Tema 6 – S3

Contenidos:

2. La iteración como estrategia de diseño: fases y elementos
   - Estrategias para descubrir la estructura iterativa del problema
   - Mecánica y Semántica del bucle `while`
   - Ejemplos y algunos ejercicios no resueltos en la Sesión 1
   - Resolución de Recurrencias usando un bucle `while`: término, serie y error; la función exponencial
3. Usando un bucle do-while para validación de datos

La iteración como estrategia de diseño
Cuestiones propuestas sobre el Ejemplo 1

1. Dado que 1 * b = b, independientemente del valor de b, se podría haber pensado en la siguiente estrategia iterativa: **Repetir a – 1 veces res = res + b**. Diseña el bucle `while` **correcto** que corresponde a esta estrategia

   En ***sesion1*** de ***ejercicios – Tema 6***

2. Los siguientes bucles se podrían haber propuesto como alternativas al que se ha diseñado. **El primero es correcto y el segundo NO termina** (bucle infinito). Razona por qué; además, en el caso del primero, indica qué representa el contador `i`

```
int res = 0, i = a;
while (i != 0) {
    res = res + b;
    i--;
}
```

Estrategia: `i` es el nº de sumas a realizar TRAS cada iteración

```
int res = 0, i = 0;
while (i >= 0) {
    res = res + b;
    i++;
}
```

Bucle **INFINITO:** al inicializar `i` a 0 e incrementarlo en 1 en cada iteración, `i >= 0` se cumple **SIEMPRE**

Resolución de Recurrencias usando un bucle while
Término, serie y **error: la función exponencial**

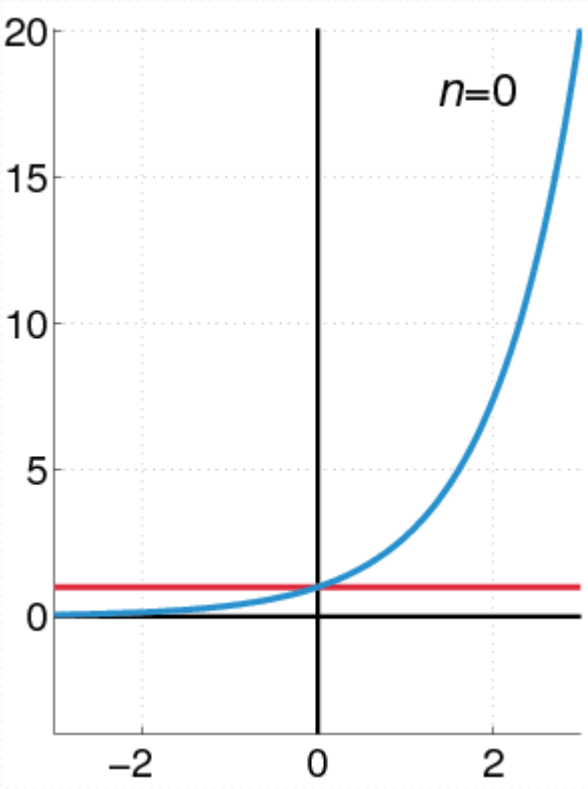
- La función exponencial e^x puede ser definida como una serie de potencias (Desarrollo en Serie de Taylor - AMA):

$$e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!} = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \cdots$$

Por tanto, si x = 1:

$$e = \sum_{n=0}^{\infty} \frac{1}{n!} = \frac{1}{0!} + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \cdots$$

$e \approx 2{,}718281828459045235\ldots$



- Completa el bucle del siguiente método para que: **(a)** devuelva el término n-ésimo de la serie; **(b)** devuelva el valor n-ésimo de la serie; **(c)** obtenga una **aproximación** al valor de e^x

En ***sesiones2Y3*** de ***ejercicios – Tema 6*** (programa EjeElevadoAx)

Resolución de Recurrencias usando un bucle while

Término, serie y error: **la función exponencial (a)**

Dado el nº Real $x \geq 0$,
obtener **el término** $n(\geq 0)$ de la serie $\sum_{n=0}^{\infty} \frac{x^n}{n!} = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \cdots$

```
// PRECONDICIÓN: x > = 0 AND n >= 0
public static double calcularTermino(int n, double x) {
    int i = [ ]; double res = [ ];
    while  ([ ]) {
           [ ]
    }

    return res;
}
```

Resolución de Recurrencias usando un bucle while
Término, **serie** y error: **la función exponencial (b) (I)**

Dado el nº real $x \geq 0$,
obtener **el valor** $n(\geq 0)$ de la serie $\sum_{n=0}^{\infty} \frac{x^n}{n!} = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \cdots$

```
// PRECONDICIÓN: x > = 0 AND n >= 0
public static double calcularSerie(int n, double x) {
    int i = 0 ; double termino = 1 ;  // calculado término a0
                                      // y sumados hasta el 0
    double res = 1 ;
    while  ( i != n ) { // i < n

        i++;
        termino = termino * x / i;
        res = res + termino;
    }
    // PARADA : i == n
    return res;
}
```

// calculado término a_i
// y sumados hasta el i

// calculado término a_n
// y sumados hasta el n

Resolución de Recurrencias usando un bucle while
Término, **serie** y error: **la función exponencial (b) (II)**

Dado el nº real $x \geq 0$,
obtener **el valor** $n(\geq 0)$ de la serie $\sum_{n=0}^{\infty} \frac{x^n}{n!} = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \cdots$

```
// Ejemplo de ejecución del programa EjercicioeElevadoAx
Introduce un numero real (>=0): 3
Introduce #terminos a sumar (>=0): 10
Termino  1 de la serie = 3,0000000000000000
Termino  2 de la serie = 4,5000000000000000
Termino  3 de la serie = 4,5000000000000000
Termino  4 de la serie = 3,3750000000000000
Termino  5 de la serie = 2,0250000000000000
Termino  6 de la serie = 1,0125000000000000
Termino  7 de la serie = 0,4339285714285714
Termino  8 de la serie = 0,1627232142857143
Termino  9 de la serie = 0,0542410714285714
Termino 10 de la serie = 0,0162723214285714
Valor aproximado de e^3.0 = 20.07966517857142 50
Valor de Math.exp(3.0) = 20.08553692318766 80
```

¿Es una buena aproximación?

¿Y sumando más de 10 términos?

Resolución de Recurrencias usando un bucle while
Término, serie y **error**: **la función exponencial (c) (I)**

Dado el nº real $x \geq 0$,
obtener **el valor** $n(\geq 0)$ de la serie $\sum_{n=0}^{\infty} \frac{x^n}{n!} = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \cdots$

```
// Ejemplo de ejecución del programa EjercicioeElevadoAx
Introduce un numero real (>=0): 3
Introduce #terminos a sumar (>=0)?: 20
Termino  1 de la serie = 3,0000000000000000
Termino  2 de la serie = 4,5000000000000000
...
Termino 10 de la serie = 0,0162723214285714
Termino 11 de la serie = 0,00443790584415 58
Termino 12 de la serie = 0,00110947646103 90
Termino 13 de la serie = 0,00025603302947 05
Termino 14 de la serie = 0,00005486422060 08
Termino 15 de la serie = 0,00001097284412 02
Termino 16 de la serie = 0,00000205740827 25
Termino 17 de la serie = 0,00000036307204 81
Termino 18 de la serie = 0,00000006051200 80
Termino 19 de la serie = 0,00000000955452 76
Termino 20 de la serie = 0,00000000143317 91
Valor aproximado de e^3.0 = 20.0855369229508440
Valor de Math.exp(3.0) = 20.0855369231876680
```

$e^x \approx a_0 + a_1 + \ldots + a_{n=20}$

$e^x = a_0 + a_1 + \ldots + a_{n=20} + \ldots + a_\infty$

R_n
Error al aproximar

¿Ha mejorado la aproximación?

¿Por qué?

Resolución de Recurrencias usando un bucle while
Término, serie y **error: la función exponencial (c) (II)**

Dado el nº real $x \geq 0$, obtener el valor de la función $e^x \approx \sum_{n=0}^{\infty} \frac{x^n}{n!}$ **con el menor error posible, ¿epsilon?**

$$0.0 \leq \text{epsilon} \leq R_n$$

```
// PRECONDICIÓN: x > = 0 AND n >= 0
public static double calcularSerie(int n, double x, double epsilon) {

    int i = 0; double termino = 1; double res = 1;

    while  ( termino >= epsilon ) {
        i++;
        termino = termino * x / i;
        res = res + termino;
    }
    // PARADA: termino < epsilon
    return res;
}
```

Resolución de Recurrencias usando un bucle while
Término, serie y **error**: **la función exponencial (c) (III)**

Dado el nº real x ≥ 0, obtener el valor de la función $e^x \approx \sum_{n=0}^{\infty} \frac{x^n}{n!}$

con el menor error posible, 0.0 ≤ **epsilon** ≤ 1.0

```
// Ejemplo de ejecución del programa EjercicioeElevadoAx
Introduce un numero real (>=0): 3
Introduce el error permitido: 1e-11
Termino  1 de la serie = 3,0000000000000000
Termino  2 de la serie = 4,5000000000000000
...
Termino 10 de la serie = 0,0162723214285714
...
Termino 20 de la serie = 0,0000000014331791
Termino 21 de la serie = 0,0000000002047399
Termino 22 de la serie = 0,0000000000279191
Termino 23 de la serie = 0,0000000000036416
Valor aproximado de e^3.0 = 20,085536923187 1460
Valor de Math.exp(3.0) = 20,085536923187 6680

          epsilon = 1e-11 = 0,00000000001
```

Usando un bucle do-while para la validación de datos (I)

- Uno de los principales usos de un bucle do-while es **filtrar/validar datos leídos desde fichero** (teclado) y, así, **garantizar las Precondiciones de los métodos** o, equivalentemente, **su CORRECTO funcionamiento**

  Por ejemplo, en el programa EjelevadoAx del package ***sesiones2Y3*** del proyecto BlueJ *ejercicios – Tema 6* tienes el código de lectura y validación del error permitido (epsilon) al calcular e^x, invocando al método calcularSerie(double epsilon, double x):

```
double epsilon;
do {
   System.out.print("Introduce el error permitido: ");
   epsilon = teclado.nextDouble();
} while (epsilon < 0.0 || epsilon > 1.0);
```

  Fíjate bien en lo que expresa: mientras que el valor leído de epsilon **NO** sea válido, es decir, mientras que **NO** pertenezca al intervalo [0.0, 1.0], **REPETIR** al usuario la petición del error permitido -lo que incluye también el System.out.print. Nota también que sólo así se garantiza la Precondición de calcularSerie

- **¿**Por qué usar en estos casos un do-while y no un while? **Porque se ejecuta al menos 1 vez**, lo que es imprescindible para validar datos leídos de fichero (teclado): ¿cómo validar un dato que NO se ha leído aún? ¡¡Igual ya es válido a la primera!!

Usando un bucle do-while para la validación de datos (II)

- Otro uso típico del bucle **do-while** es filtrar/validar la **opcion** del menú del **main** de un programa, un valor entero leído desde teclado y perteneciente a un intervalo dado:

```
public static int menu(Scanner teclado) {
    int opcion;
    do {
       System.out.println(" MENU");
       System.out.println("1. Area de un circulo");
       System.out.println("2. Perimetro de un circulo");
       System.out.println("3. Mostrar un circulo en pantalla");
       System.out.println("0. Acabar");
       System.out.print("\nElige una opcion: ");
       opcion = teclado.nextInt(); teclado.nextLine();
    } while (opcion < 0 || opcion > 3);
    return opcion;
}
```

- **¿Por qué el do-while de lectura/filtrado de opcion aparece "encapsulado" en el método menu?**
  Pues por los mismos motivos, hemos definido la clase **ValidarDatos** en una librería Java

Abre el package ***miUtil*** del proyecto y, dentro de él, la clase **ValidarDatos**. Podrás ver distintos ejemplos de uso del bucle **do-while**; luego observa cómo el resto de clases del proyecto usan los métodos de esta clase.