Detalles de la tarea de esta unidad.

Enunciado.
Para afianzar los conocimientos en el uso de estructuras de control de flujo así como de excepciones, vamos a realizar una serie de programas. **Se deberá crear un proyecto en Netbeans diferente para cada uno de los ejercicios propuestos, cuyo nombre será PROG04_EjercX, donde X es el número de ejercicio**. Cada proyecto incluirá una sola clase que contendrá un método `main`. El alumno decidirá si incluir todo el código en el método `main` o dividirlo en diferentes métodos.

- **Ejercicio 1:** Implementar un programa que muestre la tabla de multiplicar de un número leido desde teclado utilizando al menos tres bucles diferentes. El número leído desde teclado debe ser menor que 30. En caso contrario se mostrará un mensaje por pantalla y el programa finalizará.
- **Ejercicio 2:** Un número es primo si solo tiene dos divisores: el 1 y el propio número. Implementa un programa Java que pida por teclado 5 números. Para cada uno de ellos:
  - Comprueba si es negativo. En caso afirmativo, muestra el mensaje por pantalla "El número es negativo".
  - Si es positivo, deberá mostrar por pantalla si es primo o no.
  - Procesados los 5 números, el programa finaliza.
- **Ejercicio 3:** El Mínimo Común Múltiplo (MCM) de un conjunto de dos números es el número positivo más pequeño que es múltiplo de los dos números. Es posible calcular el MCM de tres o más números. Por ejemplo, el MCM (2,3) es 6. El 6 es el múltiplo mas pequeño de 2 y de 3. Implementa un programa Java que pida dos números por teclado, compruebe que son positivos y calcule su MCM. En caso de no ser ambos números positivos, el programa mostrará un mensaje por pantalla y finalizará.
- **Ejercicio 4:** Deseamos implementar un juego en Java que permita al usuario adivinar un número oculto (que será aleatorio). El funcionamiento será el siguiente:
  - El programa mostrará un pequeño menú en pantalla con las siguientes opciones (**1. Configurar, 2. Jugar, 3. Salir**).
    - Si el usuario selecciona el la primera opción, se le solicitará por teclado el número de intentos permitidos (`numInt`) y el número máximo (`numMax`) generado.
    - Si el usuario selecciona la opción 2, el programa generará un número aleatorio entre 0 y `numMax` que será el número a adivinar (`numOculto`). A partir de este momento, se le solicitarán al usuario números hasta adivinar el número oculto.
      - Si el usuario adivina el número, se mostrará un mensaje "*Has ganado!. Has necesitado X intentos"*.
      - Si se consume el número de intentos sin adivinar el número, se mostrará el mensaje "*Perdiste!. Intentos consumidos*".
      - En cada intento, si el usuario no adivina el número se le proporcionará una pista, por ejemplo, "*El número oculto es menor*".
      - En ambos casos, la siguiente acción será mostrar el menú.
      - Si el usuario selecciona Jugar sin configurar previamente el número de intentos y el número máximo generado, se tomarán como valores por defecto: `numInt=5` y `numMax=10`.
    - Si el usuario pulsa la opción 3, el programa finaliza.
  - Para generar un número aleatorio en java puedes utilizar el siguiente código:

```
int numOculto = (int)Math.floor(Math.random()*20+1); //genera un número aleatorio entre 0 y 20, ambos incluidos.
```

- **Ejercicio 5:** Cuando dividimos un número entre 0 se genera un valor indeterminado. En cualquier lenguaje de programación este tipo de operaciones genera un error de ejecución que debe ser controlado desde el código para evitar malas experiencias al usuario. En Java, cuando se produce esta operación se genera la excepción `ArithmeticException`. Queremos implementar un programa Java que calcule la división de dos números solicitados por teclado (dividendo y divisor). El programa solicitará números indefinidamente hasta que los dos números solicitados sean -1. Se debe **controlar mediante excepciones** que el divisor no sea 0. En caso de serlo, se mostrará un mensaje por pantalla. También habrá que mostrar por pantalla el número de divisiones calculadas. Utiliza número enteros para las variables.