Ejercicio U3_B5_1:

```
class Unidad3{
    public static void main(String[] args) {
        for(int i=0;i<101;i++)
            System.out.print(i+" ");
    }
}
```

Si el enunciado no nos indicara "ambos inclusive" ¿Qué límites pondrías?. Observa que el lenguaje natural (el nuestro) no es tan preciso como el matemático que es el que utilizamos en programación, hay en muchas situaciones por tanto en las que nosotros debemos interpretar libremente el enunciado. Si no pusiéramos ambos inclusive el "entre" se puede interpretar *for(int i=0;i<101;i++)* ó *for(int i=1;i<100;i++).* La solución de los ejercicios siempre debe ceñirse a un contexto y en nuestro caso el contexto es principalmente el enunciado, que por breve no puede ser perfecto para evitar enunciados largos que no interesan en pequeños ejercicios.

Ejercicio U3_B5_2:

```
class Unidad3{
    public static void main(String[] args) {
        for(int i=100;i>-1;i--)
            System.out.print(i+" ");
    }
}
```

Ejercicio U3_B5_3:

```
class Unidad3{
    public static void main(String[] args) {
        int suma=0;
        for(int i=100;i>0;i--)
            suma+=i;
        System.out.print(suma);
    }
}
```

Mejor en principio es sumarlos ascendentemente ya que es más fácil y más claro. Ante la duda, lo más sencillo es lo mejor. En el ejemplo anterior, el cuerpo del bucle sólo tiene una instrucción por lo que se pueden omitir las llaves de bloque (pero no se recomienda …)

```
class Unidad3{
    public static void main(String[] args) {
        int suma=0;
        for(int i=1;i<101;i++)
            suma=suma+i;
        System.out.print(suma);
    }
}
```

normalmente MÁS SENCILLO => MEJOR

Ejercicio U3_B5_4:

Confundimos suma+=i. con suma=+i. Al utilizar la segunda forma, en cada iteración del bucle le asignamos a la variable suma el valor de la variable +i (el operador unario + hace positivo un valor negativo de i, que no ocurre en este caso). Al final la variable suma se quedará con el valor de i en la última iteración.

Para localizar este tipo de errores puede ayudarnos utilizar un Debugger. Como ejemplo usa el tracer de Bluej para observar el comportamiento del bucle.

Ejercicio U3_B5_5:

Se puede resolver con if

```
class Unidad3{
   public static void main(String[] args) {
      for(int i=0;i<101;i++)
         if(i%2==0)
            System.out.print(i+" ");
   }
}
```

Recuerda de nuevo que los cuerpos de los bucles no necesitan corchetes si sólo contienen una sentencia. En el caso anterior el for sólo contiene una sentencia if y no precisa corchetes. De la misma forma, un if puede contener una o varias instrucciones, y de nuevo, si sólo contiene una instrucción no precisa corchetes como ocurre en el caso anterior. Recuerda que de todas formas, la norma de estilo recomienda siempre escribir con corchetes:

```
class Unidad3{
   public static void main(String[] args) {
      for(int i=0;i<101;i++){
         if(i%2==0){
            System.out.print(i+" ");
         }
       }
  }
}
```

Aunque observa aquí que para apenas escribir nada se dispara el número de corchetes y hay gente que prefiere escribirlo sin corchetes.

El if realmente no aporta nada en este caso, lo más fácil es incrementar de dos en dos

```
class Unidad3{
   public static void main(String[] args) {
      for(int i=0;i<101;i+=2)
         System.out.print(i+" ");
   }
}
```

Ejercicio U3_B5_6:

```
class Unidad3{
   public static void main(String[] args) {
      int impares=0;
      for(int i=0;i<101;i++)
         if(i%2!=0){
            System.out.print(i+" ");
            impares++;
         }
      System.out.println("\nTOTAL IMPARES: "+ impares);
   }
}
```

Observa que el if ya que comprende dos instrucciones, tal y como se escribió la solución, precisa corchetes. El cuerpo del FOR se escribió sin corchetes, para recordar que como su cuerpo sólo contiene una instrucción (el IF) pueden omitirse, pero mejor también el for con corchetes como sigue

```
class Unidad3{
```

```
   public static void main(String[] args) {
      int impares=0;
      for(int i=0;i<101;i++){
         if(i%2!=0){
            System.out.print(i+" ");
            impares++;
         }
      }
      System.out.println("\nTOTAL IMPARES: "+ impares);
   }
}
```

Además después de escribir un bucle en el que la eficiencia es importante hay que echar un vistazo a lo que escribimos para hacerlo más eficiente, por ejemplo, está mejor avanzar de 2 en dos:

```
class Unidad3{
   public static void main(String[] args) {

         int impares=0;
         for(int i=1;i<101;i+=2){
                  System.out.print(i+" ");
                  impares++;
         }
      System.out.print("\n números impares de 0 a 100:  "+ impares);
    }
}
```

Y está todavía mejor la siguiente solución ya que ahorramos hacer repetidamente e innecesariamente la instrucción "impares++;"

```
class Unidad3{
   public static void main(String[] args) {
      for(int i=1;i<101;i=i+2)
         System.out.print(i+" ");
      System.out.println("\nTOTAL IMPARES: "+ 100/2);
   }
}
```

Ejercicio U3_B5_7:

```
import java.util.Scanner;
class Unidad3{
   public static void main(String[] args) {
      Scanner teclado= new Scanner(System.in);
      System.out.println("introduce un número natural por teclado: ");
      int numero=teclado.nextInt();
      for(int i=1;i<numero;i++)
            System.out.print(i+" ");
   }
}
```

Aquí el "hasta" del enunciado se interpretó de forma que dicho número no se incluye en el listado. El enunciado debería ser más preciso si quisiéramos asegurar que sólo esta interpretación es la correcta.

Ejercicio U3_B5_8: Imprime una tabla que muestra los números de 0 a 127 en decimal con su equivalente binario, octal y hexadecimal. Consulta en el API la clase Integer y localiza los métodos que permiten convertir un entero en un String binarios, octal y hexadecimal

```
import java.util.Scanner;
class Unidad3{
    public static void main(String args[]){
        System.out.println("int\tbinario\toctal\thexadecimal");
        for(int i=0;i<128;i++){
            System.out.println(i+"\t"+Integer.toBinaryString(i)+"\t"+Integer.toOctalString(i)+"\t"+Integer.toHexString(i));
        }
    }
}
```

Ejercicio U3_B5_9:

```
class Unidad3{
    public static void main(String[] args) {
        //con for
        for(int i=50;i>-1;i-=2)
            System.out.print(i+" ");

        //con while
        System.out.println();
        int j=50;
        while(j>-1){
            System.out.print(j+" ");
            j-=2;
        }

        //con dowhile
        System.out.println();
        int k=50;
        do{
            System.out.print(k+" ");
            k-=2;
        }while(k>-1);
    }
}
```

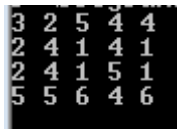
Ejercicio U3_B5_10:

Los informáticos estamos obsesionados con las potencias de 2 debido a su relación con el sistema binario. Imprimimos lo que pide el enunciado de cuatro maneras ¿Se te ocurre alguna solución adicional?

```
class Unidad3{
    public static void main(String[] args) {
        for(int i=1;i<=1024;i+=i)
            System.out.print(i+"    ");

        System.out.println();
        for(int i=1;i<=1024;i*=2)
            System.out.print(i+"    ");

        System.out.println();
        for(int i=0;i<11;i++)
            System.out.print((int)Math.pow(2, i)+ "    ");

        System.out.println();
        for(int i=1;i<=1024;i=i<<1)
```

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

}
```

Ejercicio U3_B5_11: Haz 20 tiradas de un dado e imprime su resultado en 4 filas de 5 resultados cada una como en el ejemplo.



cada ejecución evidentemente tendrá valores diferentes

para saber cuando tengo que cambiar de línea puedo utilizar una variable extra para ir contando hasta 5 o bien utilizar el truco de que cada vez que alcanzo un múltiplo de 5 es porque pasé por 5 tiradas.

System.out.println();

Si cuento desde 0 metemos al principio una línea en blanco. Si molesta habría que filtrarla

```
import java.util.Random;
class Unidad3{
    public static void main(String args[]){
        Random dado= new Random();
        for(int i=0;i<20;i++){
            int tirada=dado.nextInt(6)+1;
            if(i%5==0){
                System.out.println("");
            }
            System.out.print(tirada+"  ");


        }
    }
}
```

Como comentamos también se podría haber hecho con una variable que cuente de 5 en 5(entre otras muchas posibilidades)

```
import java.util.Random;
class Unidad3{
  public static void main(String args[]){
        Random dado = new Random();
            int tirada;
            int cuentaHasta5=1;
            for(int i=0;i<20;i++){
                        tirada=dado.nextInt(6)+1;
                        System.out.print(tirada+" ");
                        cuentaHasta5++;
                        if(cuentaHasta5>5){
                                    System.out.println();
                                    cuentaHasta5=1;
                        }
            }
  }
}
```

Ejercicio U3_B5_12: Simulamos tirar un dado usando la clase Random. Tiramos el

dado 6000000 de veces. Vamos almacenando la cantidad de veces que sale cada cara. Si realmente se generan números aleatorios los valores serán aceptablemente próximos.

```
import java.util.Random;
class Unidad3{
   public static void main(String args[]){
      Random dado = new Random();
      int cara1=0,cara2=0,cara3=0,cara4=0,cara5=0,cara6=0;
      for(int i=0;i<6000000;i++){
         switch(dado.nextInt(6)+1){
            case 1: cara1++;
                  break;
            case 2: cara2++;
                  break;
            case 3: cara3++;
                  break;
            case 4: cara4++;
                  break;
            case 5: cara5++;
                  break;
            case 6: cara6++;
         }
      }
      System.out.println("Cara 1: "+cara1);
      System.out.println("Cara 2: "+cara2);
      System.out.println("Cara 3: "+cara3);
      System.out.println("Cara 4: "+cara4);
      System.out.println("Cara 5: "+cara5);
      System.out.println("Cara 6: "+cara6);
   }
}
```

los println() los hice evidentemente con copiar y pegar. Recuerda que el uso de copiar/pegar dentro de un programa delata un error de diseño. En este caso "El error" es que en lugar de haber utilizado 6 variables "cara" debí haber usado un array de int (o similar). ¿array?. Es un concepto aún no visto y por tanto la solución anterior con sus problemas, la damos por válida. Simplemente quería hacerte recapacitar sobre "copiar/pegar => error de diseño" que ocurre en una diversidad de situaciones si no se aplican las técnicas de programación correctas. Recuerda también que nosotros le llamamos informalmente el problema de "copiar/pegar" pero el nombre apropiado del problema es "generar código duplicado".

Ejercicio U3_B5_13:

```
import java.util.Scanner;
import java.util.Random;
class Unidad3{
        public static void main(String[] args){
                Scanner teclado= new Scanner(System.in);
                String stringEleccion;
                int eleccion;
                Random moneda= new Random();
                int jugada;
                String seguirJugando;
                do{
                   System.out.println("¿Cara o Cruz?");
                   stringEleccion= teclado.next();
                   if(stringEleccion.equals("Cara")){
                      eleccion=0;
                    }else{
                       eleccion=1;
                    }
                        jugada=moneda.nextInt(2);
                        if(jugada==eleccion){
                           System.out.println("ACIERTO");
                        }else{
```

```
                        System.out.println("FRACASO");
                    }
                System.out.print("¿Quieres seguir jugando?(sí/no)?: ");
                seguirJugando=teclado.next();
            }while(seguirJugando.equals("sí"));
            System.out.print("CHAO");

        }
}
```

¡pero había que hacerlo con for!

```
import java.util.Scanner;
import java.util.Random;
class Unidad3{
        public static void main(String[] args){
                Scanner teclado= new Scanner(System.in);
                String stringEleccion;
                int eleccion;
                Random moneda= new Random();
                int jugada;
                String otraTirada="si";
                for(;otraTirada.equals("si");){
                        System.out.println("¿Cara o Cruz?");
                        stringEleccion= teclado.next();

                        eleccion=stringEleccion.equals("Cara")?0:1;

                        jugada=moneda.nextInt(2);


                        System.out.println(jugada==eleccion?"ACIERTO":"FALSO");
                        System.out.println("¿otra tirada(si/no)?");
                        otraTirada=teclado.next();

                }

        }
}
```

Ejercicio U3_B5_14:

```
class Unidad3 {
   public static void main(String[] args) {
       for(int i=1;i<4;i++)
          for(int j=1;j<4;j++)
             System.out.println(i+""+j);
   }
}
```

¿por qué concatenamos con la cadena vacía ""?
Recuerda: el operador "+" suma números pero también concatena strings. Si el operador + tiene un operando String entonces su operación va a ser concatenar, es decir, prevalece la concatenación a la suma aritmética de forma que los operandos aritméticos los convierte automáticamente a Strings para hacer la concatenación. . Por esta razón, en el println() anterior incluimos un string vacío para que el + interprete todos los operandos como un String y no haga sumas aritméticas. Conclusión, lo hicimos por pura comodidad, para evitar tener que convertir nosotros expresamente números a String.

Ejercicio U3_B5_15:

```
class Unidad3 {

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

```
        for (int i = 1; i < 10; i++) {
            System.out.println("Tabla de multiplicar del " + i);
            System.out.println("---------------------------- ");
            for (int j = 1; j < 10; j++) {
                System.out.println("\t" + i + "x" + j + "=" + (i * j));
            }
            System.out.println();
        }

    }
}
```

Ejercicio U3_B5_16:

```
class Unidad2{
    public static void main(String[] args) throws Exception{
        for(char c1='A';c1<'D'; c1++){
            for(char c2='A';c2<'D'; c2++){
                    for(char c3='A';c3<'D'; c3++){
                                    System.out.println(""+c1+c2+c3);
                    }
            }
        }
    }
}
```

Ejercicio U3_B5_17: Hay cientos de formas de escribir esta idea, la siguiente es una que se pega bastante a la descripción del enunciado.

```
import java.util.Scanner;
class Unidad3{
  public static void main(String args[]){
      Scanner teclado = new Scanner(System.in);
      System.out.print("número entero para pasar a binario: ");
      String resultado="";
      for(int numero=teclado.nextInt();numero>0;numero=numero/2){
          resultado=numero%2+resultado;
      }
      //si por teclado introducimos el valor 0 no se entra en bucle
      if(resultado.equals(""))//caso especial pasar 0 decimal a 0 binario
          System.out.println("0");
      else
          System.out.println(resultado);
  }
}
```

Ejercicio U3_B5_18:

Este tipo de ejercicios tienen muchas soluciones ya que las relaciones que se pueden establecer entre los contadores de los bucles son intrincadas, aquí simplemente vemos alguna.

Una estrategia posible, empezando a numerar las filas desde cero:
Me imagino un cuadrado de 10(filas)x10(Columnas). Y observo que

En la fila 0(i=0) => imprimo 1 *
En la fila 1 (i=1)=> imprimo 2 *

……..
En la fila 9 (i=9)=> imprimo 10 *

ES DECIR, la regla general es que EN LA FILA i imprimo i+1 asteriscos por eso en el bucle interior imprimo desde j =0 hasta j<i+1 es decir i+1 asteriscos. Para este problema a lo mejor podría ser más natural contar desde i=1 en lugar de i=0 pero luego sin en la web cotejo soluciones vemos que todo el mundo empieza con i=0 y nos liamos completamente.

```
class Unidad3{
   public static void main(String[] args) {
      for(int i=0;i<10;i++){
         for(int j=0;j<i+1;j++){
            System.out.print("*");
         }
         System.out.println();
      }
   }
}
```

Ejercicio U3_B5_19:

Ahora en cada fila voy hacer dos cosas: primero imprimo espacios y justo a continuación asteriscos.
Por lo tanto un razonamiento es:
PARA i=0; i<10;i++
    Imprimir espacios
    Imprimir asteriscos

En la fila 1(i=0) => imprimo 9 espacios y 1 *
- 9 espacios los intento expresar en función de i y resulta que es 10-(i+1)
- 1 asterisco si los expreso en función de i es (1+1)

A continuación compruebo que estas afirmaciones son verdaderas para otros valores de i y entonces podré trabajar con las expresiones en función de i dentro del bucle
En la fila 2 (i=1)=> imprimo 8 (10-(i+1)) espacios y 2(i+1) *
……..
En la fila 10 (i=9)=> imprimo 0 (10-(i+1)) espacios y 10(i+1) *

Es decir:
PARA i=0; i<10;i++
    Imprimir 10 -(i+1) espacios
    Imprimir i+1 asteriscos

Recuerdo que el ejercicio anterior ya imprimía i+1 asteriscos, así que lo único que me falta añadir es la impresión de espacios

Nota: ojo con los paréntesis en 10 -(i+1) QUE NO ES IGUAL A 10- i +1

```
class Unidad3 {
```

```
    public static void main(String[] args) {
        for (int i = 0; i < 10; i++) {
            //falta imprimir 10 - i+1 espacios
            for (int j = 0; j < i + 1; j++) {
                System.out.print("*");
            }
            System.out.println();
        }
    }
}


class Unidad3{
    public static void main(String[] args) {

            for(int i=0;i<10;i++){
                    for(int j=0;j<(10-(i+1));j++){
                            System.out.print(" ");
                    }
                    for(int j=0;j<i+1;j++){
                            System.out.print("*");
                    }
                    System.out.println();

            }
    }
```

Otras soluciones ¡hay muchas!

En la solución anterior hay "una lógica", pero no es la única, por ejemplo, otro podría querer razona "imprimo en cada fila 10 caracteres y para cada carácter decido si es blanco o asterisco"r:

```
class Unidad3{
    public static void main(String[] args) {
        for(int i=0;i<10;i++){
            for(int j=0;j<10;j++){
                    if(j<10-(i+1)){
                            System.out.print(" ");
                    }else{
                            System.out.print("*");
                    }

            }
                System.out.println();
        }
    }
}
```

Otro razonamiento es imprimir observando que si una fila tiene 10 caracteres tengo que imprimir (10-(i+1)) espacios y a continuación imprimir * desde 10-(i+1)hasta 10.

```
class Unidad3{
    public static void main(String[] args) {
        for(int i=0;i<10;i++){
            for(int j=0;j<(10-(i+1));j++){
                System.out.print(" ");
            }
            for(int j=(10-(i+1));j<10;j++){
                System.out.print("*");
            }
            System.out.println();
        }
    }
}
```

conclusión: para alcanzar una solución antes de escribir el código java tengo que razonar el problema y "ver una solución" sin pensar en java. por ejemplo el razonamiento que hicimos fila a fila para obtener la relación que siempre se cumple entre i y los espacios y asteríscos.

U3_B5_20:
Este vuelve a ser fácil porque no necesita imprimir espacios al principio

```
public class Unidad3 {
    public static void main(String[] args) {
        for (int i = 0; i<10; i++) {
            for(int j=0;j<10-i;j++){
                System.out.print("*");
            }
            System.out.println("");
        }
    }
}
```