se entregará un euro al fin benéfico.
- El profesorado patrocinador será quien tenga que pagar el importe correspondiente a las vueltas de sus patrocinados.
- Es posible que haya corrido alumnado no patrocinado.

Disponemos de

patrocinios: un Map<String, String> que contiene parejas (alumno/a que corre, profesor/a que le patrocina) . Este map permite saber qué profesor/a patrocina a cada alumno/a.

vueltas: Un list con nombres de alumnos/as. Cada vez que un alumno/a ha completado una vuelta, se ha añadido su nombre al List

Se pide, completar el método mostrarPagos para que, dados el Map y el List descritos, muestre por pantalla, con un formato como el del ejemplo:

- Cuánto tiene que pagar cada profesor/a
- Cuántas vueltas han dado alumnos/as que **no** están patrocinados

Apoyarse en un Map para contar lo que tiene que pagar cada profesor.

ExEv3.ColeccionesFicheros.Solucion

Con los datos introducidos en el método main, la salida esperada sería la siguiente:

```
Esther tiene que pagar 12 euros
Javi tiene que pagar 4 euros
Monica tiene que pagar 7 euros

Vueltas dadas por alumnado sin patrocinar: 9
```

Ejercicio 3 (2.5 puntos): Para calcular la nota final un curso se hace la media de dos evaluaciones y a ésta se le restan 0,1 puntos por cada falta de asistencia.

Disponemos de un fichero de texto (evaluaciones.txt) con dos líneas de información por cada alumno/a.

- En la primera línea está su nombre
- En la segunda línea, separado por blancos o tabuladores, la nota de la primera, la nota de la segunda y el número de faltas.

Se pide:

a) Escribir en otro fichero de binario (finales.dat) el nombre y la nota final de cada alumno. La relación de alumnos y notas tiene que estar **ordenada** por nota de menor a mayor. A igual nota, se ordenará por nombre. Para ordenar puedes crear las clases que estimes conveniente.

b) Modificar el programa para que funcione con un fichero con el formato que tiene evaluaciones2.txt. Este fichero no contiene el número de faltas cuando el alumno/a no ha tenido faltas.

Ejercicio 4 (2.5 puntos): Escribir un programa que permita al usuario comparar el contenido de dos carpetas:

El usuario introducirá por teclado el nombre de dos carpetas. El programa mostrará el nombre de las subcarpetas comunes, y el nombre de los archivos **no** comunes.

Solo se tendrán en cuenta carpetas y archivos de primer nivel (no subcarpetas de subcarpetas, etc…)

Para facilitar la prueba, en el proyecto tienes dos “folder”: carpeta1 y carpeta2. Si el usuario indicara éstas en la ejecución, la información mostrada por el programa sería (el orden no tiene por qué coincidir):

Carpetas comunes

- facturas
- música
- videos

Archivos no comunes

- fichero1.txt
- fichero4.txt