i”

Ejercicio U3_B7_E1

```
class Unidad3 {
    static void tablaMultiplicar(int tabla, int i){
        if(i==0){
            System.out.println("Fin de tabla del "+ tabla);
        }else{
            System.out.println(tabla + "x" + i + "=" + tabla*i);
            tablaMultiplicar(tabla,i-1);
        }
    }
    public static void main(String[] args) {
      tablaMultiplicar(3,10);
    }
}
```

Ejercicio U3_B7_E2

```
class Multiplicacion{
  int multiplicar(int a, int b){

        if(b==0)
            return 0;
        else if(b>0)
              return a + multiplicar(a,b-1);
        else //b es <0
              return -a + multiplicar(a,b+1);
      }


}


class Unidad3 {
    public static void main(String[] args) {
        Multiplicacion m = new Multiplicacion();
        System.out.println(m.multiplicar(-2,-5));
        System.out.println(m.multiplicar(2,-5));
        System.out.println(m.multiplicar(-2,5));
        System.out.println(m.multiplicar(-2,0));
        System.out.println(m.multiplicar(2,0));
        System.out.println(m.multiplicar(0,-2));

    }
}
```

Ejercicio U3_B7_E3

```
import java.util.Scanner;
class Unidad3 {
    static int factorial(int n){
        int fact=1;
        while(n>0){
            fact=fact*n;
            n--;
        }
        return fact;
    }
```

```
    public static void main(String[] args) {
        Scanner teclado = new Scanner(System.in);
        System.out.println("Teclea n´´umero entero positivo para calcular factorial");
        int n =teclado.nextInt();
        System.out.println("Su factorial es: "+ factorial(n));
    }
}
```

Ejercicio U3_B7_E4

```
import java.util.Scanner;


public class Unidad3{

   public static void main(String[] args){
      Scanner teclado= new Scanner(System.in);
      System.out.println("Teclea número entero para calcular factorial:");
      int n=teclado.nextInt();
      System.out.println("Su factorial es: "+factorial(n));
   }
   static int factorial(int n){
      if(n==0)
       return 1;
      else
       return n*factorial(n-1);
   }
}
```

o con una clase factorial

```
import java.util.Scanner;
public class Unidad3{
   public static void main(String[] args){
      Scanner teclado= new Scanner(System.in);
      System.out.println("Teclea número entero para calcular factorial:");
      int n=teclado.nextInt();
      Factorial f = new Factorial();
      System.out.println("Su factorial es: "+f.calcularFactorial(n));
   }
}

class Factorial{
    int calcularFactorial(int n){
        if(n==0)//caso base
            return 1;
        else  //caso recursivo
            return n*calcularFactorial(n-1);

    }
}
```

Cuando diseñes un procedimiento recursivos jamás olvides que debe haber un caso base "donde para la recursividad".

EJERCICIO U3_B7_E5. Imprimir una triangulo de caracteres con recursividad

El orden secuencial que impone imprimir en la consola hace que sea más natural resolver este tipo de problemas con bucles que con recursión, pero de todas formas vamos a buscar una solución.

LA FRASECITA: "*imprimirTriangulo con base de n* caracteres es *imprimirTriangulo de base n -1* y a continuación imprimir una base de n caracteres"

Para simplificar suponemos siempre que imprimimos el mismo caracter, #por ejemplo

No es un ejercicio típico de recursividad pero como gráficamente puedo ver dentro de un triángulo grande uno más pequeño se presta para visualizar gráficamente el concepto de recursividad.

En busca del caso recursivo

Puedo empezar primero clavando mi mirada en el gráfico y observando que dentro de un triángulo de base 10 hay uno de base 9, dentro de uno de base 9 uno de base 8 etc.

Ahora con números concretos intento describir recursivamente la impresión de un triángulo
Por ejemplo para un triángulo de base 5:
imprimirTriangulo(5) =>imprimirTriangulo(4) + imprimirbasede5caracteres

y veo que siempre es igual para toda base
imprimirTriangulo(4) =>imprimirTriangulo(3) + imprimirbasede4caracteres

Así que ya me lanzo y enuncio un caso recursivo general para un triángulo de base n

imprimirTriangulo(n) =>imprimirTriangulo(n-1) + imprimirbasedencaracteres

En busca del caso base

Si me ciño al sentido más gráfico, el caso base más lógico es aquel que se corresponde con un triángulo de base 2 porque realmente un triángulo de base 1 no es un triángulo. Así que me queda el siguiente pseudocódigo

```
caso base:
imprimirTriangulo con base 2
caso general:
imprimirTriangulo con base n => imprimirTriangulo con base n -1 + imprimir base de n caracteres.
```

Y ahora lo escribo en java con todos los matices

```
class Unidad3{
   static void imprimirBase(int base){
   for(int i=0;i<base;i++)
            System.out.print("#");
        System.out.println();
   }

   static void imprimirTriangulo(int base){
      if (base==2){
         System.out.println("#");
         System.out.println("##");
      }else{
         imprimirTriangulo(base-1);
         imprimirBase(base);
      }

   }
   public static void main(String[] args){
      imprimirTriangulo(10);
   }
}
```

Observa que se utiliza un método extra "imprimirBase" para imprimir la base para que quede límpia y clara la definición recursiva

A partir de aquí puedo hacer mil variantes, según contexto y gusto personal. Con lo de arriba es más que suficiente pero como ejemplo puedo hacer variantes considerando que el caso base es para base==1 o incluso para base==0, aunque me aleje de mi idea gráfica inicial funcionan perfectamente, por ejemplo con base==0 el caso base lo único que tiene que hacer es exactamente nada

imprimirTriangulo(3) =>
⇓
imprimirTriangulo(2) + imprimirbasede3caracteres

⇓
imprimirTriangulo(1)+ imprimirbasede2caracteres
⇓
imprimirTriangulo(0)+imprimirbasede1Caracter
⇓
nohacernada

caso base:
imprimirTriangulo con base 0 =>no se imprime nada
caso general:
imprimirTriangulo con base n => imprimirTriangulo con base n -1 + imprimir base de n caracteres.

la idea anterior la podemos escribir en Java como

```
class Unidad3{
   static void imprimirBase(int base){
        for(int i=0;i<base;i++)
                        System.out.print("#");
        System.out.println();
   }

   static void imprimirTriangulo(int base){
      if (base>0){
         imprimirTriangulo(base-1);
         imprimirBase(base);
      }else{
               //se llega cuando base es 0 y no hago nada
               //como no hago nada puedo suprimir este else
      }

   }
   public static void main(String[] args){
      imprimirTriangulo(10);
   }
}
```

si quiero puedo hacer más explícito el caso base escribiendo el método imprimirTriangulo()

```
static void imprimirTriangulo(int base){
      if(base==0){
               //no imprimo nada pero para la recursividad
       }else{
               imprimirTriangulo(base-1);
```

```
            imprimirBase(base);
        }

    }
```

un pseudocódigo con base 1
caso base:
imprimirTriangulo con base 1 => imprime un carácter (una base de tamaño 1)
caso general:
imprimirTriangulo con base n => imprimirTriangulo con base n -1 + imprimir base de n caracteres.

```
class Unidad3{
    static void imprimirBase(int base){
    for(int i=0;i<base;i++)
                System.out.print("#");
            System.out.println();
    }

    static void imprimirTriangulo(int base){
        if (base==1){
            imprimirBase(base);
        }else{
            imprimirTriangulo(base-1);
            imprimirBase(base);
        }

    }
    public static void main(String[] args){
        imprimirTriangulo(10);
    }
}
```

Escribimos una última versión ahora haciendo también recursivo el método que escribe la base y permitiendo pintar el triángulo con un carácter indicado por parámetro

```
class Unidad3{
    static void imprimirTriangulo(char c, int base){
        if (base>0){
            imprimirTriangulo(c,base-1);
            imprimirBase(c,base);
        }

    }
    static void imprimirBase(char caracter, int cantidad){
        if(cantidad==0){
            System.out.println();
        }
        else{
            System.out.print(caracter);
            imprimirBase(caracter,cantidad-1);
```

```
        }

    }
    public static void main(String[] args){
        imprimirTriangulo('@',10);
    }
}
```

EJERCICIO U3_B7_6.

buscamos la frasecita...

dos enfoques

primer enfoque:

puedo observar que si coge el primer caracter y lo paso al final ya sólo me queda darle la vuelta al resto

por ejemplo reverse(“1234”)=> reverse(“234”) + “1”

Enuncio de forma general con con frasecita: el reverse de un String de n caracteres consiste en concatenar el reverse de los ultimos n-1 caracteres con el primer carácter

segundo enfoque:

¿y porqué no pensarlo justo al revés cogiendo el último caracter y poniéndolo de primero?

por ejemplo reverse(“1234”)=> 4+ reverse(“123”)

Enuncio de forma general con con frasecita:: el reverse de un String de n caracteres consiste en concatenar el último caracter con el reverse de los n-1 primeros caracteres

Al respecto de la parada debemos pensar si el String más pequeño que puede recibir el método es el string vacio “” o bien un String de un carácter.

solución en java

vamos a elegir las dos opciones más rayantes: el primer enfoque y la parada la hacemos en el string vacío. Para esto hay que recordar que si el beginindex de un substring tiene el mismo valor que la longitud del string entonces substring devuelve el string vacío

```
jshell> "x".substring(1)
$2 ==> ""
```

```
class Unidad3 {
    static String reverse(String s){
        if(s.length()==0){//s.isEmpty() o s.equals("")
            return "";
        }else{
           return reverse(s.substring(1))+s.charAt(0);
        }
    }

    public static void main(String[] args) {
        String invertido=reverse("acasohubobuhosaca");
        System.out.println(invertido);

        System.out.println(reverse("roman"));

    }
}
```

con bucle, también muchas formas, una:

```
 static String reverse(String s){
        String resultado="";
            for(int i=s.length()-1;i>=0;i--){
                        resultado=resultado+s.charAt(i);

            }
            return resultado;
    }
```