Ejercicio U3_B8_E1:
al calcular factoriales en seguida salen números gigantes. Con el código del enunciado A partir del 13! incluido el resultado "no cabe en un int". Además en el factorial de 34 pasa algo curioso y es que da valor 0, con lo que lógicamente todos los subsiguientes factoriales serán 0, por ejemplo, fact(35)=35*fact(34)=35*0=0 y así sucesivamente.

Para contrastar que a partir de 13! no me cabe en un int puedo usar calculadora que permita trabajar en 32 y 64 bits o más fácil, consultar una tabla de factoriales por ejemplo, http://di002.edv.uniovi.es/~cueva/JavaScript/41TablaFactorial.html

2_147_483_647 Integer.MAX_VALUE
479_001_600 12!
6_227_020_800 13!

Si usaramos long también acabará habiendo pronto problemas. Otra opción sería usar double, pero podemos generar imprecisiones. Para este problema lo mejor es BigInteger que puede ser todo lo grande que queramos a pesar de ser mucho más lento que los tipos primitivos

9_223_372_036_854_775_807 Long.MAX_VALUE (tiene 19 dígitos)
1.1240007277776077e+21 22! (tiene 21 digitos)

Una versión con BigInteger

```
import java.math.BigInteger;
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();
      BigInteger factorial=BigInteger.ONE;// o tb new BigInteger( "1 ")
      while(n>0){
          factorial=factorial.multiply(new BigInteger(Integer.toString(n)));
          n--;
      }
      System.out.println("Su factorial es: "+factorial);
   }
}
```

Ahora ejecuta este código para calcular el fact(1000) y alucina, luego ejecuta fact(10000) y alucina más

La velocidad de ejecución en este caso no es un factor especialmente crítico ya que no hay bucles anidados, por ejemplo puedes calcular el factorial de 10000 y ver que responde al instante. 10000 "vueltas" no es para tanto.

Ejercicio U3_B8_E2:

```
import java.math.BigInteger;
import java.util.Scanner;
public class Unidad3{
   static BigInteger factorial(int n){
      if(n==0){//tb. funciona con n==1 pero es más coherente con definicion n==0
         return  new BigInteger("1");
      }else{
         BigInteger bigN=new BigInteger(String.valueOf(n));
         //BigInteger bigN=new BigInteger(new Integer(n).toString());
         return  bigN.multiply(factorial(n-1));
      }
   }
   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));
   }
}
```

o usando directamente BigInteger como argumento

```
import java.math.BigInteger;
class Unidad3{
    static BigInteger fac(BigInteger n){
       if (n.equals(BigInteger.ZERO)){
          return BigInteger.ONE;
       }else{
          return n.multiply(fac(n.subtract(BigInteger.ONE)));
       }
    }
    public static void main(String[] args) {
           System.out.println(fac(new BigInteger("4")));
       System.out.println(fac(new BigInteger("100000")));//stackoverfloww
     }
}
```