Arquitectura y Diseño de sistemas 2020

Trabajo Práctico: Kotlin

1. Hello World.
2. Implementar una función que indique si un número entero es par.
3. Año bisiesto: Implementar una función que indique si un año dado es bisiesto. Algoritmo
4. Imprimir la suma de los primeros n enteros.
5. Imprimir los 1ros n múltiplos de i y j. Por ejemplo, 3, 5, 6, 9, 10, 12, 15 para n=17, i=3, j=5.
6. Dado un número entero n, imprimir su tabla de multiplicación hasta el 12.
7. Implementar una función que indique si un número entero es primo.
8. Fizz Buzz: Escribir un programa que imprima los números del 1 al 100. Pero, para los múltiplos de 3 imprimir "Fizz" en vez del número, y para múltiplos de 5 imprimir "Buzz". Para los números que son multiples de 3 y de 5, imprimir "FizzBuzz".
9. Implementar una función dividir dos enteros n, m. Retornar un Float. Si m es 0, retornar NaN. No utilizar if o when.
10. Caesar cipher: Implementar el cifrador César, tanto para codificar como decodificar. Algoritmo
11. Implementar la función `createPyramid(rows:Int)`
    Ej:
    ```
    createPyramid(4)
    ```
    ```
          *
        * * *
      * * * * *
    * * * * * * *
    ```
12. La función de librería kotlin.collections.joinToString tiene valores por defecto en la declaración de sus parámetros:

    ```
    fun joinToString(
        separator: String = ", ",
        prefix: String = "",
        postfix: String = "",
        /* ... */
    ): String
    ```

    La función se puede llamar sobre una colección de Strings. Especificando solo dos argumentos, implementar la función: `joinOptions(array: Array<String>): String` que retorna una lista de Strings en formato JSON (por ej, "[a, b, c]").
13. Dado el siguiente método sobrecargado en java:

    ```
    public String foo(String name, int number, boolean toUpperCase) {
        return (toUpperCase ? name.toUpperCase() : name) + number;
    ```

```
}
public String foo(String name, int number) {
    return foo(name, number, false);
}
public String foo(String name, boolean toUpperCase) {
    return foo(name, 42, toUpperCase);
}
public String foo(String name) {
    return foo(name, 42);
}
```

Remplazarlos por una sola función en kotlin.

14. Refactorizar las funciones Ceasar Cipher del ejercicio 10 para que sean funciones de extension de la clase String.
15. Implementar la función de extensión removeFirstLastChar:

```
fun main(args: Array<String>) {
    val myString = "Hello Everyone"
    val result = myString.removeFirstLastChar()
    println("Resultado: $result")
}
```

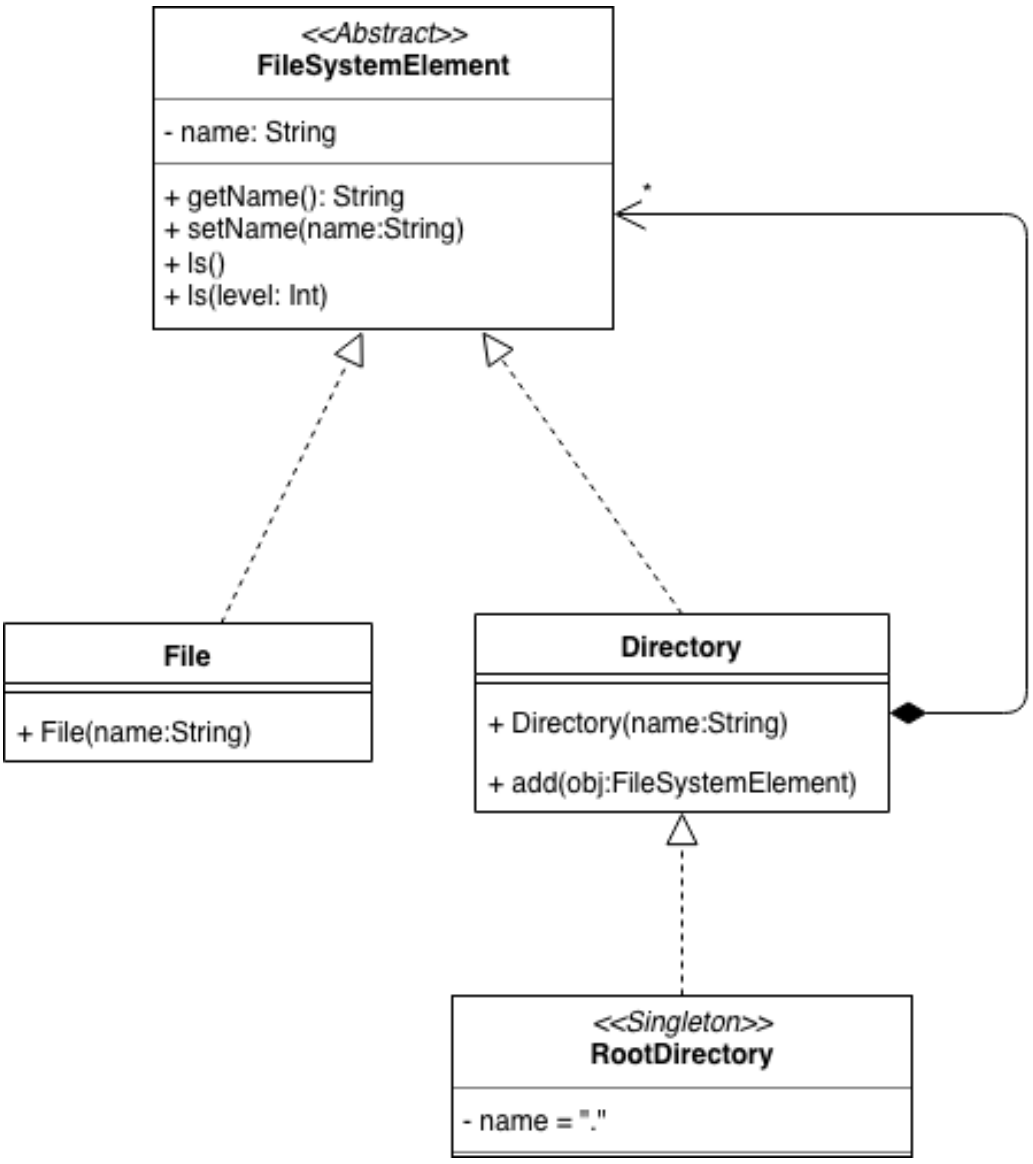
Resultado: ello Everyon

16. Implementar la función de extensión removeFirstLastDigit

```
fun main(args: Array<String>) {
    val myNum = 24556
    val result = myNum.removeFirstLastDigit()
    println("Resultado: $result")
}
```

Resultado: 455

17. Implementar la función infija "veces". Por ejemplo, "3 veces 2" retorna 6.
18. Definir una clase Pirámide. Implementar la función createPyramid del ejercicio 11 como función infija de la misma.
19. Implementar el siguiente diagrama de clases:



Crear la estructura de datos de manera que al ejecutar `RootDirectory.ls()` se liste lo siguiente por consola:

```
- ./
- - A/
- - - K/
- - - - f1.dat
- - - - R/
- - - - - f2.dat
- - - - - f3.dat
- - - L/
- - - - f4.dat
- - B/
- - - P/
- - - - f5.dat
- - - - f6.dat
- - - Q/
```

20. Ordenar una lista de strings por orden alfabético
21. Ordenar la misma lista por longitud de cada string.
22. Ordenar la misma lista por cantidad de caracteres "e".
23. Implementar una función que indique si un número entero es primo utilizando listas. (Modificar el ejercicio 7 eliminando el for)
24. Implementar un a función que tome una lista de strings y los muestre por consola dentro de un frame rectangular. Por ejemplo, con la lista ``Hello'', ``World'', ``in'', ``a'', ``frame''] se imprime:

```
*********
* Hello *
* World *
* in    *
* a     *
* frame *
*********
```

25. Implementar el ejemplo de libros y autores (class Book). Además de mostrar la lista de autores, mostrar la lista de nombres de libros de un autor en particular.
26. Dada la estructura de datos de archivos del ejercicio 19, implementar una función que liste los nombres de todos los archivos del sistema.
27. Implementar una función que nos permita buscar un archivo dado un subtérmino. Imprimir sólo los nombres de los archivos.
28. Who took the cookie?. Implementar un algoritmo que imprima la letra de la canción. El orden de los animales debe ser random. Hint: utilizar shuffled, reduce.
29. Simplificar el siguiente algoritmo utilizando safe call operators:

```
data class A(val b: String?, val c: Int?, val d: List<Boolean?>?)

fun testNullables() {

  val a1 = A("B1", null, null)
  val a2 = A(null, 2, null)
  val a3 = A(null, null, listOf(null, false, true, null))
  val a4 = A(null, null, null)

  val listA = listOf(a1, a2, a3, a4)

  listA.forEach {
    if (it.b != null) {
      println(it.b)
    } else if (it.c != null) {
      println(it.c)
    } else if (it.d != null) {
      it.d.forEach {
        if (it != null) {
          print(it)
        }
      }
      println()
    } else {
      println("all null")
    }
  }
}

Output:
B1
2
falsetrue
all null
```