

Clase10: Objetos y Clases

Problema. Programar el diálogo indicado en el siguiente ejemplo:

```
fraccion 1 (Nº/Nº) ? 3/4
fraccion 2 (Nº/Nº) ? 6/8
suma=48/32=3/2
mayor=6/8
```

Nota. Suponga los sgtes métodos en la clase U

```
//abortar(x): escribe x y termina programa
static public void abortar(String x){
    println(x); //escribir epitafio ☹
    System.exit(0); //suicidarse ☹
}
//mcd(x,y): máximo común divisor entre x e y
static public int mcd(int x,int y){
    if(x==y) return x;
    return x>y ? mcd(y,x-y) : mcd(x,y-x); //☺
}
```

Solución 1. Con datos de tipo int

```
//obtener fraccion 1: a1/b1
String s=U.readLine("fraccion 1 (Nº/Nº)?");
int i=s.indexOf("/");
a1=Integer.parseInt(s.substring(0,i)),
b1=Integer.parseInt(s.substring(i+1));
if(b1==0) U.abortar("denominador cero");
//obtener fraccion 2: a2/b2
s=U.readLine("fraccion 2 (Nº/Nº)?");
i=s.indexOf("/");
int a2=Integer.parseInt(s.substring(0,i)),
b2=Integer.parseInt(s.substring(i+1));
if(b2==0) U.abortar("denominador cero");
//calcular y mostrar suma
int a=a1*b2+b1*a2, b=b1*b2, m=U.mcd(a,b);
U.println("suma="+a+"/"+b+"="+a/m+"/"+b/m);
//mostrar fraccion mayor
if(a1*b2>a2*b1){a=a1;b=b1;}else{a=a2;b=b2;}
U.println("mayor="+a+"/"+b);
```

Solución 2. Usando clase **Fracción** definida con los sgtes métodos:

Ejemplo de uso	resultado
Fraccion x=new Fraccion("3/4");	x: fracción 3/4
Fraccion y=new Fraccion("6/8");	y: fracción 6/8
x.suma(y);	fracción 48/32
y.simple();	fracción 3/4
x.toString();	string "3/4"
x.compareTo(y)	0 si x=y, n°<0 si x<y, n°>0 si x>y

```
//obtener fracciones
Fraccion
f1=new Fraccion(U.readLine("fraccion1 (Nº/Nº)?"),
f2=new Fraccion(U.readLine("fraccion2 (Nº/Nº)?"));

//calcular y mostrar suma
Fraccion f=f1.suma(f2);
U.println("suma="+f.toString()+"="+f.simple().toString());

//mostrar fraccion mayor
if(f1.compareTo(f2)>0) f=f1; else f=f2;
U.println("mayor="+f.toString());
```

Explicaciones

1. **Fraccion f1 = new Fraccion("nº/nº");**

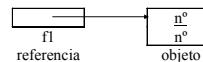
- crea objeto de clase Fracción con numerador y denominador indicados

- f1? referencia (dirección, ubicación, puntero, "flecha") a objeto

• equivalencia:
Fraccion f1; //declarar referencia a objeto

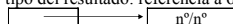


f1 = new Fraccion("nº/nº");//crear objeto y asignar dir

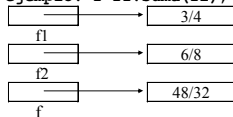


2. **f1.suma(f2)**

- suma fracciones f1 y f2
- objeto.método(parámetro)
- más preciso: referencia.método(parámetro)
- suma dos objetos de clase Fraccion
- resultado: objeto de clase Fracción
- tipo del resultado: referencia a objeto de clase Fraccion



• ejemplo: **f=f1.suma(f2);**



3. **f.toString()**

- convierte fracción f en string
- entrega String en la forma "nº/nº"

4. **f.simple().toString()**

- entrega string con fracción simplificada
- equivalencia: **Fraccion nn=f.simple(); nn.toString()**

5. **f1.compareTo(f2)**

- entrega 0 si f1=f2, n°<0 si f1<f2, n°>0 si f1>f2

6. **f=f2;**

- asignar (copiar) referencias (y no objetos)
- deja dos referencias apuntando al mismo objeto
- contenido original de f se pierde (objeto apuntado "desaparece")



Definición (declaración) de clase Fraccion

```
class Fraccion{ //en archivo Fraccion.java
//representacion
private int a,b; //numerador y denominador
//métodos
public String toString(){
    return a + "/" + b;
}
public Fraccion suma(Fraccion x){
    return new Fraccion(a*x.b+b*x.a, b*x.b);
}
public Fraccion simple(){
    int m=U.mcd(a,b);
    return new Fraccion(a/m, b/m);
}
public int compareTo(Fraccion x){
    return a*x.b - b*x.a; //resultado: 0, <0 o >0
}
```

```
//constructores: inicializan repres. de objeto
public Fraccion(int x,int y){
    a=x; b=y; validar();
}
public Fraccion(String x){
    int i=x.indexOf("/");
    if(i>=0){
        a=Integer.parseInt(x.substring(0,i));
        b=Integer.parseInt(x.substring(i+1));
        validar();
    }else{
        a=Integer.parseInt(x); b=1;
    }
}
private void validar(){//método interno
    if(b==0) U.abortar("denominador cero");
}
```

Explicaciones

1. private int a,b;

- representación/datos/atributos de cada objeto de clase Fraccion: numerador y denominador
- private? visible sólo en la clase
- ¿por qué private?
 - oculta representación
 - evita modificación (accidental o premeditada)
 - independiza programa usuario de representación

2. public String toString(){...}

- método que convierte objeto en String (devuelve String)
- ej invocación: f.toString()
- concatena numerador, slash y denominador
- public? visible también fuera de la clase

3. public Fraccion suma(Fraccion x){...}

- método que suma dos fracciones
- ej de invocación: f1.suma(f2)
- ref de segundo operando: x (datos: x.a, x.b)
- ref de primer operando: this (datos: this.a, this.b o a, b)
- crea nuevo objeto con resultado
- resultado: referencia a nuevo objeto de clase Fraccion

4. public Fraccion simple(){...}

- método que entrega fracción simplificada
- ej de invocación: f.simple()
- crea nueva fracción simplificada
- resultado: referencia a nuevo objeto de clase Fraccion

5. public int compareTo(Fraccion x){...}

- método que compara dos objetos (fracciones)
- devuelve un int <, = o > 0 (resultado de resta)

6. public Fraccion(String x){...}

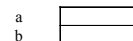
- inicializa componentes de un objeto
- método "constructor"
- mismo nombre de la clase
- recibe parámetro de clase String
- invocación: new Fraccion(string)
- se ejecuta después que operador new crea objeto

7. public Fraccion(int x,int y){...}

- inicializa un objeto con dos parámetros enteros
- método "constructor"
- invocación: new Fraccion(n°,n°)
- se ejecuta después que operador new crea objeto

Fraccion f=new Fraccion(n°,n°);

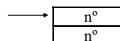
1° operador new crea objeto (reserva memoria para una fracción)



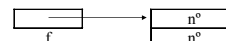
2° objeto se inicializa (invocando a método "constructor")



3° operador new devuelve referencia (dirección) de objeto



4° asignación guarda referencia



Problema. Completar la clase Fracción y el programa principal de modo que satisfaga el diálogo siguiente:

```
Fracción 1 (Nº/Nº) ? 3/8
Fracción 2 (Nº/Nº) ? 1/4
Operación (+ - * /) ? /
Resultado=3/2

static public void main(String[]args){
...
}
class Fraccion{
...
}
```

```
Fraccion
f1=new Fraccion(U.readLine("Fraccion1(nº/nº)?"),
f2=new Fraccion(U.readLine("Fraccion2(nº/nº)?"),
f; //resultado

String op=U.readLine("Operación (+ - * /)?");

if(op.equals("+"))
f=f1.suma(f2);
else if(op.equals("-"))
f=f1.resta(f2);
else if(op.equals("*"))
f=f1.mult(f2);
else if(op.equals("/"))
f=f1.div(f2);
else
U.abortar("operador incorrecto");

U.println("resultado="+f.simple().toString());
```

Solución 2. Con instrucción switch

```
Fraccion
f1=new Fraccion(U.readLine("Fraccion1(nº/nº)?"),
f2=new Fraccion(U.readLine("Fraccion2(nº/nº)?"),
f; //resultado

String op=U.readLine("Operación (+ - * /)?");

switch(op.charAt(0)){//switch admite char
case '+': f=f1.suma(f2); break;
case '-': f=f1.resta(f2); break;
case '*': f=f1.mult(f2); break;
case '/': f=f1.div(f2); break;
default: U.abortar("operación incorrecta");
}

U.println("resultado="+f.simple().toString());
```

```
class Fraccion
{
...
public Fraccion resta(Fraccion x){
return new Fraccion(a*x.b-b*x.a, b*x.b);
}
public Fraccion mult(Fraccion x){
return new Fraccion(a*x.a, b*x.b);
}
public Fraccion div(Fraccion x){
return new Fraccion(a*x.b, b*x.a);
}
}
```

Solución 2. Con métodos que modifican objeto

```
Fraccion
f1=new Fraccion(U.readLine("Fraccion1(nº/nº)?"),
f2=new Fraccion(U.readLine("Fraccion2(nº/nº)?").
f; //resultado

String op=U.readLine("Operación (+ - * /)?");

switch(op.charAt(0)){
case '+': f1.sumar(f2); break; //f1=f1.suma(f2);
case '-': f1.restar(f2); break;
case '*': f1.multiplicar(f2); break;
case '/': f1.dividir(f2); break;
default: U.abortar("operación incorrecta");
}
f1.simplificar(); //f1=f1.simple();

U.println("resultado="+f1.toString());
```

```
class Fraccion{
...
//métodos modifican objeto (no crean uno nuevo)
public void sumar(Fraccion x){
a=a*x.b+b*x.a; b=b*x.b;
}
public void restar(Fraccion x){
a=a*x.b-b*x.a; b=b*x.b;
}
public void multiplicar(Fraccion x){
a=a*x.a; b=b*x.b;
}
public void dividir(Fraccion x){
a=a*x.b; b=b*x.a;
}
public void simplificar(){
int m=U.mcd(a,b); a=a/m; b=b/m;
}
}
```