

INTRODUCCIÓN A LA PROGRAMACIÓN ORIENTADA A OBJETOS



- Objetivo
- Objetos
- Programación Orientada a Objetos
- Conceptos básicos
 - Métodos
 - Constructores
 - Atributos
- Clases con métodos estáticos



3

Objetivo(s)

Conocer y aplicar los conceptos básicos y herramientas de la programación orientada a objetos para la resolución de distintas tareas.



Programación Orientada a Objetos (POO)

4

- La programación orientada a objetos o POO es un **paradigma** de programación que usa objetos y sus interacciones, para diseñar aplicaciones y programas informáticos.
- Está basado en varias técnicas, incluyendo herencia, abstracción, polimorfismo y encapsulamiento. Su uso se popularizó a principios de la década de los años 1990.

¿Qué es un objeto?

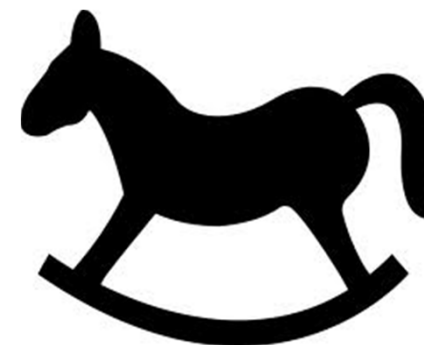
5



- En filosofía: un objeto es aquello que puede ser observado, estudiado y aprendido, en contraposición a la representación abstracta de ese objeto que se crea en la mente a través del proceso de generalización.
- Un objeto en POO representa alguna entidad de la vida real, es decir, alguno de los objetos que pertenecen al negocio con que estamos trabajando o al problema con el que nos estamos enfrentando, y con los que podemos interactuar.

Entidad de la vida real

6



Entidad de la vida real

7



Objetos

8

- Los objetos son entidades que tienen un determinado **estado** y un **comportamiento**
- El **estado** es el conjunto de valores que son asignados a las variables del objeto en un momento determinado.
- El **comportamiento** está definido por los métodos o mensajes a los que sabe responder dicho objeto, es decir, qué operaciones se pueden realizar con él.

Atributos

9

- ☐ Clase
- ☐ Longitud
- ☐ Capacidad
- ☐ Peso
- ☐ Aerolínea...



Métodos

10

- ☐ Abrir puertas
- ☐ Cerrar puertas
- ☐ Despegar
- ☐ Aterrizar
- ☐ Bajar llantas
- ☐ Subir llantas
- ☐ Prender luces...



Conceptos fundamentales

11

- **Clase:** Definiciones de las propiedades y comportamiento de un tipo de objeto concreto. La instanciación es la lectura de estas definiciones y la creación de un objeto a partir de ellas.

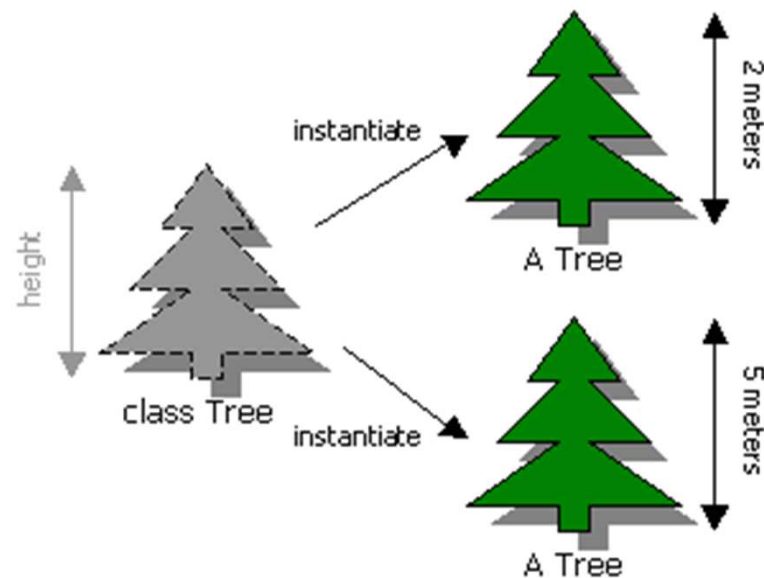


Fig. 1: Instantiating two Trees from the Tree class

En resumen

12



- La programación orientada a objetos se basa en que cada programa es ***una simulación de un mundo real o virtual.***
- Cada uno de estos mundos esta compuesto de ***objetos.***
- Los objetos se comunican a través de ***mensajes***
- Por lo tanto, un programa orientado a objetos no es mas que ***una configuración de un conjunto de objetos y los mensajes que se envían entre ellos.***
- El “molde” que guarda la descripción de todos los objetos de un mismo tipo es lo que conocemos como ***clase.***
- A su vez, las clases con propósitos similares pueden agruparse en ***paquetes.***
- La ejecución de un programa comienza en el método “***main***” de una clase.

```
import java.util.Scanner;

public class Ejercicio {

    public static void main(String[] args) {

        Scanner escaner = new Scanner (System.in);

        escaner.nextLine ();

    }

}
```

```
import java.util.Scanner;

public class Ejercicio {

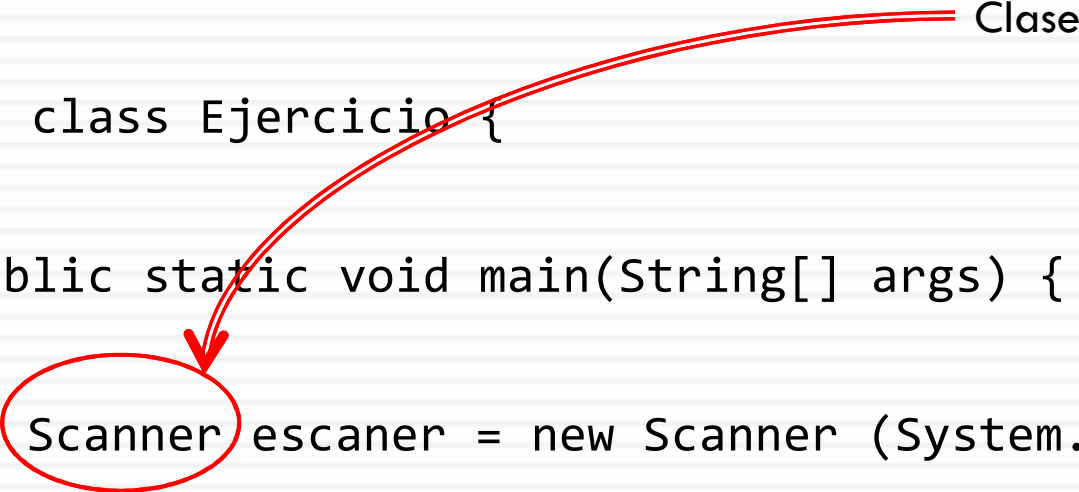
    public static void main(String[] args) {

        Scanner escaner = new Scanner (System.in);

        escaner.nextLine ();

    }

}
```



Clase

```
import java.util.Scanner;

public class Ejercicio {

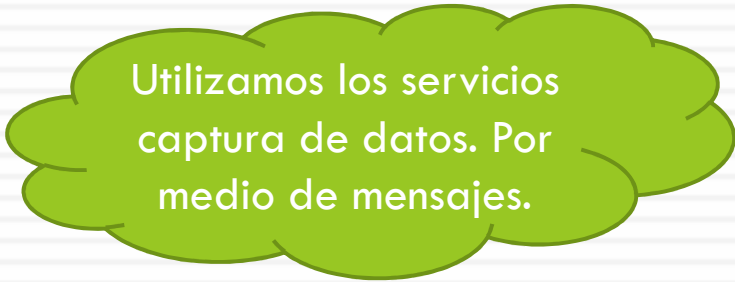
    public static void main(String[] args) {

        Scanner escaner = new Scanner (System.in);

        escaner.nextLine ();

    }

}
```



Utilizamos los servicios
captura de datos. Por
medio de mensajes.

Ejemplo

16

- Vamos a crear una clase que maneje fracciones aritméticas.

- Abstracción:
 - Todas las fracciones aritméticas tienen: Numerador, Denominador.


$$\frac{A}{B}$$

- Atributos:

- Numerador
- Denominador

- Métodos

- Obtener parte entera
- Simplificar
- Obtener representación decimal
- Obtener representación en cadena de la fracción
- Obtener fracción equivalente a partir de un multiplicador.

Programación Orientada a Objetos

17

- La POO es similar a crear cajas negras que realizan operaciones.



Programación Orientada a Objetos

18

- Cajas que contienen un estado interno persistente.

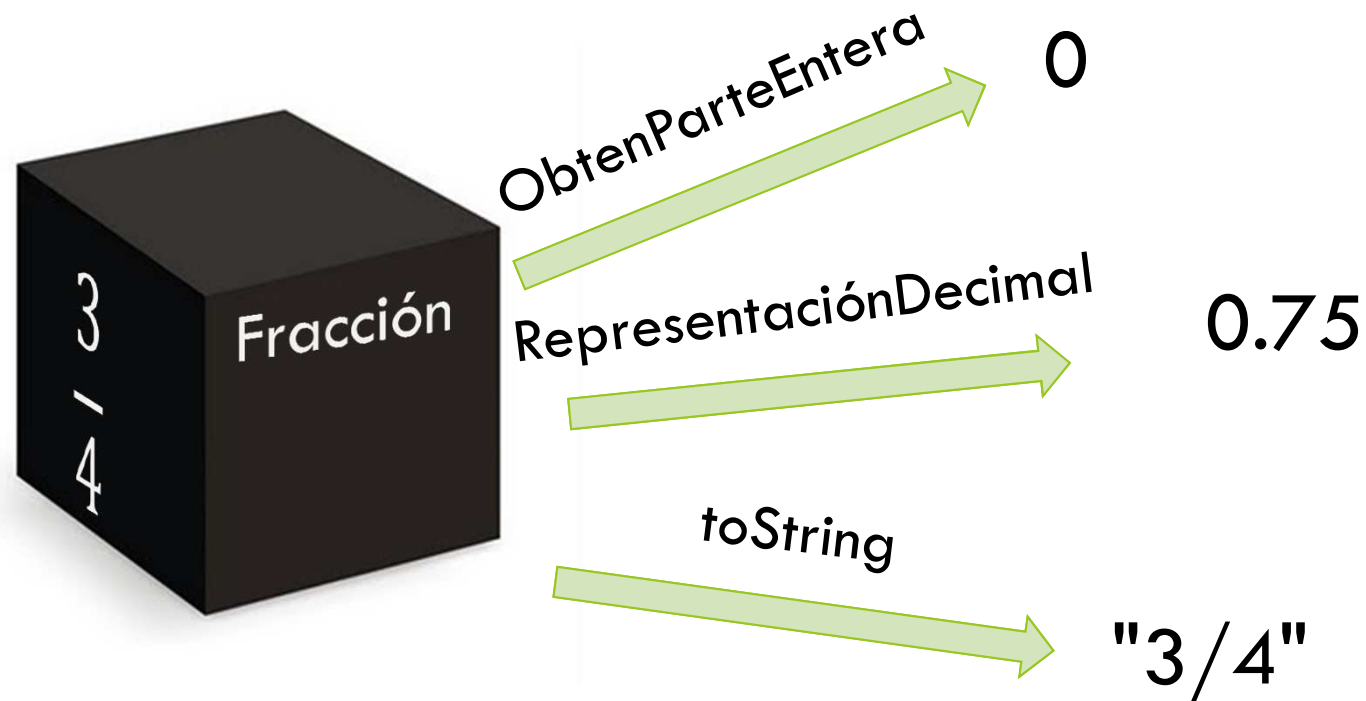
$\frac{3}{4}$



Programación Orientada a Objetos

19

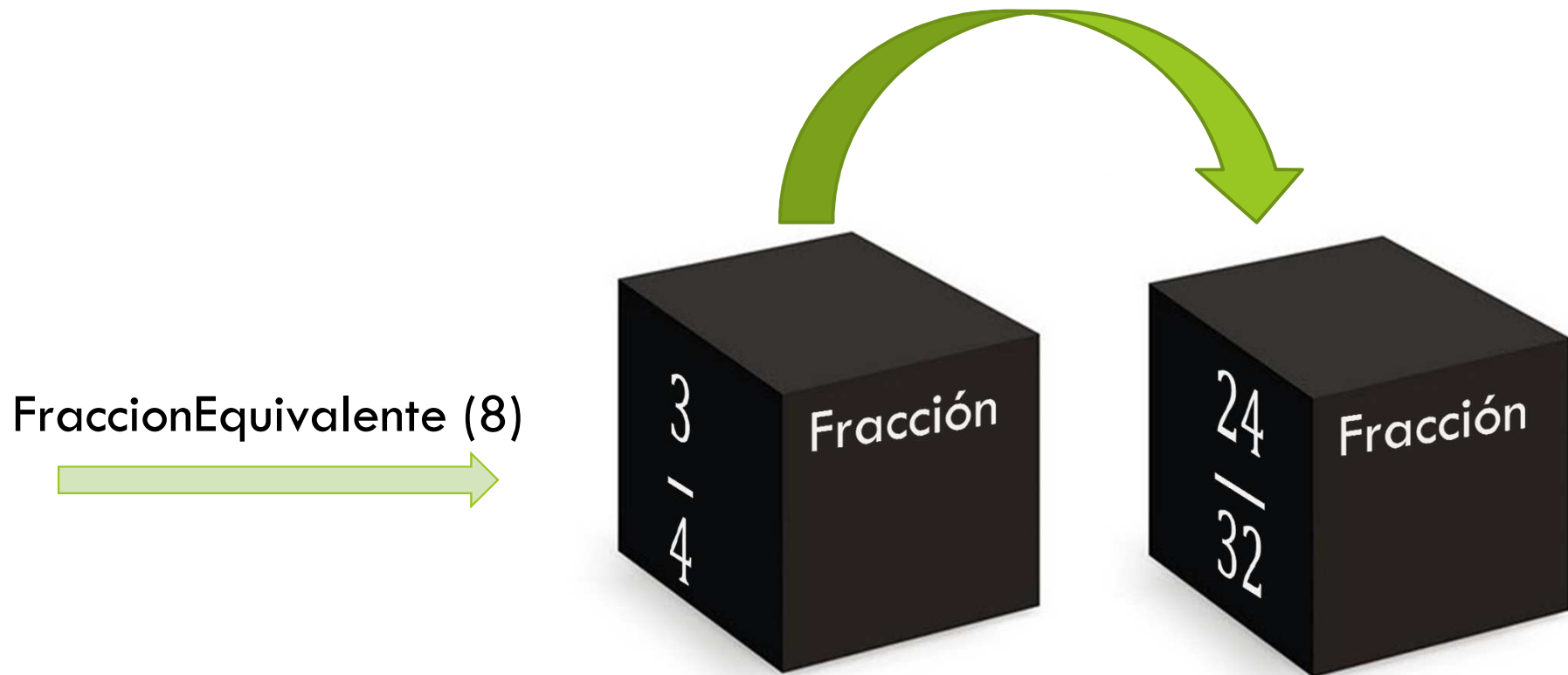
- Y que pueden realizar operaciones en base a dicho estado y regresar algún tipo de valor.



Programación Orientada a Objetos

20

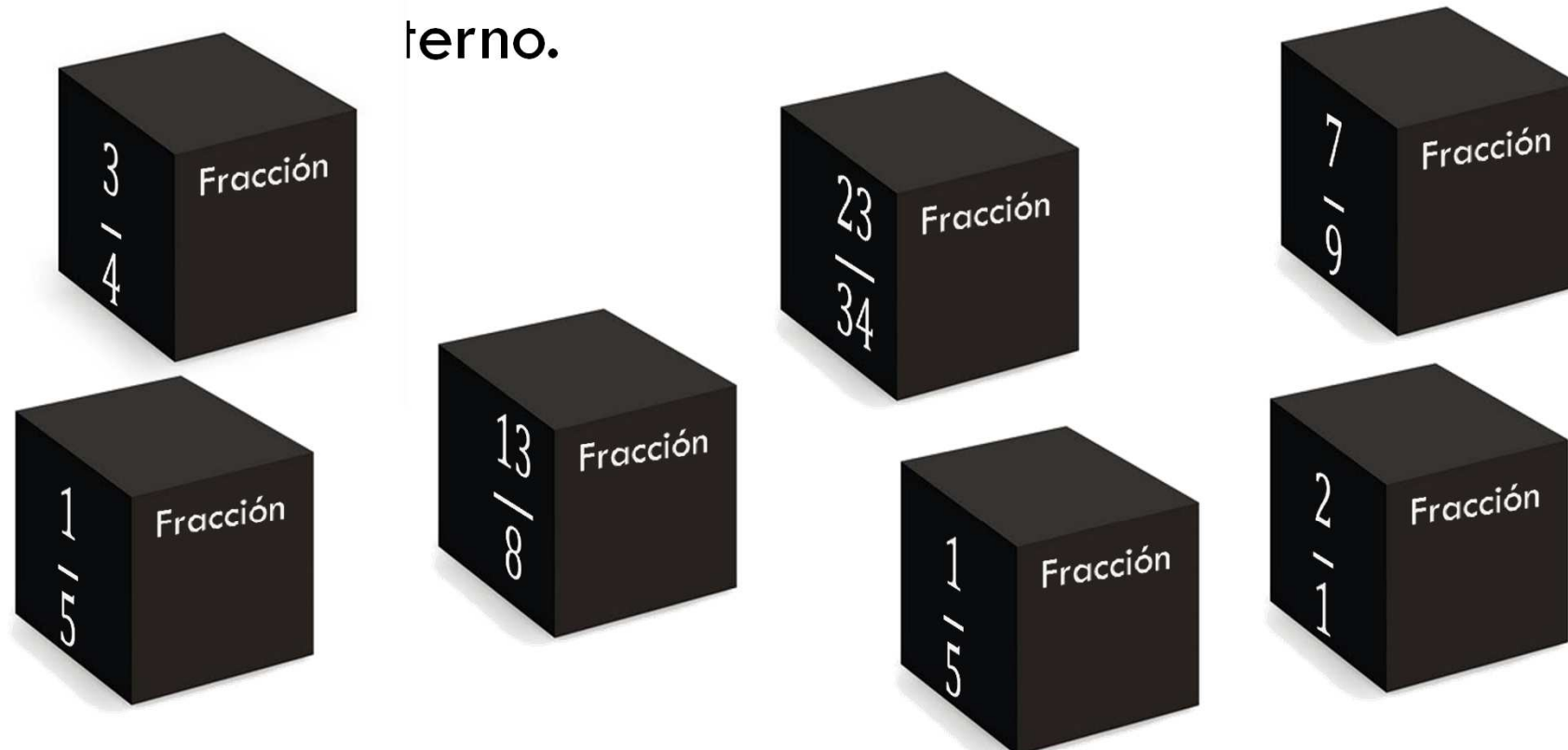
- El estado interno del objeto puede cambiar si su interfaz pública nos ofrece un método para ello.



Programación Orientada a Objetos

21

- Gracias a esto, podemos crear muchos objetos que se comportan de la misma manera, pero sus procesamientos dependen directamente de su terno.



Creando la clase

22

- ❑ Nuevo proyecto llamado Fracciones.
- ❑ Creamos una nueva clase Java llamada Fraccion
- ❑ Definiendo los atributos.

```
public class Fraccion {  
  
    private int numerador;  
    private int denominador;  
}
```

Encapsular campos

23

```
* @author yhavhe
*/
public class Fraccion {

    private int numerador;
    private int denominador;

}
```

Debugger Console × EstadísticasMorfologicas

Breakpoint DBWords

er program running

Breakpoint hit at line

read main stopped at

er program running

Breakpoint hit at line

read main stopped at

er program running

Breakpoint hit at line

read main stopped at

Breakpoint DBWords

er program running

Encapsulate Fields...

Move Inner to Outer Level...

Convert Anonymous to Member...

Introduce Variable... Alt+Mayúsculas+V

Introduce Constant... Alt+Mayúsculas+C

Introduce Field... Alt+Mayúsculas+E

Introduce Method... Alt+Mayúsculas+M

Introduce Parameter... Alt+Mayúsculas+P

Inspect and Transform...

Undo

Redo

Format Alt+Mayúsculas+F

Run File Mayúsculas+F6

Debug File Ctrl+Mayúsculas+F5

Test File Ctrl+F6

Debug Test File Ctrl+Mayúsculas+F6

Run Focused Test

Debug Focused Test

Run Into Method

New Watch... Ctrl+Mayúsculas+F7

Toggle Line Breakpoint Ctrl+F8

Profiling

Cut Ctrl+X

Copy Ctrl+C

Paste Ctrl+V

Code Folds

Select in Projects

mitted.

fologicas.BD.DBLemma by thread main.

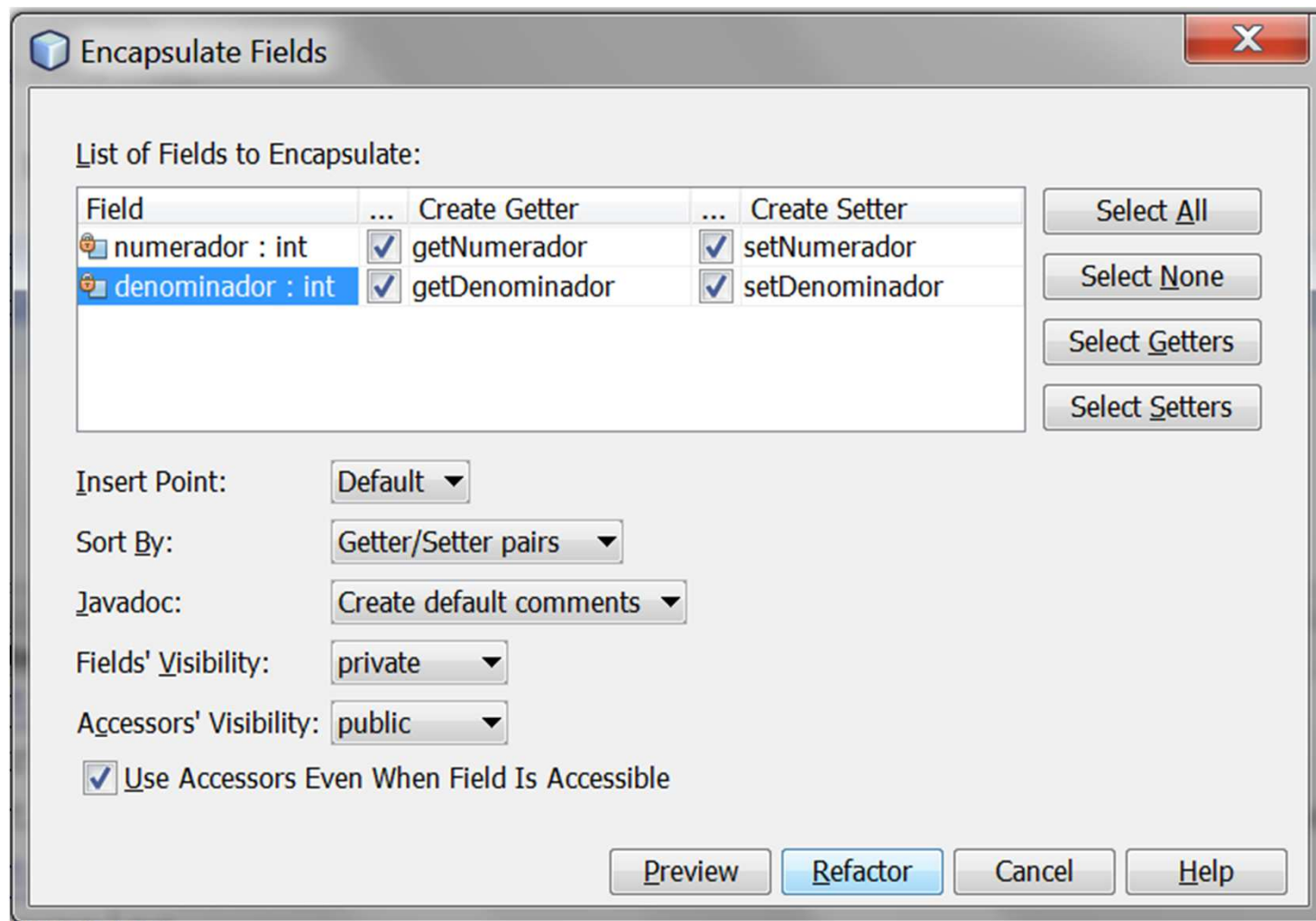
fologicas.BD.DBLemma by thread main.

mitted.

EstadísticasMorfologicas (run)

Encapsulando campos

24



Definiendo Métodos

25

- Obtener parte entera

$$\frac{8}{3} = 2\frac{2}{3}$$

- Dividimos de manera entera el numerador entre el denominador y regresamos el resultado.

Creando el método

26

```
public class Fraccion {  
  
    private int numerador;  
    private int denominador;  
  
    public int obtenerParteEntera () {  
        int parteEntera = numerador/denominador;  
        return parteEntera;  
    }  
  
    // Sets y Gets...  
  
}
```

Método simplificar

27

- Encontrar una fracción equivalente, con el numerador y denominador mas pequeños posibles.

$$\frac{100}{90} = \frac{50}{45} = \frac{10}{9}$$

- Buscamos un común divisor para el numerador y el denominador y los dividimos a ambos.
- Repetimos la operación hasta que no se encuentre un divisor común para el numerador y el denominador.

Creando el método

28

```
public class Fraccion {  
  
    private int numerador;  
    private int denominador;  
  
    public void simplificar (){  
  
        int divisor = 2;  
        while (divisor < numerador && divisor < denominador){  
            if (numerador % divisor == 0 && denominador % divisor == 0){  
  
                numerador = numerador/divisor;  
                denominador = denominador/divisor;  
            }  
            else  
                divisor++;  
        }  
    }  
}  
// Método obtener parte entera y sets y gets.
```

Definiendo método

29

- Obtener representación decimal.
- El método regresa la división fraccionaria (con punto decimal) del numerador entre el denominador.

$$\frac{1}{2} = 0.5$$

Creando el método

30

```
public class Fraccion {  
  
    private int numerador;  
    private int denominador;  
  
    public double getRepresentacionDecimal () {  
  
        double representacionDecimal =  
  
            (double) numerador / (double) denominador;  
  
        return representacionDecimal;  
  
    }  
  
}
```

...

```
// Método obtener parte entera y sets y gets.
```

miriam.balbuena@gmail.com

www.cic.ipn.mx

31/05/2013

Mostrar la fracción como cadena

31

```
public class Fraccion {  
  
    private int numerador;  
    private int denominador;  
  
    public String toString (){  
        String representacionCadena =  
            ""+numerador+"/"+denominador;  
        return representacionCadena;  
    }  
    ...  
}
```

Obtener una fracción equivalente a partir de un multiplicador.

32

- Se recibe un número que multiplique tanto al numerador como al denominador para obtener una fracción equivalente.

- Ejemplo:

- Multiplicador = 3

$$\left(\frac{5}{7}\right) \times 3 = \frac{5 \times 3}{7 \times 3} = \frac{15}{21}$$

Obtener fracción equivalente

33

```
public class Fraccion {  
  
    private int numerador;  
    private int denominador;  
  
    public void convierteFraccionEquivalente (int multiplicador){  
        numerador = numerador * multiplicador;  
        denominador = denominador * multiplicador;  
    }  
  
    public String toString (){  
        //String representacionCadena = ""+numerador+"/"+denominador;  
        String representacionCadena = "Obla di Obla da life goes on..";  
        return representacionCadena;  
    }  
    ...  
}
```

Probando los métodos

34

- En la clase `Fracciones.java` (la que se creó automáticamente cuando se creó el proyecto) en el método `main`.

Probando los métodos.

35

```
public class Fracciones {  
  
    public static void main(String[] args) {  
  
        Fraccion unaFraccion = new Fraccion ();  
        System.out.println ("Mi fracción es: "+unaFraccion);  
        unaFraccion.setNumerador(80);  
        unaFraccion.setDenominador(30);  
        System.out.println ("Mi fracción es: "+unaFraccion);  
        System.out.println ("Representación Decimal: "  
                               +unaFraccion.getRepresentacionDecimal());  
        System.out.println ("Parte entera: "  
                               +unaFraccion.obtenerParteEntera());  
        unaFraccion.simplificar();  
        System.out.println ("La fracción simplificada: "  
                               +unaFraccion);  
        unaFraccion.convierteFraccionEquivalente(5);  
        System.out.println ("Fracción equivalente: "+unaFraccion);  
    }  
}
```

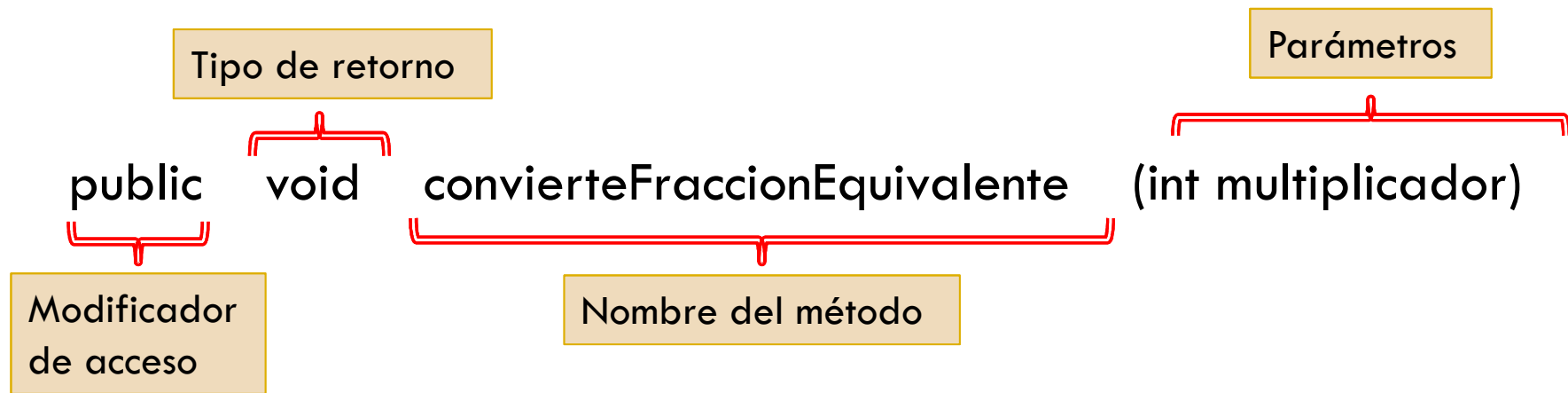
Interfaz pública

36

```
1
2 package fracciones;
3
4
5 public class Fraccion {
6     private int numerador;
7     private int denominador;
8
9     public void convierteFraccionEquivalente (int multiplicador) {...}
10    public String toString () {...}
11
12    public double getRepresentacionDecimal () {...}
13
14    public void simplificar () {...}
15
16    public int obtenerParteEntera () {...}
17
18    /**...*/
19    public int getNumerador() {...}
20
21    /**...*/
22    public void setNumerador(int numerador) {...}
23
24    /**...*/
25    public int getDenominador() {...}
26
27    /**...*/
28    public void setDenominador(int denominador) {...}
29
30 }
```

Definición de los métodos

37



Ejercicio



38

- Crear un método en la clase fracción que se llame "invertir".
- El método debe invertir el numerador y el denominador.

- Ejemplo: $\frac{3}{4} \rightarrow \frac{4}{3}$
- Probar:



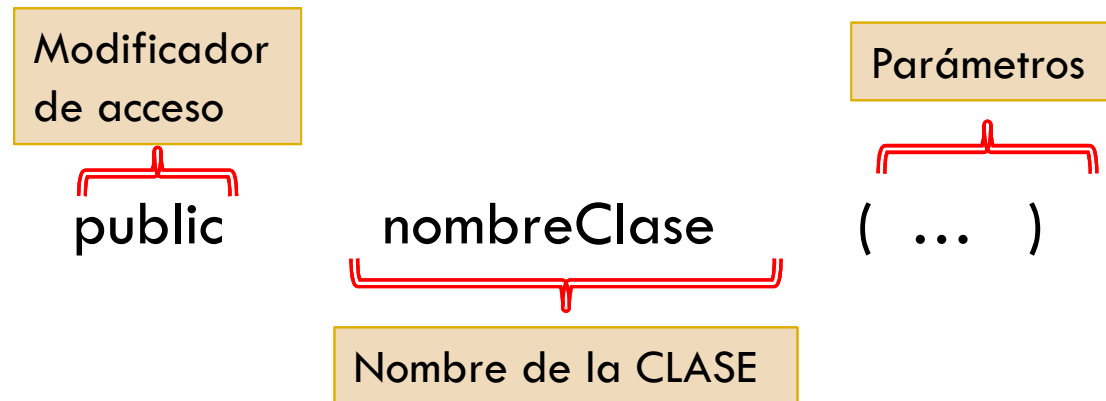
```
unaFraccion.invertir ();  
System.out.println ("Fraccion invertida: "+ unaFraccion);
```

Constructores

39

- Métodos para inicializar los atributos de la clase
- Se ejecutan automáticamente cuando se invoca la creación del objeto con el operador "new"

- Sintaxis:



Creando constructores para la clase fracción.

40

```
public class Fraccion {  
    private int numerador;  
    private int denominador;
```

```
  
    public Fraccion () {  
        numerador = 1;  
        denominador = 1;  
    }
```

```
    ...
```


Creando constructores para la clase fracción.

41

```
public class Fraccion {  
    private int numerador;  
    private int denominador;  
  
    public Fraccion (int n, int d){  
        numerador = n;  
        if (d != 0){  
            denominador = d;  
        } else {  
            throw new RuntimeException (  
                "El denominador no puede ser cero");  
        }  
    }  
}
```

Probando el constructor

42

```
public static void main(String[] args) {  
  
    Fraccion f = new Fraccion ();  
    System.out.println (f);  
  
    Fraccion f2 = new Fraccion (0,10);  
    System.out.println (f2);  
  
    f2.setDenominador (0);  
    System.out.println (f2);  
  
    Fraccion f3 = new Fraccion (10, 0);  
    System.out.println (f3);  
  
}
```

Cambiando el setDenominador

43

```
public void setDenominador (int denominador){  
    if (denominador != 0){  
        this.denominador = denominador;  
    }  
    else{  
        throw new RuntimeException (  
            "El denominador no puede ser cero");  
    }  
}
```

Probando el constructor

44

```
public static void main(String[] args) {  
  
    Fraccion f = new Fraccion ();  
    System.out.println (f);  
  
    Fraccion f2 = new Fraccion (0,10);  
    System.out.println (f2);  
  
    f2.setDenominador (0);  
    System.out.println (f2);  
  
    Fraccion f3 = new Fraccion (10, 0);  
    System.out.println (f3);  
  
}
```

Ejercicio

45

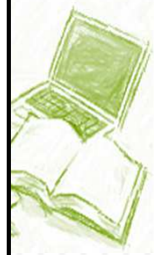
- Agregar un constructor que reciba dos cadenas llamadas ns y ds.
- El constructor debe convertir las cadenas a enteros y asignarlos al numerador y al denominador respectivamente.



int n = new Integer (cadena);

- Para probar el constructor, agregar el siguiente código al main de la clase Fracciones:

```
Fraccion fraccion = new Fraccion ("2", "8");  
System.out.println ("La última fracción es: "+ fraccion);  
  
Fraccion fraccion2 = new Fraccion ("2", "0");  
System.out.println ("La última fracción es: "+ fraccion);
```



Clases con métodos estáticos

46

- Se pueden definir clases que no instancien objetos.
- Los métodos se declaran con la palabra reservada: "static"
- Para ejecutar dichos métodos se usa el nombre de la clase y posteriormente el nombre del método.

Ejemplo

47

- Definir una clase con métodos estáticos que permita operaciones entre fracciones.

- Los métodos que debe proveer son:
 - Suma
 - Resta
 - Multiplicación
 - División

- Crear una clase llamada: OperacionesFraccionarias

Creando la clase con métodos estáticos:

48

```
public class OperacionesFraccionarias {  
  
    public static Fraccion multiplicacion(Fraccion fraccion1,  
                                         Fraccion fraccion2){  
        Fraccion fraccionResultado;  
  
        int numeradorResultado =  
            fraccion1.getNumerador()*fraccion2.getNumerador();  
  
        int denominadorResultado =  
            fraccion1.getDenominador()*fraccion2.getDenominador();  
  
        fraccionResultado =  
            new Fraccion(numeradorResultado, denominadorResultado);  
  
        return fraccionResultado;  
    }  
}
```

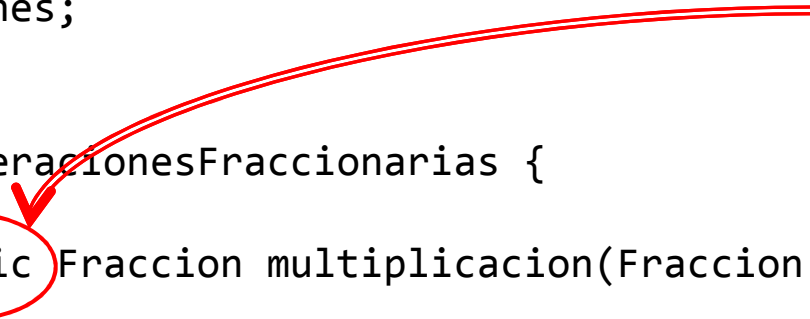

Creando la clase con métodos estáticos:

49

```
package fracciones;

public class OperacionesFraccionarias {
    public static Fraccion multiplicacion(Fraccion fraccion1, Fraccion fraccion2){
        Fraccion fraccionResultado;
        int numeradorResultado =
            fraccion1.getNumerador()*fraccion2.getNumerador();
        int denominadorResultado =
            fraccion1.getDenominador()*fraccion2.getDenominador();
        fraccionResultado =
            new Fraccion(numeradorResultado, denominadorResultado);
        return fraccionResultado;
    }
}
```

Palabra reservada static



Probando la clase OperacionesFraccionarias

50

- En el método main de la clase Fracciones agregar el siguiente código:

```
Fraccion unaFraccion = new Fraccion (1, 2);
Fraccion otraFraccion = new Fraccion (3, 4);

Fraccion resMult =
    OperacionesFraccionarias.multiplicacion(unaFraccion,
                                              otraFraccion);

System.out.println ("Resultado de la multiplicacion:"+resMult);

resMult.simplificar();
System.out.println ("Resultado simplificado: "+resMult);
```

Ejercicio

51

- Implementar los métodos estáticos para la suma, la resta y la división de fracciones en la clase OperacionesFraccionarias.

- Suma:
$$\frac{a}{b} + \frac{c}{d} = \frac{(d * a) + (b * c)}{b * d}$$

- División
$$\frac{a}{b} \div \frac{c}{d} = \frac{a * d}{b * c}$$

Ejercicios.



52

- ❑ Crea la clase Rectangulo representado por su longitud y su anchura.
- ❑ La longitud y anchura no pueden ser negativos.
- ❑ Agrega el código para que la representación como cadena del Rectangulo sea su longitud y anchura separada por comas.
- ❑ Código para probarlo:

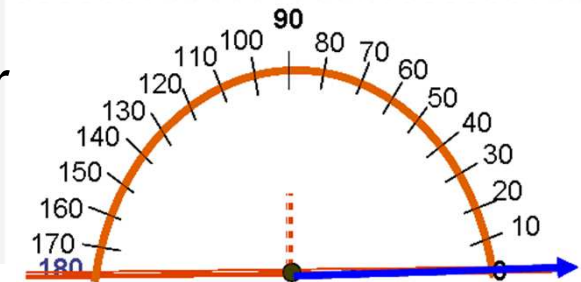
```
Rectangulo unRectangulo = new Rectangulo (8, 10);  
System.out.println (unRectangulo); // Rectangulo [8, 10]  
System.out.println ("Area = "+  
                    unRectangulo.area ()); // Area = 80  
System.out.println ("Perimetro = "+  
                    unRectangulo.perimetro ()); // Perimetro = 36;  
unRectangulo.setLongitud (-100); // Error  
unRectangulo = new Rectangulo (-5, 10); // Error
```

Ejercicios



53

- ★ □ Elaborar la clase Angulo, que modela ángulos en grados ($0^\circ - 360^\circ$). La clase debe ofrecer un método para obtener su equivalencia en radianes y otros métodos para sumar y restar otro ángulo o un valor entero.
- El ángulo nunca debe ser negativo ni superar 360°



```
Angulo angulo = new Angulo (45);  
System.out.println ("Radianes: "+angulo.radianes());  
angulo.suma (360); // 45  
System.out.println ("Angulo mas 360: "+angulo); // 45  
Angulo otroAngulo = new Angulo (270);  
angulo.resta (otroAngulo);  
System.out.println ("Angulo menos "+otroAngulo+": "+angulo); //135
```

Ejercicios

54

- ★ □ Crea la clase PUNTO con los métodos para obtener su representación cartesiana "(x,y)" y su representación polar "(r, α)" en una cadena de caracteres, entre paréntesis y separadas por una coma. Agregar también un método para calcular la distancia euclideana a otro punto. Definir un constructor que reciba x, y.

```
Punto unPunto = new Punto (1, 2);
System.out.println ("Cartesiana: "+unPunto.cartesiana());
System.out.println ("Polar: "+unPunto.polar());

Punto otroPunto = new Punto (3,5);
double distancia = unPunto.distancia (otroPunto);
System.out.println ("Distancia entre puntos: "+distancia);
distancia = otroPunto.distancia (unPunto);
System.out.println ("Distancia entre puntos: "+distancia);
```