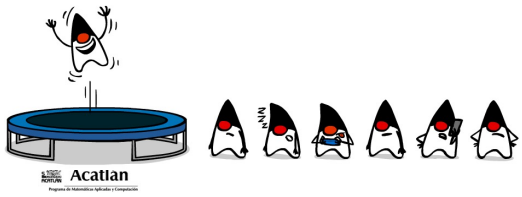
Creando Objetos (instancias)

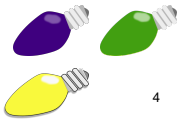


2

Ejercicio

- Codifica el contenido de la clase `Foco`
  - Métodos Constructores
    - Asigna watts y si el foco está prendido o apagado
    - Crea focos fundidos
  - Métodos
    - fundir: Funde el foco, con vatios en cero y apagado
    - apagar: Apaga el foco
    - prender: Prende el foco
    - calcularConsumo: Si está fundido regresa 0, si no multiplica los vatios por las horas
    - estaPrendido: Indica si el foco está prendido

Foco
+ vatios : int - prendido : boolean ~ fundido : boolean
+ Foco(w:int,estado:boolean) + Foco( ) + fundir( ) + apagar( ) + prender + calcularConsumo(horas:double):double + estaPrendido():boolean



4

```java
package foco;
public class Foco {
    public int vatios;
    private boolean prendido;
    boolean fundido;

    public Foco (int w, boolean estado){
        vatios = w;
        prendido = estado;
    }

    public Foco()
    {
      fundido = true;
      vatios = 0;
    }

    public void fundir(){
      fundido = true;
      vatios = 0;
    }

    public boolean estaPrendido(){
        if(prendido)
            return(true);
        else
            return(false);
    }
```

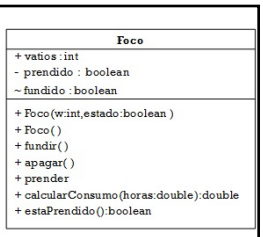
```java
public void apagar(){
    prendido = false;
}

public void prender(){
    prendido = true;
}

public double calcularConsumo(double horas){
    double calculo;
    if (fundido)
    {
        calculo = 0.0;
    }
    else
    {
        calculo = vatios*horas;
    }
    return(calculo);
}
```


6

Ejemplo de clase y objeto




```java
public class PruebaFoco {
    public static void main(String[] args) {
        Foco foco_uno = new Foco();
        Foco foco_dos = new Foco(100,false);
        Foco foco_tres = new Foco(75,true);
        System.out.println(foco_uno.estaPrendido());
        System.out.println("En diez horas: "+foco_dos.calcularConsumo(10));
        foco_dos.vatios=60;
        System.out.println("En diez horas: "+foco_dos.calcularConsumo(10));
        System.out.println(foco_tres.estaPrendido());
        foco_tres.apagar();
        System.out.println(foco_tres.estaPrendido());
        foco_tres.fundir();
        System.out.println(foco_tres.estaPrendido());
    }
}
```

```
run:
false
En diez horas: 1000.0
En diez horas: 600.0
true
false
false
BUILD SUCCESSFUL (tota
```

Clase — Objetos o Instancias

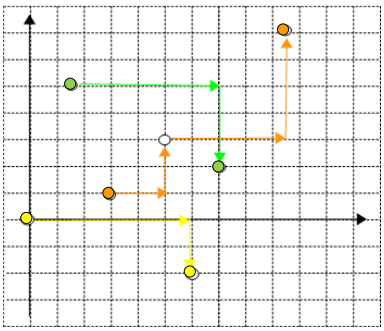
Foco	foco_uno	foco_dos	foco_tres
vatios			
prendido			
fundido			

8

Ejercicio métodos punto

1. Haz el código de los métodos del diagrama, en tu clase `Punto`

Punto
~ x : double ~ y : double
+Punto(cx:double, cy:double) +Punto() +Punto(coord[]:double) +mover(mov_x:double, mov_y:double) +distancia(p:Punto):double +mostrarPunto()

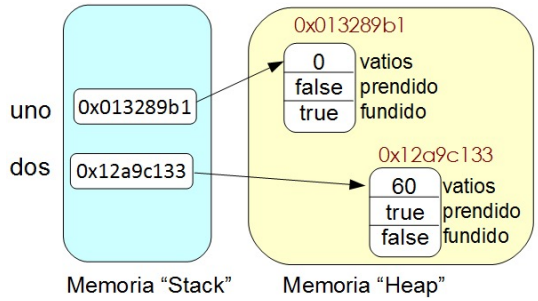


2. En un Java `main class`:
- Obtén la distancia entre los puntos amarillo-verde y verde-naranja
- Programa el recorrido de los puntos y muestra la posición final

Manejo de memoria en Java

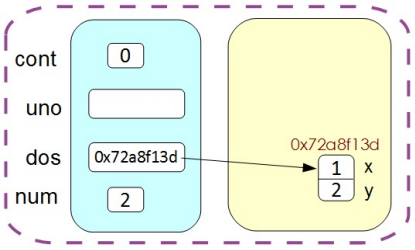
- Stack
  - Almacena valores de tipos de datos primitivos
  - Almacena la referencia (dirección) de un objeto
- Heap
  - Almacena los valores de los objetos

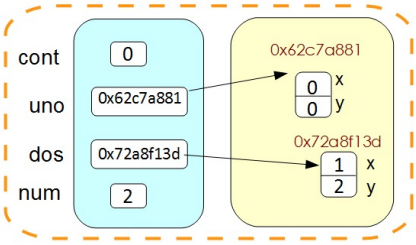
```
Foco uno, dos;
uno= new Foco();
dos= new Foco(60,true);
```

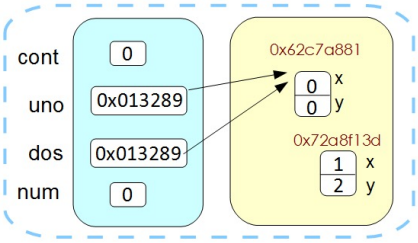


Manejo de memoria en Java… ejemplos

```
int cont=0, num=2;
Punto uno;
Punto dos = new Punto(1,2);
uno = new Punto(0,0);
dos = uno;
num = cont;
```



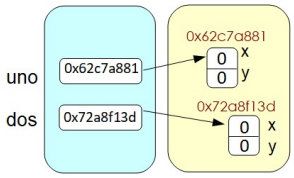




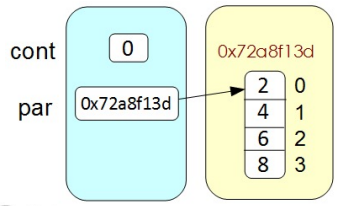
14

Manejo de memoria en Java… ejemplos

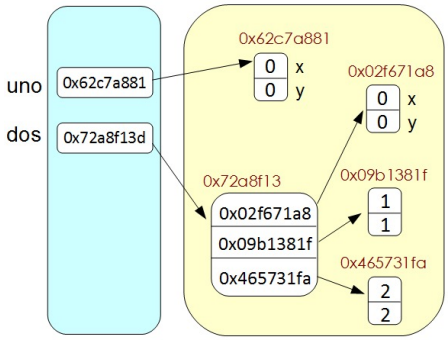
```
dos.x = uno.x;
dos.y = uno.y;
```




```
int cont=0;
int par[]={2,4,6,8}
```

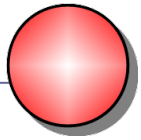



```
Punto uno= new Punto(0,0);
Punto [] dos={ new Punto(0,0),
               new Punto(1,1),
               new Punto(2,2)};
```



16

Más sobre clases

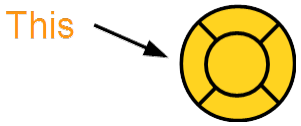


- Clase Círculo
  - Atributos o variables estado (datos)
    - coordenadas (x,y) del centro
    - Radio
- Métodos (comportamiento)
  - Constructores
  - Cálculo del perímetro
  - Cálculo del área
  - Elegir mayor de dos círculos
  - Imprimir coordenadas del centro

Circulo
+ x : double + y : double + r : double +numCirculos:int
+Circulo(x:double,y:double,r:double) +Circulo() +Circulo(r:double) +Circulo(c:Circulo) +perimetro():double +area():double +mayor(c:Circulo):Circulo +imprimirCentro()

this

- Todo objeto tiene una referencia a si mismo, es un argumento implícito
- Se vuelve explícito con la palabra reservada **this**
  - Su uso aumenta la claridad
  - Se usa cuando algún argumento o variable local del método tiene el mismo nombre en la variable de clase



this (cont)

```
public class Circulo {
  public double x,y,r;  // centro y radio
  public Circulo(double  x, double y, double r) {
    this.x = x;
    this.y = y;
    this.r = r;
  }
  public double perimetro(){ return 2 *3.14159 * r;}
  public double area(){ return 3.14159 * r * r;}
  public void imprimirCentro() {
    System.out.println("("+x+","+y+")");
  }
}
```


Múltiples constructores

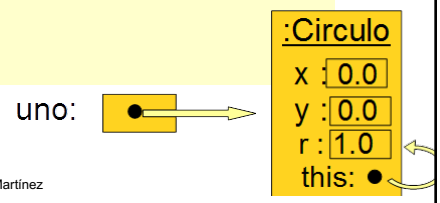
- Sobrecarga de métodos
  - Son métodos con el mismo nombre
  - El compilador elige el apropiado por orden y tipo de argumentos

```
public class Circulo {
  public double x,y,r;  // centro y radio
  public Circulo(double x, double y, double r){
     this.x=x; this.y = y; this.r=r;
  }
  public Circulo(){
     x=0.0;  y=0.0;  r=1.0;
  }
  public Circulo(double r){
     x=0.0;  y=0.0; this.r=r;
  }
  public Circulo(Circulo c){
     x=c.x;  y=c.y;  r=c.r;
  }
  public double perimetro(){ return 2 *3.14159 * r;}
  public double area(){ return 3.14159 * r * r;}
}
```

Otro uso de `this`

- Un constructor invoca a otro en la misma clase

```
public Circulo(double x, double y, double r){
    this.x=x; this.y = y; this.r=r;
}
public Circulo(){
    this(0.0,0.0,1.0);
}
public Circulo(double r){
    this(0.0,0.0,r);
}
public Circulo(Circulo c){
    this(c.x,c.y,c.r);
}
```



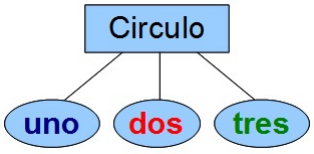

```
Circulo uno = new Circulo();
```


26

Ejemplo: Clase y objetos

```
public class Circulo {
  private double x,y,r ;
  public Circulo (double  x, double y, double r) {
      this.x=x; this.y = y; this.r=r; }
  public Circulo (double r) { this(0.0,0.0,r); }
  public Circulo (Circulo c) { this(c.x,c.y,c.r); }
  public Circulo () { this(0.0,0.0,1.0); }
  public void imprimirCentro() {
              System.out.println("("+x+","+y+")"); }
}
```

```
public class PruebaCirculo {
  public static void main(String[] args)
    Circulo uno = new Circulo(1,2,3);
    Circulo dos = new Circulo(6);
    Circulo tres = new Circulo(uno);
    uno.imprimirCentro();
    dos.imprimirCentro();
    tres.imprimirCentro();
  }
}
```



28

Ejercicio métodos Reloj

1. Usa `this` para los métodos constructores de la clase `Reloj`
2. Crea los métodos conocidos como “setters” (modificadores) y “getters” (consultores):
   `setHh, setMm, setSs`: Reciben valor a colocar en el atributo correspondiente
   `getHh, getMm, getSs`: Regresan el valor del atributo correspondiente
3. Método `validarHora`, regresa un valor de TRUE si los rangos de hora (0-23), minutos y segundos (0-59) son correctos, en caso contrario envía FALSE.

1. Usa `this` para los métodos constructores de la clase `Reloj`
2. Crea los métodos conocidos como “setters” (modificadores) y “getters” (consultores):
   `setHh, setMm, setSs`: Reciben valor a colocar en el atributo correspondiente
   `getHh, getMm, getSs`: Regresan el valor del atributo correspondiente
3. Método `validarHora`, regresa un valor de TRUE si los rangos de hora (0-23), minutos y segundos (0-59) son correctos, en caso contrario envía FALSE.

- Método `imprimir` que desplegará la leyenda "Son las hh con mm minutos y ss segundos AM o PM"
- En un java `main class`, crea un objeto de tipo Reloj, si la hora con que fue creado es correcta, imprime la hora, si no imprime la frase "Hora incorrecta".

Variables de clase

- Semejante a una variable global en C
  - Se puede llamar desde otra clase.
- Se usa la palabra reservada `static`

```
public class Circulo {
  static int numCirculos = 0;
  public double x,y,r ;  //  centro y radio
  public Circulo (double  x, double y, double r) {
    this.x=x; this.y = y; this.r=r;
    numCirculos++;
  }
  public Circulo (double r) {this(0.0,0.0,r);}
  public Circulo (Circulo c) {this(c.x,c.y,c.r);}
  public Circulo () {this(0.0,0.0,1.0);}
  public double perimetro() { return 2 *3.14159 * r;}
  public double area() { return 3.14159 * r * r;}
  public void imprimirCentro() {
    System.out.println("("+x+","+y+")");
  }
}
```

Variables de clase constantes

- Usar una variable de clase
  - No se crea una referencia, se usa directamente

```
System.out.print("círculos creados"+Circulo.numCirculos);
```

- Constantes
  - Equivalente al "#define" de C
  - Se escriben en mayúsculas
  - Se usa la palabra reservada `final`
- Un ejemplo es la clase Math
  - `Math.PI`

```
public static final double PI=3.141592653589793238;
```


38

Métodos de clase: métodos estáticos.

- Se usa la palabra **static**
- No pasan una referencia implícita tipo this, se invocan con nombre de la clase
- En el diagrama UML, se subrayan

```
public class Circulo {
   static int numCirculos = 0;
   public double x,y,r ;  //  centro y radio
   public Circulo(double x, double y, double r){
     this.x=x; this.y = y; this.r=r;
     numCirculos++;
   }
   public Circulo(double r) {this(0.0,0.0,r);}
   public Circulo(Circulo c) {this(c.x,c.y,c.r);}
   public Circulo() {this(0.0,0.0,1.0);}
   public double perimetro() { return 2 *3.14159 * r;}
   public double area() { return 3.14159 * r * r;}
   public void imprimirCentro() {
     System.out.println("("+x+","+y+")");
   }
   DOS OPCIONES: MÉTODO DE INSTANCIA O DE CLASE
}
```


40

Métodos de clase: métodos estáticos (cont)

```
// Métodos de instancia
public Circulo mayor(Circulo c)
{
   if(c.r > this.r) return c;
   else return this;
}

Circulo uno = new Circulo(2.0);
Circulo dos = new Circulo(3.0);
Circulo tres = uno.mayor(dos);

// Método de clase
public static Circulo
mayor(Circulo a, Circulo b)
{
   if(a.r > b.r) return a;
   else return b;
}
```

```
public class Circulo {
   static int numCirculos = 0;
   public double x,y,r ;  //  centro y radio

   public Circulo(double x, double y, double r){
     this.x=x; this.y = y; this.r=r;
     numCirculos++;
   }
   public Circulo(double r) {this(0.0,0.0,r);}
   public Circulo(Circulo c) {this(c.x,c.y,c.r);}
   public Circulo() {this(0.0,0.0,1.0);}
   public double perimetro() { return 2 *3.14159 * r;}
   public double area() { return 3.14159 * r * r;}
   public void imprimirCentro() {
     System.out.println("("+x+","+y+")");
   }
   DOS OPCIONES: MÉTODO DE INSTANCIA O DE CLASE
}
```

```
Circulo uno = new Circulo (2.0);
Circulo dos = new Circulo (3.0);
Circulo tres = Circulo.mayor(uno,dos);
```

Facultad de Estudios Superiores Acatlán

42

Ejercicio:

- Crea la clase `Vector3D`

Vector3D
- x : int - y : int - z : int
+ Vector3D(cx:int,cy:int,cz:int) + escalar(a:Vector3D, e:int):Vector3D + sumar(a:Vector3D, b:Vector3D):Vector3D + sumar(a:Vector3D) + sumarE(a:Vector3D, b:Vector3D):Vector3D + mostrar(mensaje:String)

- El método mostrar
  - Recibe un mensaje y muestra el contenido del vector entre corchetes [x,y,z]
- En un Java `main class`
  - Multiplica [5,3,7] por 4 y muestra el resultado
  - Usa todos los métodos de suma y muestra el resultado de [5,3,7] + [2,9,4]

44

Método `toString`

- Permite mostrar el valor de los atributos de un objeto
  - Su información, concatenada en un `String`
- `System.out.println()` invoca al método `toString()` de un objeto
  - Regresa un `String`
- Se puede "sobre-escribir" y mandar el contenido deseado

Método `toString` (cont)

Java main class

```
//Usando el método estático
res = Vector3D.sumarE(vector1, vector2);
res.mostrar("[5,3,7]+[2,9,4] = ");

// Este metodo modifica el valor del objeto que llama
vector1.sumar(vector2);
vector1.mostrar("[5,3,7]+[2,9,4] = ");

System.out.println(vector1);
```

Sin método **`toString`** en la clase

```
[5,3,7]+[2,9,4] = [7,12,11]
[5,3,7]+[2,9,4] = [7,12,11]
ejercicios.Vector3D@659e0bfd
BUILD SUCCESSFUL (total time:
```

Java class

```
53  public void mostrar(String mensaje){
54      System.out.println(mensaje+"["+x+","+y+","+z+"]");
55  }
56
57  @Override
59  public String toString(){
60      return "["+x+","+y+","+z+"]";
    }
```

Con método **`toString`** en la clase

```
[5,3,7]+[2,9,4] = [7,12,11]
[5,3,7]+[2,9,4] = [7,12,11]
[7,12,11]
BUILD SUCCESSFUL (total time:
```

Arreglo de objetos en `main class`

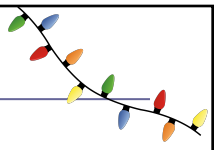
```
12  public class ArregloVector3D {
13      public static void main(String[] args) {
14          Vector3D [] arr1 = { new Vector3D(1,2,3), new Vector3D(4,5,6),
15                               new Vector3D(7,8,9), new Vector3D(0,1,2),
16                               new Vector3D(3,4,5)};
17          Vector3D [] arr2 = new Vector3D[5];
18          Vector3D [] arr3 = new Vector3D[5];
19
20          for(int i=0; i<5; i++)arr2[i]=new Vector3D(i,2*i,3*i);
21          System.out.println("arr1");
22          for(int i=0; i<arr1.length; i++){
23              System.out.println(arr1[i]);
24          }
25          System.out.println("arr2");
26          for(Vector3D v: arr2){
27              System.out.println(v);
28          }
29          for(int i=0; i<5; i++){  //SUMAS
30              arr3[i]=Vector3D.sumarE(arr1[i],arr2[i]);
31              arr1[i].sumar(arr2[i]);
32          }
33          System.out.println("Suma arreglos");
34          for(Vector3D v: arr3){
35              System.out.println(v);
36          }
37          System.out.println("Nuevo arr1");
38          for(Vector3D a: arr1){
39              System.out.println(a);
40          }
41      }
42  }
```

```
Output - LOO (run)
run:
arr1
[1,2,3]
[4,5,6]
[7,8,9]
[0,1,2]
[3,4,5]
arr2
[0,0,0]
[1,2,3]
[2,4,6]
[3,6,9]
[4,8,12]
Suma arreglos
[1,2,3]
[5,7,9]
[9,12,15]
[3,7,11]
[7,12,17]
Nuevo arr1
[1,2,3]
[5,7,9]
[9,12,15]
[3,7,11]
[7,12,17]
BUILD SUCCESS
```

50

50

Clase con arreglo de objetos



```
public class SerieNavidad {
    Foco[] serie;
    int cantidad;
    public SerieNavidad(int num, int w){
        cantidad=num;
        serie = new Foco[cantidad];
        for(int i=0; i<num; i++){
//            if(Math.random()>0.05){serie[i]=new Foco(w,false);
//            }else{serie[i]=new Foco();
//            }
            serie[i]=(Math.random()>0.05)?new Foco(w,false):new Foco();
        }
    }
    public void encender(){
        for(int i=0; i<cantidad; i++){
            if(!serie[i].fundido) serie[i].prender();
        }
    }
    public String toString(){
        String salida="";
        for(Foco f:serie)
           salida+=(f.estaPrendido())?"_1":"_0";
        return salida;
    }
}
```

```
public class PruebaSerieNavidad {
    public static void main(String[] args) {
        SerieNavidad s = new SerieNavidad(50,5);
        s.encender();
        System.out.println(s);
        s = new SerieNavidad(50,5);
        s.encender();
        System.out.println("La segunda\n"+s);
    }
}
```

```
run:
_1_1_1_1_1_1_1_1_0_1_1_1_1_1_1_1_1_1_1_1_1_1_1_1_1_1_1_1_1_1_1_1_1_1_1_1_1_1_1_1_1_1_1_1
La segunda
_1_1_1_1_1_1_1_1_1_1_1_1_1_1_1_1_1_1_1_1_1_1_0_0_1_1_1_1_0_0_1_1_1_1_1_1_1_1_1_1_1_1_1_1
BUILD SUCCESSFUL (total time: 0 seconds)
```

52

Java main class con la clase dentro

```
public class SerieNavidadP {
    Foco[] serie;
    int cantidad;

    public SerieNavidadP(int num, double w){
        cantidad=num;
        serie = new Foco[cantidad];
        for(int i=0; i<num; i++){
            serie[i]=(Math.random()>0.05)?new Foco(5,false):new Foco();
        }
    }
    public void encender(){
        for(int i=0; i<cantidad; i++){
            if(!serie[i].fundido) serie[i].prender();
        }
    }
    public String toString(){
        String salida="";
        for(Foco f:serie)
           salida+=(f.estaPrendido())?"_1":"_0";
        return salida;
    }
    public static void main(String[] args) {
        SerieNavidadP s = new SerieNavidadP(50,5);
        s.encender();
        System.out.println(s);
        s = new SerieNavidadP(50,5);
        s.encender();
        System.out.println("La segunda\n"+s);
    }
```

Si quieres usar tu clase sólo una vez

Si sólo quieres Probar tu clase

54