**1. Escribir, al menos el pseudocódigo correspondiente al método o a los métodos identificados**

En este caso vamos a distinguir dos métodos principales, el primero que va a ser el constructor de la clase Fecha, el cual nos va a permitir comprobar que los datos de entrada son los correctos y si no elevar una excepción. Y el segundo método, llamado "Esbisiesto" el cual comprueba que el objeto fecha sea un año bisiesto.

Códgio:

```
    public FechaP(int dia, int mes, int anno) throws ExcepcionFecha {
        try {
            if (dia <= 0 || dia > 31){
                throw new ExcepcionFecha("dia invalido");
            }
            if (mes <= 0 || mes > 12) {
                throw new ExcepcionFecha("mes invalido");
            }
            if (anno <= 0 ) {
                throw new ExcepcionFecha("año invalido");
            }
        }catch (InputMismatchException e){
            System.err.print(e);
        }
        this.dia = dia;
        this.mes = mes;
        this.anno = anno;

    }
```

```
public boolean Esbisiesto(FechaP fecha) {
    if ((fecha.getAnno() % 4 == 0) && ((fecha.getAnno() % 100 != 0) || (fecha.getAnno() % 400 == 0))) {
        System.out.println("Es bisiesto");
        return true;
    }else {
        System.out.println("No es bisiesto");
        return false;
    }
}
```

**2. Identificar las variables que se deben tener en cuenta para probar el método de interés.**

En este caso las variables a tener en cuenta son: **día, mes y año**, las cuales son números enteros

**3. Identificar los valores de pruebas para cada una de las variables anteriores usando las tres técnicas vistas en teoría, especificando para cada una cual es la que ha sido usada.**

**Leyenda de los valores de prueba:**

**Rojo: Conjetura de Errores.**

**Azul: Variante ligera**

**Naranja: Variante pesada.**

| Parámetros | Clases de Equivalencia | Valores de Prueba |
|---|---|---|
| Día | (-∞, 0]U(0, 31]U(31,+∞) | **0, 31, -1, 1, 30, 32, Null** |
| Mes | (-∞, 0]U(0, 12]U(12, +∞) | **0, 12, -1, 1, 11, 13, Null** |
| Año | (-∞, 0)U([0, +∞)-{Xmod4, Xmod100, Xmod400})U(Xmod4, Xmod100, Xmod400) | **0, 4, 100, 400, -1, 1, 3, 5, 99, 101, 399, 401, Null** |

Las clases de equivalencia quedan bastante claras en el caso de los días y de los meses, no obstante, en el caso de los años debemos tener en cuenta que según la lógica del programa esperamos un comportamiento diferente con los numero entre menos infinito y 0, otro comportamiento con los números entre 0 e infinito que nos sean múltiplos de 4, 100 y 400, y otro comportamiento para los números múltiplos de 4, 100 y 400.

**4. Calcular el número máximo posible de casos de pruebas que se podrían generar a partir de los valores de pruebas (combinatoria).**

Para calcular el número máximo de casos de prueba que se pueden generar, debemos tener en cuenta el número de valores de prueba por cada parámetro en el apartado anterior. Para los parámetros Día y Mes, tenemos 7 valores de prueba, por lo que entre ellos, podría generar un total de 49 casos de prueba, y teniendo en cuenta el parámetro año, que tiene 13 valores de prueba, podríamos generar un total de **637 casos de prueba diferentes.**

**5. Defina un conjunto de casos de pruebas para cumplir con each use (cada valor una vez)**

Para cumplir con el criterio each use, tenemos que usar cada uno de los valores de prueba una vez, por lo tanto, el **conjunto de prueba** estará formado por un total de **13 casos de prueba.**

El conjunto de casos de prueba puede ser (considerando que el formato es (día, mes, año)):

1.(0, 0, 0)

2. (31, 12, 4)

3.(-1, -1. 100)

4.(1, 1, 400)

5.(30, 11, -1)

6.(32, 13, 1)

7.(Null, Null, 3)

8.(0, 0, 5)

9.(31, 12, 99)

10.(31, 12, 101)

11.(31, 12, 399)

12.(31, 12, 401)

13.(31, 12, Null)

**6. Defina conjuntos de pruebas para alcanzar cobertura pairwaise usando el algoritmo explicado en clase.**

Para saber el número de casos de prueba del **conjunto de prueba** que podemos obtener siguiendo el criterio pair-wise, lo que tenemos que hacer es multiplicar el numero de valores de prueba para las dos variables con más valores de prueba en este caso 13 y 7, por lo tanto tendríamos que generar un total de 91 casos de prueba, se adjunta un txt con los casos de prueba.

**7. Para los trozos de código que incluyan decisiones, proponga conjunto de casos de prueba para alcanzar cobertura de decisiones.**

En este apartado para proponer los casos de prueba, vamos a separarlos en dos dependiendo el método sobre el que se vayan a ejecutar (el constructor o "Esbisiesto"). Para alcanzar la cobertura de decisiones, lo que tenemos que hacer es proponer un conjunto de casos de prueba que evalúen a true y false todas las condiciones del programa.

**Constructor:**

En el caso del constructor, tenemos 3 sentencias que se pueden evaluar a verdadero o falso, el primer caso de prueba propuesto es este: (1, 1, 100) el cual evalúa las 3 condiciones a verdadero, por lo tanto, solo nos queda proponer casos de prueba para evaluarlas a falso:

-evaluar a falso el día: (-1, -1, 100)

-evaluar a falso el mes: (1, -1, 100)

-evaluar a falso el año: (1, 1, -1)

Y además deberíamos de tener en cuenta un caso de prueba con valores que pudiesen elevar la excepción del constructor, como por ejemplo ("a", 1, 100)

Por lo tanto para conseguir el nivel de cobertura de decisiones en el método constructor, usaremos los siguientes casos de prueba (1, 1, 100); (-1, -1, 100);(1, -1, 100);(1, 1, -1);("a", 1, 100)

**"Esbisiesto":**

En este caso contamos solo con una sentencia condicional, por lo que para cumplir con la cobertura de decisiones, necesitaremos 2 casos de prueba.

El primero para evaluar la condición a verdadero: (1, 1, 100)

Segundo para evaluar la condición a falso: (1, 1, 99)

Como podemos observar el número de casos de prueba que vamos a utilizar para alcanzar el nivel de cobertura de decisiones es de 5, ya que existe un caso de prueba común para los métodos Constructor y "Esbisiesto", de tal modo que con el mismo caso de prueba podemos probar diferentes partes del sistema.

**8. Para los trozos de código que incluyan decisiones, proponga conjunto de casos de prueba para alcanzar cobertura MC/DC.**

Para alcanzar la cobertura múltiple de condiciones/decisiones lo que tenemos que hacer es proponer casos de prueba en los que, cada decisión venga determina por cada una de las condiciones posibles. Para ello vamos a construir una tabla de verdad de la expresión a evaluar.

| A | B | C | A\|\|(B && C) |
|---|---|---|---|
| 1 | 1 | 1 | 1 CP1 |
| 1 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 0 | 0 | 1 |
| 0 | 1 | 1 | 1 CP3 |
| 0 | 1 | 0 | 0 CP2 |
| 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 0 |

La expresión A corresponde con: año es divisible entre 4.

La expresión B corresponde con años es divisible entre 100.

La expresión C corresponde con años es divisible entre 400.

Hemos seleccionado un total de 3 casos de prueba, CP1 en el que la decisión viene totalmente determinada por la condición A, CP2 en el que la condición queda determinada por A y C, y CP3, en el que la salida viene determinada por B y C.

Los casos de prueba podrían ser los siguientes:

-CP1: día = 1, mes = 1, año = 4, de tal modo que A B y C se evalúan a verdadero y la decisión también.

-CP2: día = 1, mes = 1, año= 100, de tal modo que A y C se evalúan a falso y la decisión no.

-CP3: día = 1, mes = 1, año = 400, de tal modo que B y C se evalúan a verdadero y la decisión también.

**9. Comente los resultados del número de los casos de pruebas conseguidos en los apartados 4, 5 y 6 ¿qué podría decirse algo de la cobertura alcanzada?**

Dependiendo del conjunto de casos de prueba que seleccionemos el grado de cobertura será mejor o peor dependiendo del número de casos de prueba, o de la técnica utilizada para la generación de conjunto de casos de prueba.

Si utilizásemos los valores de prueba generados por el apartado 4, tendríamos que realizar un total de 637 casos de prueba diferente, lo que muy probablemente nos llevase a un nivel de cobertura muy alto, no obstante, realizar 637 casos de prueba puede resultar muy costoso.

En el caso de los conjuntos de casos de prueba generados en el apartado 5, utilizando la técnica de each use, obteníamos un total de 13 casos de prueba, lo que resulta mucho más sencillo de codificar, no obstante, el nivel de cobertura alcanzable si aplicásemos estos casos de prueba sería bastante bajo, por lo tanto, utilizar este conjunto casos de prueba se ha descartado.

En el caso del apartado 6, hemos obtenido un total de 91 casos, de prueba, este número parece más razonable en el caso, además con esta técnica y aplicando el conjunto de casos de pruebas generado, se alcanza un buen nivel de cobertura, y el número de casos de prueba, encuentra un equilibrio entre el numero de casos de prueba, pues no es un número muy bajo como el caso de la técnica each use, ni tampoco un número desorbitado de casos de prueba como sería para el caso de generar casos de prueba con combinatoria.

Conclusión:

A pesar de esto, de entre los 3 apartados que tenemos que comentar, si tuviéramos que elegir uno, elegiríamos la cobertura PairWaise si tuviéramos tiempo. En cambio, si nos corriera más prisa nos quedaríamos con la cobertura each-use gracias a la gran cantidad de casos de prueba cubiertas. En definitiva, nunca nos decantaríamos por hacer el máximo número de pruebas debido al gran número de casos de prueba que se requieren.