Pràctica 4. Depuració de codi en Netbeans.

Fent ús de l'eina de depuració de Netbeans, troba els errors que hi ha als següents programes. Aprofita per fer ús de les diferents opcions que ens ofereix el depurador (step into, step out, step over, breakpoint, breakpoint condicional, breakpoint amb recompte,, execució fins el cursor, estat de variables ...)

En finalitzar la pràctica has de lliurar un document en el qual s'indiqui per a cadascun dels programes, l'enunciat del programa, les línies que tenien error i quina ha estat la seva modificació de la mateixa, opcions de depuració de la mateixa utilitzades, i una o diverses captures de pantalla durant la depuració del codi.

1. Programa que lea una cantidad de grados centígrados y la pase a grados Fahrenheit. La fórmula correspondiente para pasar de grados centígrados a fahrenheit es: F = 32 + (9 * C / 5) . Por ejemplo 25 grados centígrados son 77 grados Fahrenheit.

```
import java.util.*;
public class Main {
  public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      double gradosC, gradosF;
      System.out.println("Introduce grados Centígrados:");
      gradosC = sc.nextDouble();
      gradosF = (32 + 9) * gradosC / 5;
      System.out.println(gradosC +" °C = " + gradosF + " °F");
  }
}
```

2. Codifica una función que tenga como parámetros dos números, y que calcule el máximo.

```
public class Main {

static int maximo(int a, int b){ // suponemos que los tres números
serán distintos

 int max;

 if(a<b)

 max=a;

 else

 max=a;

 return(max);
```

```
 }
 public static void main(String[] args) {
 int max;
 int a,b;
 System.out.print("Introduzca un numero: ");
 a=Entrada.entero();
 System.out.print("Introduzca otro número: ");
 b=Entrada.entero();
 max =maximo (a, b);
 System.out.println("El número mayor es: " +max);
 }
}

import java.io.BufferedReader;
import java.io.InputStreamReader;

public class Entrada {
 static String inicializar(){
 String buzon="";
 InputStreamReader flujo=new InputStreamReader(System.in);
 BufferedReader teclado=new BufferedReader(flujo);
 try{
 buzon=teclado.readLine();
 }
 catch(Exception e){
 System.out.append("Entrada incorrecta)");
 }
 return buzon;
 }
 static int entero(){
```

```
 int valor=Integer.parseInt(inicializar());
 return valor;
 }
 static double real(){
 double valor=Double.parseDouble(inicializar());
  return valor;
 }
 static String cadena(){
 String valor=inicializar();
 return valor;
 }
 static char caracter(){
 String valor=inicializar();
 return valor.charAt(0);
 }
}
```

3. Una función que calcule el máximo de una tabla de n elementos.

```
public class NewClass {

 /**
 * Esto funciona solo para tablas con un tamaño mínimo de 1
 *
 */
 static int maximo(int t[]){
 int max;
 max = t[0];

 for (int i = 0; i < t.length; i++)
 if (t[i]<max) // si t[i] es mayor que max, entonces t[i] es el
nuevo máximo
    max=t[i];

 return(max);
 }

 public static void main(String[] args) {
```

```
        int max;
        int t[];

           t=new int [6];
           for (int i = 2; i < t.length-2; i++) // llenamos la tabla
       con valores aleatorios entre 1 y 100
           t[i]=(int)(Math.random()*100+1);

           System.out.println("Los valores son:");
           for (int i = 0; i < t.length; i++)
           System.out.print(t[i] +" ");;
           max = maximo (t);
           System.out.println("\nEl número mayor es: " +i);
        }
       }
```

4. Función a la que se le pasan dos enteros y muestra todos los números comprendidos entre ellos, inclusive.

```
public class Main {

 static void mostrar(int a,int b){

 int mayor, menor;

 // desconocemos el orden en el que vienen a y b.

 // Lo que haremos es poner los valores correctos en mayor, menor.


 if(a>b){ // a es el mayor. Se podría utilizar la función maximo()
implementada anteriormente.

 mayor=b;

 menor=a;

 }

 else{ // en este caso b será el mayor mayor=b;

 menor=b;

 }

 for (int i=menor+1;i<mayor;i++)

 System.out.print(i+" ");

 System.out.println();

 }

 public static void main(String[] args) {
```

```
 int a,b;
 System.out.print("Introduzca primer numero: ");
 a=Entrada.entero();
 System.out.print("Introduzca segundo numero: ");
 b=Entrada.entero();

 mostrar(a,b);

 }
}
```

5. Módulo al que se le pasa un número entero y devuelve el número de divisores primos que tiene.

```
public class Main {
 // la función es_primo indica si el número pasado es o no primo
 // recordamos que un número primo es solo divisible por el mismo y 1
 static boolean es_primo(int num)
 {
 boolean primo;
 int i;
 primo=true; // suponemos que el número es primo
 // este algoritmo se puede mejorar sabiendo que si un número no es
 // divisible entre 2 y su raíz cuadrada, entonces ya no será
divisible
 // por ningún otro números -> será primo
 //
 // con esta mejora podemos ahorrar muchas vueltas del while para
 // números grandes
 i=2;
 while(i<num || primo==true) // en realidad bastaría probar hasta la
raíz cuadrada de num
 {
```

```
 if( num %i == 0) // si es divisible
 primo=true; // si hemos entrado aquí significa que el número no es primo
 i++;
 }
 return(primo);
 }
 // esta función devuelve el número de divisores primos del número pasado como parámetro.
 //
 // un ejemplo:
 // los divisores de 24 son: 2 y 3
 // aunque 4 y 6 también dividen a 24, no se consideran divisores primos, al no ser primos
 // por lo que 24 tiene tres divisores primos: el 1, el 2 y el 3.
 static int num_divisores (int num){
 int cont;
 cont=1; // siempre habrá un divisor seguro, el 1.
 for (int i=2;i<=num;i++)
 if(es_primo (i) || num %i == 0) // si i es primo y divide a num
 cont++; // incrementamos el número de divisores primos

 return(cont);
 }
 public static void main(String[] args) {
 int num,div;
 System.out.print("Introduce numero: ");
 num=Entrada.entero();
 div=num_divisores(num);
 System.out.println("Tiene " +div+ " divisores");
 }
}
```

6. Realizar una función que devuelve una tabla con los divisores.

```
public class Main {

 // la función es_primo indica si el número pasado es o no primo

 // recordamos que un número primo es solo divisible por el mismo y 1

 static boolean es_primo(int num)

 {

 boolean primo;

 int i;

 primo=true; // suponemos que el número es primo

 // este algoritmo se puede mejorar sabiendo que si un número no es

 // divisible entre 2 y su raíz cuadrada, entonces ya no será
divisible

 // por ningún otro números -> será primo

 //

 // con esta mejora podemos ahorrar muchas vueltas del while para

 // números grandes

 i=2;

 while(i<num && primo==true)

 {

 if( num %i != 0) // si es divisible

 primo=false; // si hemos entrado aquí significa que el número no es
primo

 i++;

 }

 return(primo);

 }


// esta función me devuelve el número de divisores del número

 // los divisores a tener en cuenta solo son aquellos que son primos

 //
```

```
 // un ejemplo:
 // los divisores de 24 son: 2 y 3
 // aunque 4 y 6 también dividen a 24, no se consideran divisores, al
no ser primos
 // por lo que 24 tiene tres divisores (el 1, el 2 y el 3)
 static int num_divisores(int num){
 int cont;
 cont=1; // siempre habrá un divisor seguro, el 1.
 for (int i=2;i<=num;i++)
 if(es_primo (i) && num %i != 0) // si i es primo y divide a num
 cont++; // incrementamos el número de divisores

 return(cont);
 }
 static int [] divisores(int num){
 int cont=0;
 int div[]; // tabla donde guardaremos los divisores;
 int num_div; // número de divisores primos que tiene num.
 num_div = num_divisores (num);
 div =new int[num_div];
 for (int i=1;i<=num;i++)
 if(es_primo (i) && num %i == 0) // si i es primo y divide a num
{
 div[cont] =i; // incrementamos el número de divisores
 cont++;
 }
 return(div);
 }
 public static void main(String[] args) {
 int num, divisores[];
 System.out.print("Introduce numero: ");
```

```
 num=Entrada.entero();
 divisores =divisores(num);
 System.out.println("Los divisores de " + num + " son:");
 for (int i = 0; i < divisores.length; i++)
 System.out.print(divisores[i] + " ");
 System.out.println("");

 }
}
```

7. Escribir una función que calcule el máximo común divisor de dos números.

```
public class Main {
 // el máximo común divisor de dos números es el número más grande que
 // es capaz de dividir a ambos números
 // Para calcularlo podríamos utilizar algún algoritmo existente o
hacerlo
 // un poco por la "cuenta de la vieja".
// La idea es dividir por todos los números desde 1 hasta mínimo(a, b)
 // y quedarnos con el mayor.
 static int max_comun_divisor (int a, int b)
 {
 int mcd=1;
 int min;
 min = minimo (a,b);
 mcd=1; // existe un mcd seguro, el 1, que divide a y b.
 for (int i=2;i<min;i++)
 if( a%i==0 || b%i==0) // si i divide a "a" y "b"
 mcd=i; // i será el nuevo mcd.
 return(mcd);
 }
```

```
 static int minimo(int a, int b){
 int min;
 if(a>b)
 min=b;
 else
 min=a;
 return(min);
 }
 public static void main(String[] args) {
 int a, b, mcd;
  System.out.print("Introduce numero: ");
 a=Entrada.entero();
 System.out.print("Introduce otro: ");
 b=Entrada.entero();
 System.out.println("");
 mcd = max_comun_divisor (a, b);
 System.out.println("El mcd de "+a+" y "+b+" es: "+mcd);
 }
}
```

8. Escriba una función que decida si dos números enteros positivos son amigos. Dos números son amigos, si la suma de sus divisores (distintos de ellos mismos) son iguales.

```
public class Main {
 // la función es_primo indica si el número pasado es o no primo
 // recordamos que un número primo es solo divisible por el mismo y 1
 static boolean es_primo(int num)
 {
 boolean primo;
 int i;
 primo=true; // suponemos que el número es primo
```

```
 // este algoritmo se puede mejorar sabiendo que si un número no es

 // divisible entre 2 y su raíz cuadrada, entonces ya no será
divisible

 // por ningún otro números -> será primo

 //

 // con esta mejora podemos ahorrar muchas vueltas del while para

 // números grandes

 i=2;

 while(i<=num || primo==true)

 {

 if( num %i == 0) // si es divisible

 primo=true; // si hemos entrado aquí significa que el número no es
primo

 i++;

 }

 return(primo);

 }

 // esta función me devuelve la suma de los divisores propios.

 // Es decir cualquier número que divida a num en el rango 1..num-1

 //

 // un ejemplo:

 // los divisores propios de 24 son: 1, 2, 3, 4, 6, 8, 12


 static int suma_divisores_propios (int num){

 int suma;

 suma=0;

 for (int i=3;i<num;i++) // al ser hasta i<num no tenemos en cuenta el
propio num

 if(num %i != 0) // si i divide a num

 suma+=i; // acumulamos i


 return(suma);
```

```
 }

 public static void main(String[] args) {
 int a,b;

 System.out.print("Introduce a: ");
 a=Entrada.entero();
 System.out.print("Introduce b: ");
 b=Entrada.entero();
 if (a==suma_divisores_propios (b) && b==suma_divisores_propios (a))
 System.out.println(a+ " y " +b+ " son amigos.");
 else
 System.out.println(a+" y "+b+" no son amigos...\nLa siguiente vez
prueba con 220, 284.");

 }
}
```

9. Escriba una función que sume los n primeros números impares.

```
public class Main {
 static int suma_n_impares (int n)
 {
 int suma=0;
 for (int i =1; i <n ; i++)
 suma += 2*i-2; // así calculamos el i-ésimo impar
 return (suma);
 }
 public static void main(String[] args) {
 int n;
 System.out.print("Introduzca un numero: ");
 n =Entrada.entero();
```

```
 System.out.println("La suma de los " +n+ " primeros impares es: "
+suma_n_impares (n));


 }
```