Clases especiales

String

1. Realizar un programa que muestre los códigos ASCII así como el carácter correspondiente para los símbolos desde el 32 hasta el 126, ambos incluídos (son los símbolos imprimibles del código ASCII)

NOTA: Los métodos que se requieren a continuación, para que sean fáciles de probar desde un "main", han de ser **estáticos**.

2. Realizar y probar un método llamado "esVocal(char):boolean", que dado un **char** me devuelva un **boolean** que valga **true** en el caso de que el **char** proporcionado sea vocal (mayúscula o minúscula), y **false** en caso contrario.
3. Realizar y probar un método llamado "esMayuscula(char):boolean", que dado un **char** me devuelva un **boolean** que valga **true** en el caso de que el **char** proporcionado sea cualquier letra mayúscula, y **false** en caso contrario.
4. Realizar y probar un método llamado "esMinuscula(char):boolean", que dado un **char** me devuelva un **boolean** que valga **true** en el caso de que el **char** proporcionado sea cualquier letra minúscula, y **false** en caso contrario.
5. Realizar y probar un método llamado "aMayuscula(char):char", que dado un **char** me devuelva ese mismo **char** pero en mayúscula. Si el **char** inicial no era una letra minúscula, se queda igual que estaba.
6. Realizar y probar un método llamado "aMinuscula(char):char", que dado un **char** me devuelva ese mismo **char** pero en minúscula. Si el **char** inicial no era una letra mayúscula, se queda igual que estaba.
7. Realizar y probar un método ""numMinusculas(String):int, que dado un **String**, me devuelva un **int** que represente el número de minúsculas contenido en dicho String.
8. Realizar y probar dos métodos "numMayusculas(String):int", "numVocales(String):int", que cuenten y devuelvan el número de mayúsculas y vocales, respectivamente, en un determinado String.
9. Realizar y probar un método "numVecesChar(String,char):int", me devuelva el número de veces que aparece el **char** proporcionado en el **String** proporcionado.
10. Realizar y probar un método "quitaEspacios(String):String", que dado un **String**, me devuelva el mismo **String**, pero sin espacios en blanco.
11. Realizar y probar un método "reves(String):String", que dado un **String**, me devuelva el mismo **String**, pero al revés.
12. Realizar y probar un método "charAMay(String,char):String", que dado un **String,** devuelva el mismo **String**, en el que las apariciones del **char** proporcionado han sido sustituidas por ese mismo char, pero en mayúsculas (si no lo estuviera).

13. Realizar y probar un método “vocAMay(String):String”, que dado un **String,** devuelva el mismo **String**, en el que todas las vocales que contenga aparezcan en mayúsculas.
14. Realizar y probar un método “vocalizacion(String,char):String”, que dado un **String**, devuelva otro en el que aparecen sustituidas todas las vocales que contenga, por el char proporcionado.
15. Realizar y probar un método “generoYNumero(String):String”, que dado un **String**, devuelva otro **String** cuyos valores pueden ser “masculino singular”, “masculino plural”, “femenino singular” o “femenino plural”, en función de la terminación del **String** inicial. Hágase para casos básicos de palabras “normales”
16. Realizar y probar un método “quitaTildes(String):String”, que dado un **String** me devuelva el mismo, pero sin tildes en las vocales que pudiera contener. Para averiguar los códigos de ASCII extendido correspondientes con las tildes, ajustar el ejercicio 1 para mostrar los códigos desde el 128 hasta el 255.
17. Realizar y probar un método “esPalindromo(String):boolean”, que dado un **String**, me devuelva un **boolean**, que valga true si el **String** proporcionado es un palíndromo y **false** en caso contrario. Conseguir que el método ignore los espacios, sea “inmune” a tildes y no sea sensible a mayúsculas.

Object

18. Crear una clase Persona con atributos nombre, ape1, ape2 y nacionalidad, con un constructor que permita introducir estos datos al crear objetos de la clase. Ajustar toString() para ver la información de personas en el formato **ape1 ape2, nombre (nacionalidad)**

Date

19. Realizar un método **esperar()**, que espere (no haga nada) exactamente durante un segundo, y otro método sobrecargado **esperar(int)**, que se valga del primero para esperar el número de segundos indicado. Para realizar el primero, hacer un par de bucles anidados cuyo bucle interno haga 10.000 vueltas fijas. Para fijar el número de vueltas que se han de hacer en el externo, para que tarde un segundo exactamente, se ha de ajustar manualmente y este ajuste dependerá de la velocidad de la CPU donde se esté ejecutando y de la carga que tenga en ese momento. Por tanto, este ajuste se hará probando “esperar()” desde otro método y fijando un Date antes de invocarlo y otro después y utilizando el método getTime(), para obtener la diferencia de tiempos.
20. Utilizar el método **esperar()** anterior, para crear un **cronómetro** que mida minutos y segundos. Pista: mostrar el número de minutos y segundos transcurridos desde que arranca el programa. Actualizar dicho número en un bucle infinito, tras cada segundo. El API de JAVA para manejo de entrada / salida básica es muy escueto y no dispone de ningún comando para “limpiar la pantalla”. Se puede simular con un for que realice unos cuantos println() vacíos.