UNIVERSIDAD NACIONAL DE LOJA

Área de la Energía, las Industrias y los Recursos Naturales No Renovables

CARRERA DE INGENIERÍA EN SISTEMAS

Nombre: Byron Montaño **Fecha:** 19-05-2019

Ciclo: 6 ”A”

CONGRUENCIAL MIXTO EN JAVA

CODIGO

LINK: https://github.com/byronmb/CONGRUENCIAL.git

El generador congruencial mixto se representa de la siguiente manera:

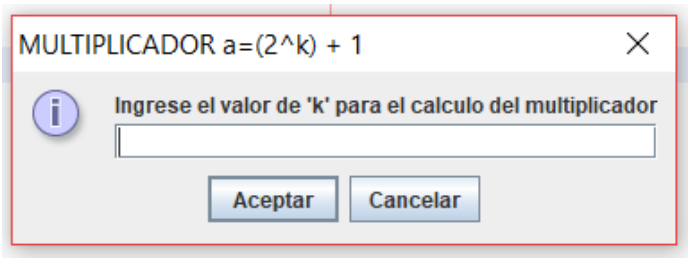
$$X_{n+1} = (aX_n + c) \bmod m$$

- En primer lugar, están definidas las variables principales que se utilizaran para poder realizar el cálculo del generador congruencial mixto. Estas son:
  X0 = la semilla
  a = el multiplicador
  c = constante aditiva
  m = el módulo

```
int a = 0;
int c = 0;
int x0 = 0;
int m = 0;
```

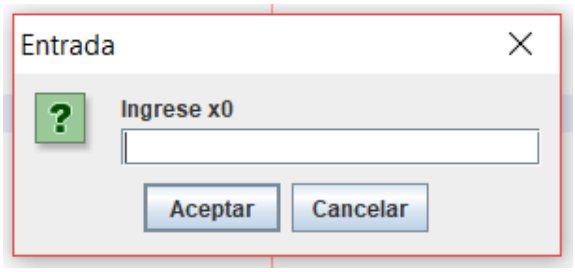
- Para el cálculo del multiplicador “*a*” se lo realiza leyendo una variable “*k*” del usuario basado en la regla que dice que “usualmente se selecciona ‘*a*’ como $2^k + 1$ cuando se trabaja en sistema binario”.
  Además, se controla con el bucle *while* para que el usuario solo pueda ingresar el valor de k mayor a 0.

```
//calculo del multiplicador "a"
while (k <= 0) {
    k = Integer.parseInt(JOptionPane.showInputDialog(null, "Ingrese el valor de 'k' para el calculo del multiplicador", "MULTIPLICADOR \na=(2^k) + 1", JOptionPane.INFORM
    if (k <= 0) {
        JOptionPane.showMessageDialog(null, "El valor de 'k' debe ser mayor a cero");
    }
    a = (int) Math.pow(2, k) + 1;
}
```



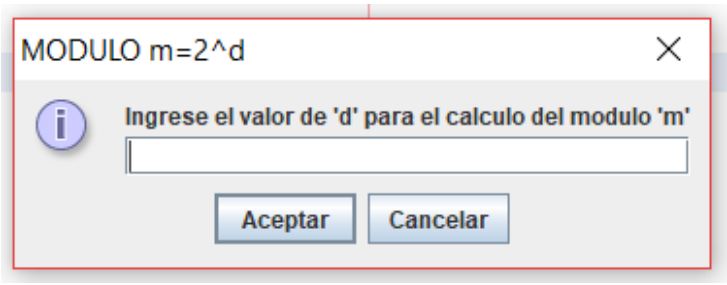
- Para el cálculo de la semilla "*x0*" se lo realiza simplemente leyendo una variable con el mismo nombre, y estableciendo de igual manera que el usuario solo pueda ingresar un valor mayor a cero.

```
//calculo de la semilla "X0"
while (x0 <= 0) {
    x0 = Integer.parseInt(JOptionPane.showInputDialog(null, "Ingrese x0"));
    if (x0 <= 0) {
        JOptionPane.showMessageDialog(null, "El valor de 'x0' debe ser mayor a cero");
    }
}
```



- Para el cálculo del módulo "*m*" se lo realiza leyendo una variable "*d*" del usuario basado en una de las opciones de selección la cual dice " seleccionar '*m'* como p^d ".
  Además, se controla que el valor del módulo sea mayor al multiplicador "a" y a la semilla "x0".

```
//calculo del modulo "m"
int d = 0;
while (m <= a || m <= x0) {
    d = Integer.parseInt(JOptionPane.showInputDialog(null, "Ingrese el valor de 'd' para el calculo del modulo 'm'", "MODULO \nm=2^d", JOptionPane.INF
    m = (int) Math.pow(2, d);
    if (m <= a || m <= c || m <= x0) {
        JOptionPane.showMessageDialog(null, "El valor de 'm' debe ser mayor a los valores de a, c y x0");
    }
}
```

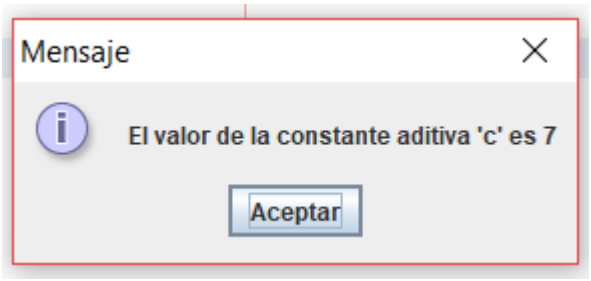


- Para el cálculo de la constante aditiva “c” se lo realiza basado en la regla que dice “el valor de c debe ser un valor impar y relativamente primo a m”.
  Para esto se utilizó un método para el cálculo del número primo más cercano, el cual recibirá como parámetro el valor de m.

```
//calculo de la constante aditiva "c"
c = nPrimo_Cercano(m);
JOptionPane.showMessageDialog(null, "El valor de la constante aditiva 'c' es " + c);
```

```
//Metodo para calcular el numero primo mas cercano
    public static int nPrimo_Cercano(int num) {
        int cont = 2;
        boolean n_primo = true;

        while ((n_primo) && (cont != num)) {
            if (num % cont == 0) {
                n_primo = false;
            }
            cont++;
        }
        if (n_primo) {
            return num;
        } else {
            return nPrimo_Cercano(num - 1);
        }
    }
```



- Para la generación de los numero pseudoaleatorios se la realizo con un ciclo repetitivo el cual se repetirá el valor de m veces, en caso de que la secuencia de los valores posibles se repitan, se controlan con una condición que compara el

primer valor de X_{n+1} con los valores siguientes , en caso de que estos se repitan, el ciclo termina.

Además, se realiza el cálculo de los números pseudoaleatorios mediante la fórmula establecida, y la semilla ira tomando cada valor de estos durante todo el ciclo repetitivo.

```
int n_pseudo = 0;
int cociente = 0;
System.out.println("n" + " | " + "xn" + " | " + a + "Xn+" + c + "/" + m + " | " + "Xn+1" + " | " + "Numeros Uniformes" + "\n");
int aux = x0;
for (int i = 1; i <= m; i++) {
    n_pseudo = ((a * x0) + c) % m; //(aXn+c) mod m
    cociente = ((a * x0) + c) / m; //operacion -> ((a * x0)+c) {divisor}
    System.out.println(i + " | " + x0 + "  |  " + cociente + "+" + n_pseudo + "/" + m + " |   " + n_pseudo + "  |    " + n_pseudo + "/" + m);
    x0 = n_pseudo;
    if (i > 1 && aux == n_pseudo) {
        break;
    }
}
```

PRESENTACION DE LOS RESULTADOS

```
run:
a-->5
x0-->4
m-->8
c-->7
n | xn | 5Xn+7/8 | Xn+1 | Numeros Uniformes

1 | 4  |  3+3/8 |   3  |    3/8
2 | 3  |  2+6/8 |   6  |    6/8
3 | 6  |  4+5/8 |   5  |    5/8
4 | 5  |  4+0/8 |   0  |    0/8
5 | 0  |  0+7/8 |   7  |    7/8
6 | 7  |  5+2/8 |   2  |    2/8
7 | 2  |  2+1/8 |   1  |    1/8
8 | 1  |  1+4/8 |   4  |    4/8
BUILD SUCCESSFUL (total time: 2 minutes 7 seconds)
```