UNIVERSIDAD DE CASTILLA-LA MANCHA

ESCUELA SUPERIOR DE INFORMÁTICA

GRUPO BC.05

TRABAJO TEÓRICO PROBLEMA 3

Trabajo realizado por:

Isaac González del Pozo

Natalia Jiménez Zapata

Asignatura: Ingeniería del Software II

Grupo: BC.05

Titulación: Grado en Ingeniería Informática

1. Código correspondiente al método

```
import java.util.*;


public class Aniobisiesto {
    static Scanner TECLADO = new Scanner (System.in);
    public static void main(String[] args) throws Exception {
        //String ntrada;
        int dia = 0, mes = 0, anio = 0;
        boolean esBisiesto;
        Fecha f = new Fecha();
        System.out.println("Introduce una fecha por teclado...");
        System.out.println("Dia: ");
                dia = introducirDato();
                f.setDia(dia);
        System.out.println("Mes: ");
            mes = introducirDato();
            f.setMes(mes);
        System.out.println("AÃ±o: ");
            anio = introducirDato();
            f.setAnio(anio);

            f.toString();

        esBisiesto= calcularbisiesto(anio);
        if(esBisiesto){
            System.out.println("El aÃ±o es bisiesto");

        }else{
            System.out.println("El aÃ±o no es bisiesto");
        }


    }

    public static int introducirDato() throws Exception{
        int dato=0;
        try{
         dato = TECLADO.nextInt();
        }
        catch(InputMismatchException i){
            System.out.println("Se ha introducido un valor no numerico, reinicie el programa");

        }
        return dato;
    }
    public static boolean calcularbisiesto(int anio){
        if(anio % 4 == 0 && anio % 100 != 0 || anio % 400 == 0){
            return true;
        }else{
            return false;

        }
    }

}
```

```java
public class Fecha {

    private int dia;
    private int mes;
    private int anio;

    public void Fecha() throws Exception {
        this.setDia(dia);
        this.setMes(mes);
        this.setAnio(anio);
    }

    public int getDia() {
        return dia;
    }

    public void setDia(int dia) throws Exception {
        try {
            if(dia <= 0 || dia > 31) {
                throw new NumberFormatException("El nÃºmero no puede ser mayor que 31");
            }
            this.dia = dia;

        }catch(Exception e) {
            System.out.println("El dia no es correcto, reinica el programa");
            System.exit(0);
        }

    }

    public int getMes() {
        return mes;
    }

    public void setMes(int mes) throws Exception{

        try {
            if(mes <= 0 || mes > 12) {
                throw new NumberFormatException("El nÃºmero no puede ser mayor que 12");
            }
            this.mes = mes;
        }catch(Exception e) {
            System.out.println("El mes no es correcto, reinica el programa");
            System.exit(0);
        }
    }

    public int getAnio() {
        return anio;
    }


  public void setAnio(int anio) throws Exception{

      try {
          if(anio <= 0) {
              throw new NumberFormatException("El nÃºmero no puede ser menor que 0");
          }
          this.anio = anio;
      }catch(Exception e) {
          System.out.println("El aÃ±o no es correcto, reinica el programa");
          System.exit(0);
      }
  }


  public String toString() {
      System.out.println("La fecha introducida es: " + dia + "/" + mes + "/" + anio);
      return "fecha [dia=" + dia + ", mes=" + mes + ", anio=" + anio + "]";
  }
```

2. Identificar las variables que se deben tener en cuenta

Se deben tener en cuenta las siguientes variables:

- Día = {1,31} = Indica los días de la semana los cuales tendrán la limitación con la fecha, es decir, desde el día 1 del mes hasta el día 31
- Mes = {1,12} = Indica los meses del año, al igual que la anterior, tiene la limitación desde el mes de enero hasta diciembre.
- Anio = {0,1000000} = Indica el año que posteriormente haremos uso de ello para saber si es bisiesto o no
- Bisiesto={true, false} = variable que utilizaremos junto con la variable de año para sacar si el año es bisiesto o no

3. Indicar los conjuntos de valores de prueba para cada variable

PARÁMETROS	VALORES OBJETIVOS	CLASES DE EQUIVALENCIA	VALOR LIMITE	CONJETURA DE ERRORES	VALORES TOTALES
DIA	(1,31)	INT (-∞,1),[1,31],(31, ∞)	1,31	-1,32	-1. 0, 1, 31, 32, 35
MES	(1,12)	INT (-∞,1),[1,12],(12, ∞)	1,12	-1,13	-1. 0, 1, 12,13,25
ANIO	(0,1000000)	INT(∞,0],(0,1000000),[1000000, ∞)	0,1000000	-1,1000001	-1. 0, 1, 1000000,1000001, 2000000
DATO	(-∞,∞)	(-∞,∞)	-	-	-1,0,1
BISIESTO	(true,false)	BOOLEAN true,false	True, false	-	True, false

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

Tres de nuestros parámetros tienen 6 valores totales, otro de nuestros parámetros tiene tres valores y por último un parámetro con 2 valores. Con lo cual se nos quedaría un número máximo de posibles casos de prueba será $6^3 * 2 * 3 = 1296$ casos de prueba

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

Para cumplir each use tenemos que fijarnos que parámetro tiene más valores totales, para ello nos fijamos en los tres parámetros comentados anteriormente consiguiendo así el número de casos de prueba para poder realizar each use, estos son:

(-1, -1, -1, -10, true)

(0, 0, 0, 1, false)

(1, 1, 1, -10, true)

(31, 12, 1000000, 1, false)

(32, 13, 1000001, -10, true)

(35, 25, 2000000, 1, false).

6. Defina conjuntos de pruebas para alcanzar cobertura pairwaise usando el algoritmo explicado en clase. Se pueden comprobar los resultados con el programa PICT2

El pairwaise se define como todos los pares los casos de prueba deben visitar, al menos una vez, todos los pares de valores de dos parámetros cualesquiera .En nuestro caso hemos decidido usar los parámetros día y mes para hacer todas las combinaciones posibles entre ellos dos.

Día	Mes
-1	-1
-1	0
-1	1
-1	12
-1	13
-1	25
0	0
0	1
0	12
0	13
0	25
0	-1
1	1
1	12
1	13
1	25
1	-1
1	0
31	12
31	13
31	25
31	-1
31	0
31	1
32	13
32	25
32	-1
32	0
32	1
32	12
35	25
35	-1
35	0
35	1
35	12
35	13

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

Método para introducir el día

- Día <= 0 → A
- Dia > 31 → B

```
public void setDia(int dia) throws Exception {

if(dia <= 0 || dia > 31) {

                                    throw new NumberFormatException("El numero no puede ser
mayor que 31");

        }

}
```

A	B	A OR B
0	0	0
0	1	1
1	0	1
1	1	1

Método para introducir el mes

- Mes <= 0 → C
- Mes > 12 → D

```
public void setMes(int mes) throws Exception{

if(mes <= 0 || mes > 12) {

                                    throw new NumberFormatException("El nÃºmero no puede
ser mayor que 12");

                            }

}
```

C	D	C OR D
0	0	0
0	1	1
1	0	1
1	1	1

Método para introducir el año

- Año <= 0 → E → 0 ó 1

Método para saber si el año es bisiesto o no

- esBisiesto → F → true o false

Método para calcular el año bisiesto

- anio%4 == 0 → G
- anio % 100 != 0 → H
- anio % 400 == 0 → I

```
public static boolean calcularbisiesto(int anio){
        if(anio % 4 == 0 && anio % 100 != 0 || anio % 400 == 0){
            return true;
        }else{
            return false;

        }
    }
```

G	H	I	G AND H OR I
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Simplificando todos los casos de prueba posibles podemos a escoger los siguientes para tener una cobertura completa

{1,1,1,2]

[-1,1,2,3}

{0,1,2,2}

{1,2,2,2}

{-1,3,2,5}

{1,4,2,2}

{1,8000,0,0}

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

Método para establecer el día

```
public void setDia(int dia) throws Exception {
if(dia <= 0 || dia > 31) {
                                    throw new NumberFormatException("El numero no puede ser
mayor que 31");
        }
}
```

A	B	A OR B	CONDICION DOMINANTE
0	0	0	A, B
0	1	1	B
1	0	1	A
1	1	1	A, B

Método para establecer el mes

```
public void setMes(int mes) throws Exception{
if(mes <= 0 || mes > 12) {
                                    throw new NumberFormatException("El nÃºmero no puede
ser mayor que 12");
                                }
}
```

C	D	C OR D	CONDICION DOMINANTE
0	0	0	C, D
0	1	1	D
1	0	1	C
1	1	1	C, D

Método para calcular el bisiesto

```
public static boolean calcularbisiesto(int anio){
		if(anio % 4 == 0 && anio % 100 != 0 || anio % 400 == 0){
			return true;
		}else{
			return false;

		}
	}
```

G	H	I	G AND H OR I	CONDICION DOMINANTE
0	0	0	0	G, H, I
0	0	1	1	I
0	1	0	0	H
0	1	1	1	H, I
1	0	0	0	G
1	0	1	1	G, I
1	1	0	1	G, H
1	1	1	1	G, H, I

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?

En el apartado 4 en el que tenemos todos los posibles casos de prueba vamos a obtener una cobertura del 100%, que son todas las posibilidades que puede existir en el sistema.

En el caso 5 no alcanzaremos una cobertura tan satisfactoria como la del apartado 4 ya que tendremos tantos casos de uso como valores haya en la variable que más posibles valores tenga.

Por último, del ejercicio 6 podemos concluir que va a haber muy poca cobertura, ya que solo vamos a coger todas las posibilidades, pero únicamente de entre dos variables cualquiera del sistema, por lo que, al no cogerlas toda la cobertura será ínfima.