

# Uso de objetos

*Bibliotecas, objetos, clases y null*

Cátedra 6

Otoño 2005

Para aumentar la productividad al escribir programas, un programador puede recurrir a bibliotecas de objetos, es decir, lugares donde existe una gran variedad de objetos clasificados según su utilidad. Mediante estos objetos uno puede solucionar problemas de forma más sencilla

## 1 El problema

Se desea construir un programa que use aritmética de funciones.

## 2 La biblioteca

Revisando internet se encuentra una biblioteca de objetos que contiene Fraccion.class. Uno descarga el archivo al directorio de trabajo. Además, generalmente se puede descargar la documentación.

## 3 La documentación

La documentación de la clase Fracción, que acompaña a Fraccion.class, es la siguiente:

| Método                     | Descripción                                    |
|----------------------------|--|
| new Fraccion (int a,int b) | Crea un nuevo objeto Fraccion con el valor a/b |
| void escribir(Console c)   | Escribe la fraccion en la consola c            |
| void ponderar(int i)       | Pondera la fraccion por i                      |
| int comparar(Fraccion p)   | Devuelve la comparacion entre la fraccion y p  |
| Fraccion suma(Fraccion p)  | Devuelve la suma de la fraccion y p            |
| void sumar(Fraccion p)     | Le suma p a la fraccion                        |

## 4 La aplicación

Una vez que se tiene la documentación y el archivo con la clase, se puede emplear en un programa:

```

1 Console con=new Console();
2 con.println("Bienvenido al Sumador 2005");
3 con.println("Ingresa la primera fraccion ");
4 int a=con.readInt();
5 int b=con.readInt();
6 Fraccion f=new Fraccion(a,b);
7 con.println("Ingresa la segunda fraccion ");
8 a=con.readInt();
9 b=con.readInt();
10 Fraccion g=new Fraccion(a,b);
11 f.sumar(g);
12 con.print("El resultado es:");
13 f.escribir(con);
14 con.println("");

```

### 4.1 Descripción

6 Se crea una variable f que hará referencia a un objeto Fracción, que acaba de ser creado

10 Lo mismo para la variable g

11 Se le dice a f que le sume el valor de g a su propio valor

13 Se le dice a f que escriba su valor en la consola

## 5 Clases

No todos los objetos son fracciones. Hay objetos que saben archivar información en disco, estimar el área bajo una curva, tocar música, calcular el máximo de una función o multiplicar matrices. Tales objetos aceptan órdenes distintas de las que realiza una fraccion. El tipo

de órdenes que un objeto puede llevar a cabo depende de la fábrica en donde fueron manufacturados.

En Java, una fábrica de objetos se denomina clase. Por ejemplo la fábrica en donde se arman las fracciones se llama Fraccion (la primera F con mayúscula). La fábrica de objetos para almacenar información en disco se llama TextWriter, etc (en la práctica una clase es un trozo de código donde se define al objeto). Las clases se almacenan en s con la extensión .class.

## 6 Construcción de una Fraccion

Como acabamos de ver, en un programa se pueden fabricar objetos mediante **new**:

```
new Fraccion(int a,int b);
```

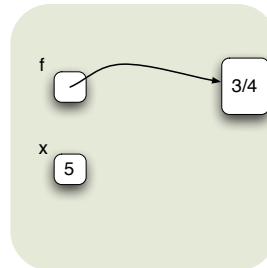
La evaluación de esta expresión construye un objeto en la fábrica de fracciones, con un valor inicial a/b. En general, entre los paréntesis se especifican los valores con los que debe ser inicializado el objeto (aún cuando todas las fracciones son producidos en la misma fábrica, no todas tienen el mismo valor).

De la fábrica uno no obtiene directamente el objeto, sino una referencia (numero celular al que se puede llamar para darle órdenes). La referencia de un objeto debe ser almacenada en variables del tipo adecuado. Por ejemplo para recordar la referencia de una fraccion se necesita colocarla en una variable de tipo Fraccion.

```
Fraccion f;  
f=new Fraccion(3,4);
```

```
int x=5;
```

No hay que olvidar que lo que se guarda en la variable f es una **referencia** a la fracción, como se esquematiza en la figura de la derecha.



## 7 Métodos e invocación de métodos

Cada objeto sabe recibir las órdenes para las que fue fabricado. Por ejemplo, un objeto Fracción sabe escribirse o es capaz de sumarse con otra:

```
f.sumar(g);
```

A esta categoría de expresión se le denomina invocación de un método. Uno le pide a la fracción f que le sume a su valor el valor de la fracción g. El objeto que es invocado (f) se llama objeto **this**.

No solo se pueden dar instrucciones, también se le puede pedir datos al objeto. Es el caso del método comparar, que devuelve -1 si f es menor que g, 0 si son iguales o 1 si f es mayor:

```
if (f.comparar(g)==0)  
    con.print("Las dos fracciones son iguales");
```

## 8 null

Existe un valor especial en Java que sirve para marcar que una variable no está haciendo referencia a ningún objeto: null. En el caso anterior, si se hace

```
f=null;
```

entonces la fraccion 3/4 queda sin nadie que la referencia... y es destruida

### AL MARGEN

#### Problemas propuestos

El objeto Tortuga tiene los siguientes métodos:

```
new Tortuga(double z, double y)  
void avanzar(double distancia)  
void girar(double angulo)
```

El primero crea una Tortuga “Gráfica” en la coordenada (z,y) en la pantalla. El segundo hace que la tortuga avance cierta distancia (y vaya dejando un rastro). El tercero, hace que la tortuga gire.

Sabiendo esto, haz el siguiente dibujo:

