Introducción a la computación numérica
Propósito. Resolver problemas numéricos (con mucho cálculo de reales) con **eficiencia** (en poco tiempo) y **precisión** (con poco error)

Problemas numéricos:
1.Evaluar polinomios
2.Evaluar series
3.Calcular raíces de funciones continuas
4.Calcular área bajo la curva (integral definida)

"Errores" en números reales

Si A y B son reales, A=a±a' y B=b±b' (a´,b´: errores)

A + B = (a+b) ± (a'+b')

A * B = (a*b) ± (a*b'+a´*b+a'*b')

Nota. Tiempo de * y / ≈ 10 a 100 veces tiempo de +,-,y comparar

Problema. Evaluar un polinomio

Ejemplo 1: evaluar $3x^0+2x^1-6x^3+4x^5$ en x=0.1, 0.2, ... , 1.0

```
double[]a={3,2,0,-6,0,5};
Polinomio p=new Polinomio(a);//ctor c/coef
for(double x=0.1; x<=1.0; x+=0.1)
   U.println(p.valor(x));
```

Ejemplo 2: evaluar $3.5x^{23} - 4x^7$ para x=0.5

```
Polinomio p=new Polinomio(23);//ctor c/grado
p.coeficiente(23,3.5);//agregar coeficiente
p.coeficiente(7,-4);
U.println(p.valor(0.5));
```

```
class Polinomio{
protected double[] a; //coeficientes
protected int n;      //grado
public Polinomio(double[]x){
   n = x.length-1;
   a = new double[n+1];
   for(int i=0;i<=n;++i) a[i]=x[i];
}
public Polinomio(int x){ //ctor para grado
   a=new double[(n=x)+1];//ceros
}
public void coeficiente(int i,double x){
   a[i]=x;
}
public double valor(double x){
   ...
}
}
```

Solución 1: $\sum a_i x^i$

```
public double valor(double x,int n){
   double suma=0;
   for(int i=0; i<=n; ++i)
      suma += a[i] * Math.pow(x,i);
   return suma;
}
```

en cada iteración
- una suma de reales
- control del ciclo es eficiente y exacto(++i , i<=n no son críticos)
- una multiplicación de reales (un orden más lento que una suma)
- evaluación de función pow
  - pow(x,i) es equivalente a exp(i*log(x))
  - exp(x): $1 + x + x^2/2! + x^3/3! + \ldots$
  - log(x): $2(z + z^3/3 + z^5/5 + \ldots)$ con z =(x-1)/(x+1)

Solución 2: con función potencia propia

```
public double valor(double x,int n){
   double suma=0;
   for(int i=0; i<=n; ++i)
      suma += a[i]*potencia(x,i);//1*,1+,potencia
   return suma;
}
```

potencia con i multiplicaciones

```
static public double potencia(double x,int i){
   return i==0 ? 1.0 : x * potencia(x,i-1);
}
```

potencia con aprox $\log_2 i$ multiplicaciones

```
static public double potencia(double x,int i){
   if(i==0) return 1.0;
   double aux = potencia(x,i/2);
   return i%2==0 ? aux*aux : x*aux*aux;
}
```

Solución 3. Calcular potencias sucesivas

```
public double valor(double x,int n){
   double suma=0, potencia=1;
   for(int i=0; i<=n; ++i) {
      suma += a[i] * potencia; // 1+ y 1*
      potencia *= x;           // 1*
   }
   return suma;
}
```

Solución 4. Factorización de Horner:

$(\ldots((a_n * x + a_{n-1})x + a_{n-2})x * \ldots + a_1)x + a_0$

```
public double valor(double x,int n){
   double suma=0;
   for(int i=n; i>=0; --i)
      suma = suma * x + a[i];  // 1* y 1+
   return suma;
}
```

Problema. Evaluar e^x mediante la serie $1 + x + x^2/2! + x^3/3! + \ldots$
Solución 1.

```
static public double exp(double x,int n){
   double suma=0;
   for(int i=0; i<n; ++i)
     suma += potencia(x,i) / factorial(i);
   return suma;
}
```

en cada iteración
- una suma
- una división
- $\log_2$ i multiplicaciones en potencia
- i multiplicaciones en factorial

```
static public long factorial(int x){
   if(x==0)
     return 1;
   else
     return x * factorial(x-1);
}
```

abreviando:

```
static public long factorial(int x){
   return x==0? 1 : x*factorial(x-1);
}
```

tipo long
- enteros de 64 bits
- mayor: 2^{63}-1 = 9.223.372.036.854.775.807
- hasta 19 dígitos
- hasta factorial de 20
- ¿x>20? tipo double, pero máximo 15 dígitos significativos

Solución 2. Acumulando potencia y factorial

```
static public double exp(double x,int n)
{
   double suma=1, potencia=1, factorial=1;
   for(int i=1; i<n; ++i){
     potencia *= x;
     factorial *= i;
     suma += potencia/factorial;
   return suma;
}
```

en cada iteración
- una suma
- dos multiplicaciones
- una división

Solución 3. nuevo término=f(término anterior)

```
double exp(double x,int n){
   double suma=1, termino=1;
   for(int i=1; i<n; ++i){
     termino *= x/i;
     suma += termino;
   }
   return suma;
}
```

n	exp(1,n)	n	exp(1,n)
1	1.0	10	2.7182815255731922
2	2.0	11	2.7182818011463845
3	2.5	12	2.718281826198493
4	2.6666666666666665	13	2.7182818282861687
5	2.708333333333333	14	2.7182818284467594
6	2.7166666666666663	15	2.71828182845823
7	2.7180555555555554	16	2.718281828458995
8	2.7182539682539684	17	2.718281828459043
9	2.71827876984127	**18**	**2.7182818284590455 exp(1)**

Solución 4.

```
double exp(double x,double epsilon)
{
   double suma=1, termino=1;
   for(int i=1; true; ++i){
     double s = suma;  //suma anterior
     suma += termino *= x/i;
     if(suma-s <= epsilon) break;
   }
   return suma;
}
```

- epsilon: precisión deseada (ej: 0.001)
- iteraciones terminan cuando se alcance la precisión (ej: resultado con 3 decimales)
- en cada iteración se agrega una resta y una comparación de reales

```
double exp(double x,double eps){
   double suma=1, termino=1;
   for(int i=1; termino>eps; ++i)
     suma += termino *= x/i;
   return suma;
}
```

eps	exp(1,eps)
0.1	2.708333333333333
0.01	2.716666666666663
1E-3	2.718253968253968 4
1E-4	2.7182787698412 7
1E-5	2.7182815255731922
1E-6	2.7182818011463845
1E-7	2.718281826198493
1E-8	2.7182818282861687
1E-9	2.7182818284467594
1E-10	2.718281828458 5823
1E-11	2.718281828458 8995
1E-12	2.718281828458 995
1E-13	2.718281828459 043
1E-14	2.718281828459 0455 **Math.exp(1)**

Ejercicio clase 19:

Calcular seno de un ángulo en radianes con precisión epsilon mediante la serie $x - x^3/3! + x^5/5! - \ldots$

```
static public double seno(double x,double epsilon){
```

Solución 1.

```
double seno(double x,double epsilon)
{
   double suma=0;
   int signo=-1;
   for(int i=1; true; ++i){
     double s = suma;  //suma anterior
     int n=2*i-1; signo=-signo; //entre int
     suma += signo * Math.pow(x,n)/factorial(n);
     if(Math.abs(suma-s) <= epsilon) break;
   }
   return suma;
}
```

en cada iteración, las sgtes operaciones entre reales

- `Math.pow y factorial`
- `1 +, 1 -, 1 *, 1 / y 1 comparación`

Solución 2.

```
double seno(double x,double epsilon)
{
   double suma=0;
   int signo=-1;
   for(int i=1; true; i+=2){
     double s = suma;  //suma anterior
     signo = -signo;
     suma += signo * Math.pow(x,i)/factorial(i);
     if(Math.abs(suma-s) <= epsilon) break;
   }
   return suma;
}
```

Solución 3. Acumulando potencia y factorial

```
double seno(double x,double epsilon)
{
   double suma=x,potencia=x,factorial=1;
   int signo=1;
   for(int i=3; true; i+=2){
     potencia *= x*x;
     factorial *= i*(i-1);
     signo = -signo;
     double s = suma;  //suma anterior
     suma += signo*potencia/factorial;
     if(Math.abs(suma-s) <= epsilon) break;
   }
   return suma;
}
```

Solución 4. calculando término del anterior

```
double seno(double x,double epsilon)
{
double suma=x, termino=x;
for(int i=3; true; i+=2){
  termino *= -(x*x)/((i-1)*i);
  if(Math.abs(termino) <= epsilon) break;
   suma += termino;
}
return suma;
}
```

Solución 5. abreviando

```
double seno(double x,double epsilon)
{
double suma=x, termino=x;
for(int i=3;
    Math.abs(termino*=-(x*x)/((i-1)*i))>epsilon;
    i+=2)
  suma += termino;
return suma;
}
```

Propuestos

Arcotangente: $x - x^3/3 + x^5/5 - \ldots$

Coseno: $1 - x^2/2! + x^4/4! - \ldots$

Logaritmo natural: $2(Z + Z^3/3 + Z^5/5 + \ldots)$ con $Z = (x-1)/(x+1)$