FUNDAMENTOS DE PROGRAMACIÓN

Proyecto Entregable – 2º cuatrimestre: Entrega 01

Las entregas en este segundo cuatrimestre consisten en la implementación en Java de los mismos problemas que nos han acompañado durante el primer cuatrimestre. De esta manera, podremos comprobar que lo relevante a la hora de programar son las ideas que utilizamos más que el lenguaje de programación en sí.

A continuación, se explican los pasos y ejercicios a realizar en este primer entregable.

1. Funciones

Construya dos clases de nombres *Funciones* y *TestFunciones* respectivamente. La primera es una clase con métodos estáticos y estará en un paquete de nombre *fp.funciones* y la segunda es la correspondiente clase de testeo y debe estar en un subpaquete de nombre *fp.funciones.test*. Cada método de la primera clase tendrá otro método asociado en la segunda donde se comprobará su correcto funcionamiento a través de varios ejemplos.

Se pide[1]:

1. Dado un número entero, determinar si es un número primo o no.
2. Dados los números enteros n, k con $n \geq k$ diseñar una función que calcule el número combinatorio $\binom{n}{k}$.
3. Dados los números enteros n, k con $n \geq k$ diseñar una función que calcule el número $S(n,k)$ dado por:

$$S(n,k) = \frac{1}{k!}\sum_{i=0}^{k}(-1)^i \binom{k}{i}(k-i)^n$$

4. Dada una lista con números enteros, diseñar una función que devuelva otra lista con las diferencias entre cada valor y el anterior.
5. Dada una lista de cadenas de caracteres, diseñar una función que devuelva la cadena más larga.

2. Diseño de tipos

Cree un tipo inmutable – es decir, se puede programar mediante un *record* – de nombre *Fecha* en un paquete de nombre *fp.tipos*. De manera análoga al ejercicio anterior, se programará una clase de nombre *TestFecha* en un subpaquete de nombre *fp.tipos.test* con el testeo asociado a cada funcionalidad.

En el paquete *java.time* se encuentra el tipo *LocalDate* que se puede utilizar para trabajar con fechas en Java. En este ejercicio no se puede utilizar este tipo de Java en los métodos de Fecha. Sin embargo, la manera de hacer los métodos de testeo de la clase *TestFecha* puede ser comprobando que la funcionalidad es la misma – o casi la misma – en el tipo Fecha y el tipo *LocalDate*.

Descripción del tipo

[1] Estas funcionalidades son las mismas que se programaron en la primera entrega del primer cuatrimestre.

1. **Fecha (inmutable)**

Propiedades:
- *año*, de tipo *Integer*, propiedad básica. Indica el año de la fecha.
- *mes,* de tipo *Integer*, propiedad básica. Indica mes de la fecha.
- *dia,* de tipo *Integer*, propiedad básica. Indica el día de la fecha.
- *nombreMes,* de tipo *String*, propiedad derivada. Indica el nombre del mes asociado. Esta propiedad se calcula a partir del valor del mes y de una lista auxiliar con los nombres de los doce meses.
- *diaSemana*, de tipo *String*, propiedad derivada. Indica el nombre del día de la semana. Esta propiedad se basa en el algoritmo de zeller, que se programará como un método estático que se indica más adelante del enunciado. Dicho método devuelve un valor entero que, junto con una lista auxiliar con los días de la semana, permitirá programar esta propiedad.
- *sumarDías*, de tipo *Fecha*, propiedad derivada. Dado un número entero con un número de días, indica la fecha resultante de sumar ese número de días a la fecha. Esta propiedad se basa en el método estático *díasEnMes* que se describe más adelante.
- *restarDias,* de tipo *Fecha,* propiedad derivada. Dado un número entero con un número de días, indica la fecha resultante de sumar ese número de días a la fecha. De manera análoga al método anterior, esta propiedad se basa en el método *díasEnMes*.
- *diferenciaEnDias*, de tipo *Integer*, propiedad derivada. Dada una fecha, indica el número de días que hay entre la fecha dada como parámetro y el objeto fecha asociado.

Representación como cadena:

Ejemplo: "Viernes, 6 de Octubre de 2023". Es decir, nombre del día de la semana, seguido del valor del día, junto al nombre del mes y año asociado como se puede ver en el ejemplo.

Criterio de igualdad: dos fechas son iguales sí coinciden su día, mes y año.

Orden natural: por defecto

Métodos de factoría:
- *of(año:Integer, mes:Integer, dia:Integer):* construye un objeto fecha a partir del año, mes y día indicados como parámetros.
- *parse(cadenaFecha: String):* construye un objeto fecha a partir de una cadena de caracteres con la información del año, mes y día. El formato de la cadena es similar al siguiente ejemplo: "2023-09-20".

    - Comentarios: los métodos de factoría son, por definición, métodos estáticos.

Orden natural: No tiene.

Otros métodos:
- *esAñoBisiesto(año: Integer)*: dado un determinado año, indica si es o no un año bisiesto.
- *díasEnMes(año:Integer, mes:Integer)*: dado un año y un mes, indica el número del mes de dicho año.
- *congruenciaZeller (año: Integer, mes: Integer, dia: Integer)*: dado un año, un mes y un día, indica el día de la semana asociado a dicha fecha.

_*_

Ayuda de programación:

a) Recupere el código del tipo Fecha implementado en Python y lea la algoritmia de los métodos más complicados. Es decir, refactorice el código de Python en Java.