Detalles de la tarea de esta unidad.

Enunciado.

Si hasta el momento hemos utilizado en todos los casos estructuras estáticas (arrays) para almacenar datos, llega el momento de aprovechar las bondades de las estructuras dinámicas y la funcionalidad del API **Collections**.

Ejercicio 1

En la tarea de la Unidad de Trabajo 7 utilizamos un array para almacenar las cuentas bancarias. Esta estructura provoca que nuestra aplicación esté limitada a utilizar 100 cuentas. Modifica dicho proyecto para:

1. Utilizar la estructura de datos dinámica arrayList.
2. Añade las siguientes opciones al menú que se implementarán con un método:

    - Eliminar Cuenta Bancaria. A través de esta opción se pedirá el CCC de una cuenta bancaria y se eliminará de la estructura siempre que exista y su saldo sea 0. No se podrán eliminar cuentas con saldo superior a 0.
    - Mostrar el número de Cuentas de ahorro que hay en la lista. Se implementará con un método que devolverá el número de cuentas bancarias que son de la clase CuentaAhorro.
    - Mostrar el saldo acumulado de todas las cuentas corrientes con una comisión de mantenimiento superior a un valor introducido por teclado.
    - Ordenar la lista de cuentas bancarias por saldo y mostrar por pantalla los nombres de los 3 titulares más ricos. Consulta el apartado 7.1 de los contenidos.

Ten en cuenta que deberás utilizar el operador **instanceof** para determinar si un objeto está instanciado como CuentaAhorro o como CuentaCorriente.

En cuanto a la organización del código, se deberán conservar las clases del proyecto anterior substituyendo la clase ArrayCuentas por ListaCuentas:

- Principal: Contendrá el método main y todo el código de interacción con el usuario (lectura de teclado y salida por pantalla).
- ListaCuentas: mantendrá como atributo la estructura que almacena las cuentas, el constructor y métodos solicitados.
- Todas las clases necesarias para representar la jerarquía de cuentas.
- Otras clases que pudieran ser de utilidad.

Ejercicio 2

Escribe un programa que generará números de forma aleatoria entre 0 i 14 y los irá almacenando en un conjunto HashSet. Para cada número se mostrará por pantalla un mensaje de si el número ha sido incluido en el conjunto o si no ha podido incluirse en caso de repetición. El programa finalizará al tener el conjunto ocho números. Para finalizar, mostrar por pantalla el % de colisiones (números que no se han podido añadir al conjunto)