Ejercicio 1

La siguiente expresión es una descripción recursiva para la integral de una función f entre dos puntos a y b.

$$\int_a^b f(x)dx = \int_a^m f(x)dx + \int_m^b f(x)dx$$

Se trata de diseñar un **algoritmo recursivo** en el lenguaje C que calcule la integral aproximada de la función ya implementada en C tal que su prototipo es ***double f(double x),*** sabiendo que m=(a+b)/2 y que se trata de dividir el intervalo [a,b] por la mitad hasta que sea lo bastante pequeño |b-a| < epsilon, en cuyo caso se acepta que

$$\int_a^b f(x)dx \cong (b-a) * f(m).$$

SE PIDE:

1. Diseñar el correspondiente algoritmo recursivo
2. ¿Cuál es el tamaño del problema?
3. ¿Calcular T(n) para el caso mejor y peor del algoritmo diseñado para hallar la integral aproximada?

Tiempo estimado: 30 min. Puntuación: 3.0 puntos

Ejercicio 2

La sucesión de **Jacobsthal-Lucas** es la secuencia de números enteros grandes P(n) definida por los siguientes valores iniciales

$$P(0) = 2,\ P(1) = 1$$

y la siguiente relación de recurrencia

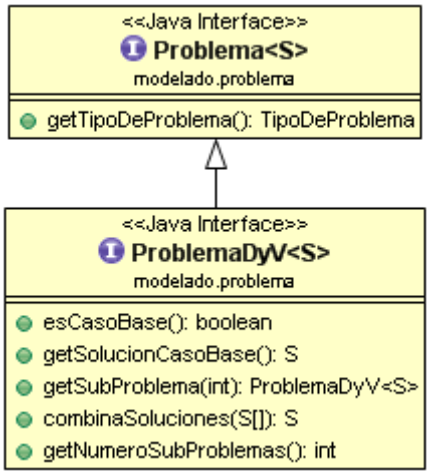
$$P(n) = P(n-1) + 2 * P(n-2)$$

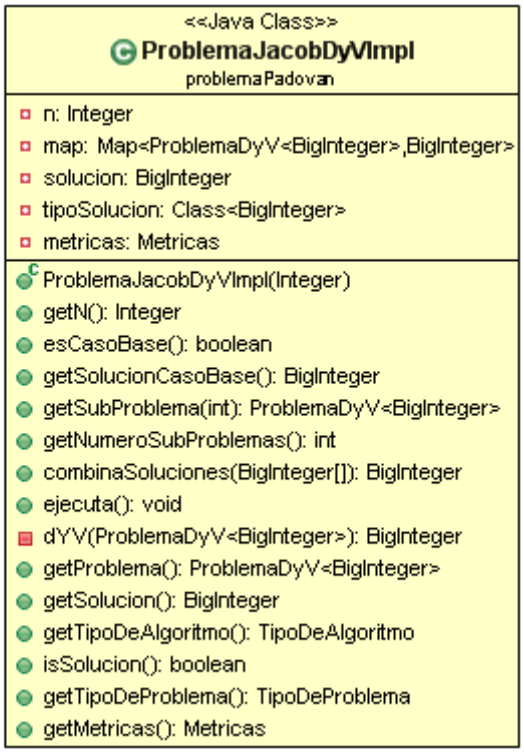
Los primeros valores de P(n) son: 2, 1, 5, 7, 17, 31, 65, 127, 257, 511, ...

Se pide:

a) Rellene la Ficha adjunta (**que deberá entregar al finalizar**) siguiendo la información del problema de Jacobsthal-Lucas detallado anteriormente. Debe decidir cuál es la mejor opción para la técnica de Divide y Vencerás en este problema concreto: ¿con o sin memoria? Justifíquelo brevemente.

b) Implemente los métodos pertenecientes a la clase `ProblemaJacobDyVImpl`: para ello, ayúdese de los diagramas UML proporcionados a continuación:





a. `private BigInteger dYV(ProblemaDyV<BigInteger> p)`. Resuelve la técnica Divide y Vencerás.
b. `public boolean esCasoBase()`. Indica si el problema que se está resolviendo es tan simple que no necesita recursión para su resolución.
c. `public BigInteger getSolucionCasoBase()`. Devuelve la solución al problema, resuelto de forma directa.
d. `public int getNumeroSubProblemas()`. Indica el número de subproblemas.
e. `public ProblemaDyV<BigInteger> getSubproblema(int i)`. Devuelve el subproblema iésimo en el que se descompone el problema que se está resolviendo.
f. `public BigInteger combinaSoluciones (BigInteger[] soluciones)`. Combina cada solución de la lista de soluciones de manera que juntas resuelvan el problema completo. Puede utilizar las operaciones de BigInteger: *add(BigInteger a):BigInteger* y *multiply(BigInteger a):BigInteger*.

Ayuda: algunas operaciones de BigInteger son: BigInteger add(BigInteger val) – BigInteger multiply(BigInteger val) – BigInteger div(BigInteger val)

Tiempo estimado: 45 min Puntuación: 3.5 puntos

Ficha	
Sucesión de Jacobsthal-Lucas	
Técnica: ***// Rellenar*** *Breve justificación (con o sin memoria):* ***// Rellenar***	
Representación: ***Un objeto por problema***	
Tamaño: ***N***	
Propiedades Compartidas	***// Rellenar***
Propiedades Individuales	***// Rellenar***
Solución: ***BigInteger***	
// Rellenar	
Complejidad	***// Rellenar***