****

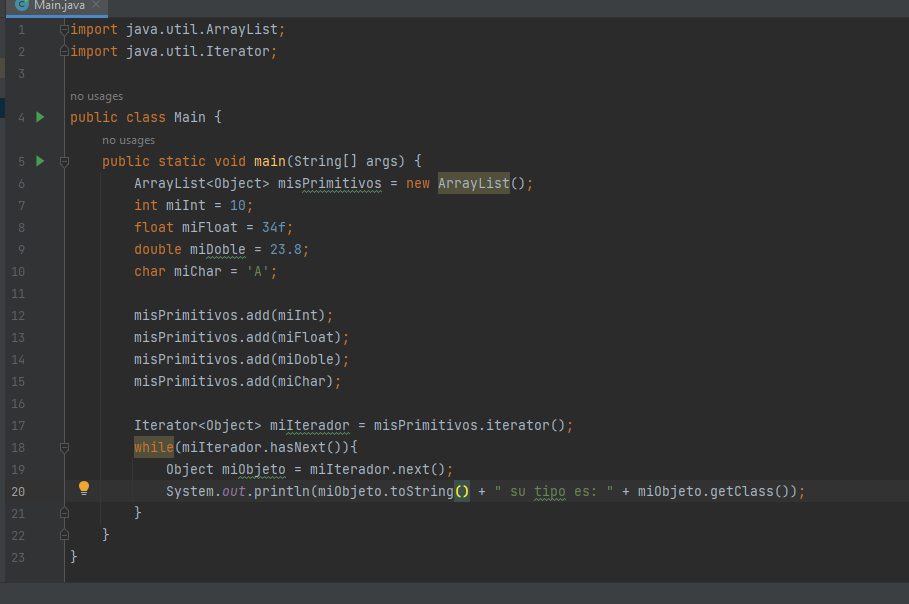
***Activad UT6.1***

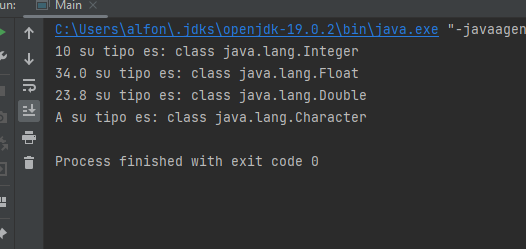
***Programación***

*Alfonso García Jorge*

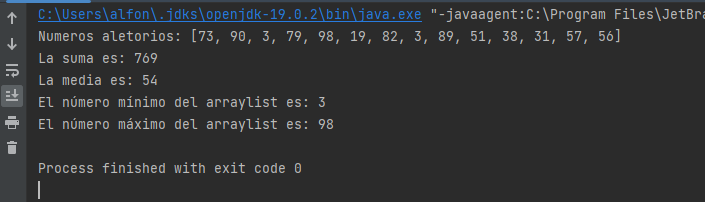
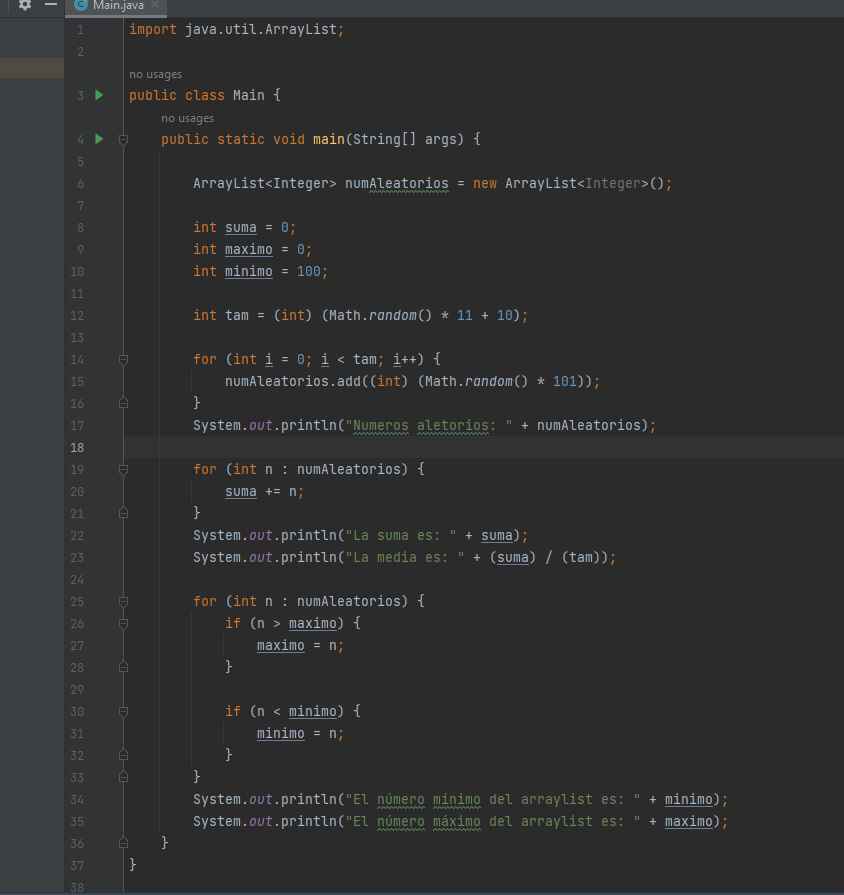
*1ºDAWNA*

1.- Implemente una claseArrayList, que permita mediante el método add(tipo\_primitivo) agregar a la lista elementos de tipos primitivos int, float y double, y que automáticamente los cargue con sus respectivas “clases envolventes”.

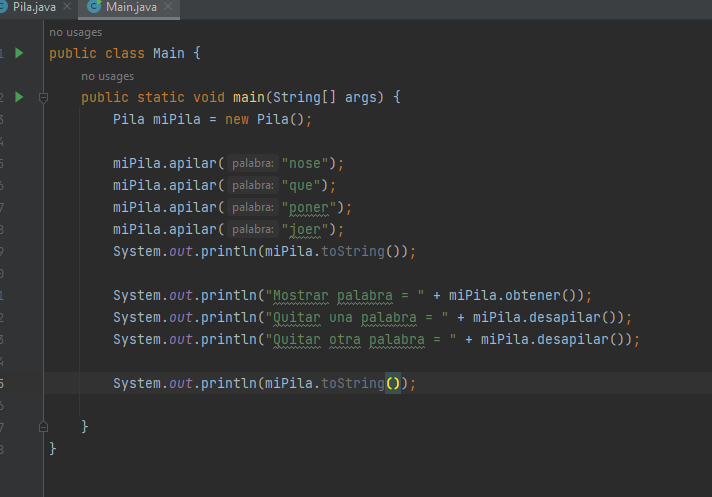


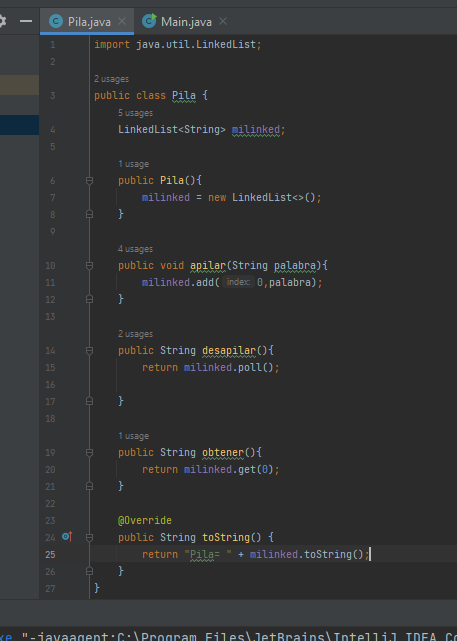


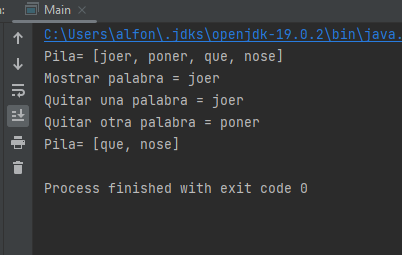
2.- Realiza un programa que introduzca valores aleatorios (entre 0 y 100) en un ArrayList y que luego calcule la suma, la media, el máximo y el mínimo de esos números. El tamaño de la lista también será aleatorio y podrá oscilar entre 10 y 20 elementos ambos inclusive.



3.- Escribir una clase, de nombre PilaPalabras, para gestionar una estructura de pila que permita apilar y desapilar objetos de la clase String. La clase implementará el método apilarPalapra para poner una palabra encima de la pila, el método desapilarPalabra para quitar el elemento de la cima de la pila de devolviéndolo y el método obtenerPalabra para obtener la palabra situada en la cima de la pila sin quitarla de ella. También implementará el método pilaPalabrasVacia para determinar si la pila está o no vacía. Los métodos deben implementarse utilizando la clase LinkedList







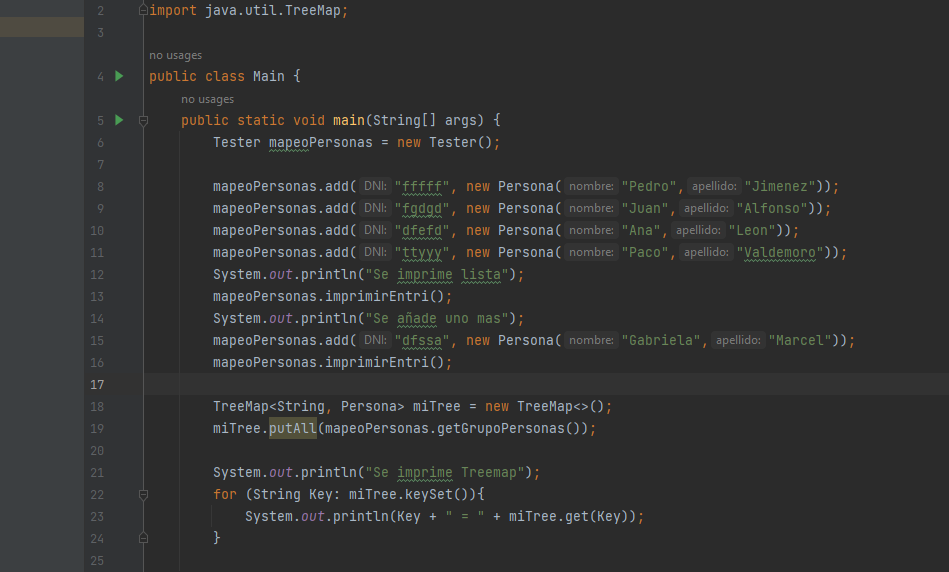
4.- Implementa una clase Ficha (ficha de dominó) con su constructor. Después implementa una clase tester con su constructor y su toString() que genere una Lista de 6 fichas al azar y las imprima ordenadas ascendentemente y descendentemente de acuerdo a un “peso ponderado” (que consistirá en el número más alto de la ficha multiplicado por 6 más el otro número). Para eso implementa también el método pesoPonderado(). Realizar el ejercicio con un TreeSet considerando que las fichas del dominó no se pueden repetir.

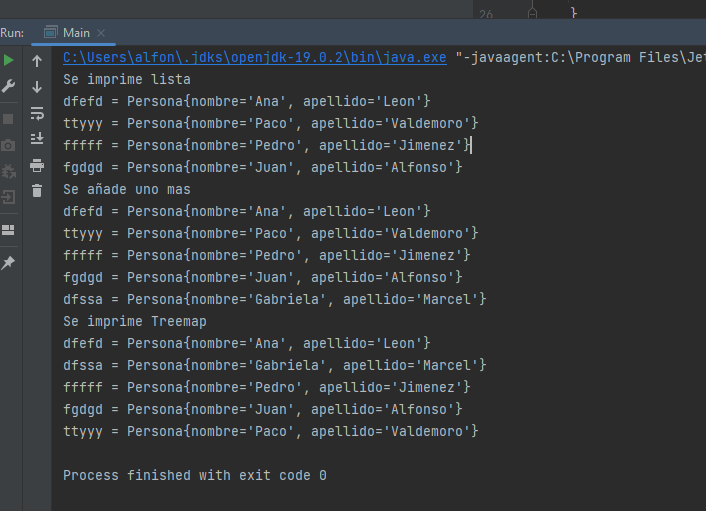
5.- Implementa una clase Persona (o Person) con atributos nombre y apellido con su constructor, sus getters y su toString. Después implementa una clase tester que añada unas cuantas personas en un HashMap<String, Person> (donde la key es el NIF de la persona).

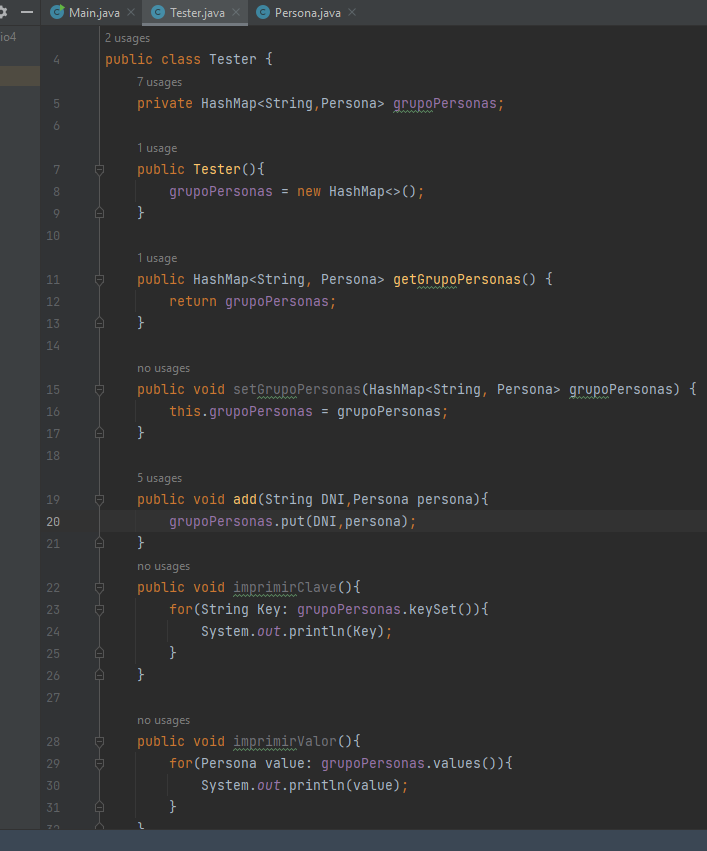
Esta clase tester deberá tener métodos que impriman: las keys, los values y las entries del mapa.

Y comprobar que:

* Si intentas obtener un valor con una llave que no existe devuelve null.
* Si intentas añadir una entrada <llave, valor> con una llave ya existente, esta entrada sobrescribe la pre-existente.
* Si generamos un TreeMap a partir del HashMap sus entradas estarán ordenadas por key.







6.- Escribir un programa Java que crea un HashSet de Objetos de tipo Coche. El programa pide por teclado los datos de los coches y los guarda en el HashSet. A continuación utilizará un iterator para mostrar por pantalla lo siguiente:

* Todos los coches introducidos.
* Todos los coches de una marca determinada.
* Todos los coches con menos de un número determinado de Kilómetros.
* El coche con mayor número de Kilómetros.
* Todos los coches ordenados por número de kilómetros de menor a mayor.

7.- Crea un programa en Java para calcular el String de mayor longitud de todos los contenidos en un ArrayList de String. El programa utilizará los siguientes métodos:

* Método leerArray(): este método recibe como parámetro el arrayList de Strings vacío. El método pide por teclado cadenas de caracteres y las añade al ArrayList. La lectura de cadenas termina cuando se introduce la palabra “FIN”.
* Método cadenaMasLarga():este método recibe como parámetro el ArrayList de Strings con todas las cadenas leídas anteriormente y devuelve el String de mayor longitud.

