INICIACIÓN A LA PROGRAMACIÓN

LENGUAJE JAVA

Introducción

¿Qué es programar?

- *Idear y ordenar las acciones necesarias para realizar un proyecto (R.A.E)*
- En nuestro contexto:
  - Resolver problemas, Automatizar procesos,...

¿Cómo programaremos?

- Abstrayendo la realidad
  - definir los datos relevantes
- Diseñando una manipulación lógica y coherente de los datos

Introducción

¿Cómo programaremos un computador?

- Nuestro lenguaje y el del computador no se parecen en nada.
- Buscamos soluciones intermedias: lenguajes de programación
- Son lenguajes siempre más cercanos al lenguaje de la máquina que al nuestro.

Introducción

Paradigmas de la programación

- Cada lenguaje de programación está pensado para programar según un paradigma: forma de pensar y resolver problemas.
- Tipos de paradigmas de programación:
  - Estructurada: C, Pascal
  - Funcional: Lisp, Haskell
  - Lógica: Prolog
  - Orientada a objetos: Java, C++

Introducción

Elección de un lenguaje de programación

- La mayoría de lenguajes permiten programar con varios paradigmas, aunque estén especialmente ideados para uno de ellos.
- **JAVA**:
  - lenguaje orientado a objetos
  - desarrollado por Sun Microsystems a principios de los años 90.

Introducción

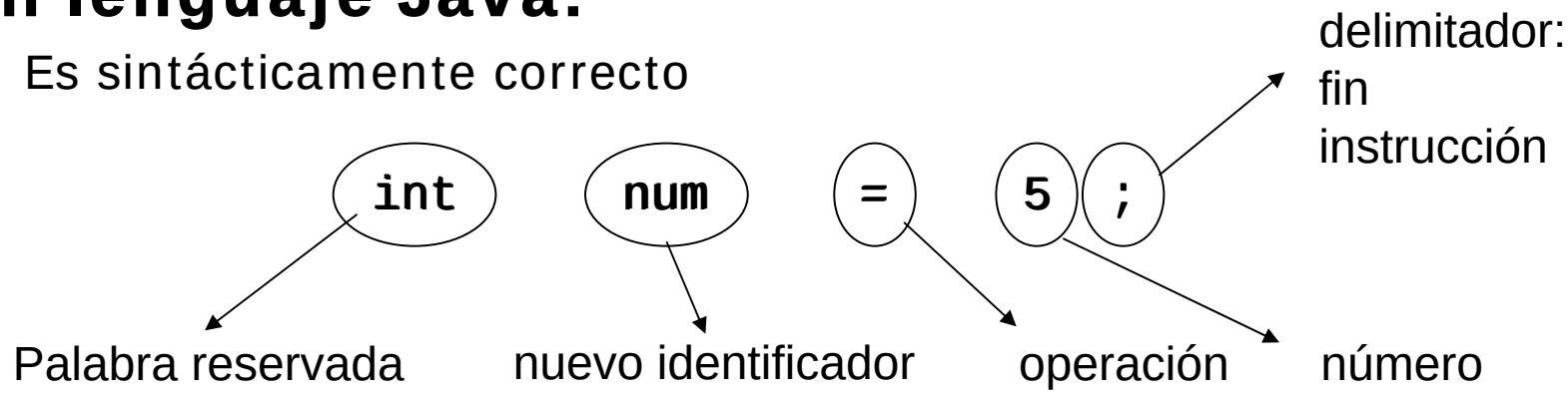
Lenguajes: sintaxis y semántica

- Los lenguajes son reglas que permiten a partir de unos símbolos iniciales...
  - crear nuevos símbolos correctos: **sintaxis**
  - Dotar de significado a los nuevos símbolos: **semántica**
- Símbolos iniciales:
  - palabras reservadas: palabras con un significado especial dentro del lenguaje (int, if, public, ...)
  - caracteres especiales: caracteres que identifican operaciones, delimitadores (*,-,{,}, etc).
  - Caracteres para formar nuevos identificadores, números.

Introducción Lenguajes

En lenguaje Java:

- Es sintácticamente correcto



- Su semántica: “define una variable para guardar números enteros con valor inicial 5”

- Pero:

`int n*2 = 5;` es sintácticamente incorrecto

Introducción

Elección del entorno de desarrollo de programas

- **BlueJ**
  - Entorno de desarrollo integrado para Java, indicado en contextos de educación y para proyectos pequeños de desarrollo de software.
  - Facilita el aprendizaje de la programación orientada a objetos.
  - Durante el desarrollo
    - Visualiza gráficamente la estructura de clases
    - Permite crear y probar los objetos de forma interactiva.
    - El interfaz de usuario es simple.

INICIACIÓN A LA PROGRAMACIÓN

LENGUAJE JAVA con BlueJ

Tema 3 *Clases y Objetos*

Tema 4 *Comunicación entre objetos. Algoritmos*

Tema 5 *Herencia y abstracción de datos*

Tema 6 *Diseño de clases*

TEMA 3 : Clases y Objetos

1. Introducción
2. Qué es un programa: clases y objetos
3. Componentes básicos de una clase
4. Campos
   - Tipos de datos primitivos
   - Identificadores
5. Constructores
6. Métodos
   - Sintaxis
   - Instrucciones básicas
   - Metodos especiales: *get* y *set*
7. Variables
8. Paquetes

TEMA 3 : Clases y Objetos

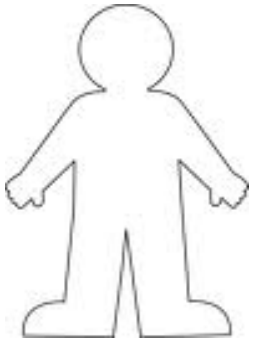
Introducción

Programación orientada a objetos:

- En este curso: programación orientada a objetos (POO) + Java
- Conceptos claves en POO: **clase y objeto**
- Una clase es una *plantilla* donde se definen las propiedades y comportamientos que tienen en común un conjunto de elementos
- Una clase es una abstracción de un concepto

TEMA 3 : Clases y Objetos

Introducción



clase Persona puede ser una clase con

- **Campos**(atributos): nombre, edad, dni,...
- **Métodos**(comportamientos): retorna su nombre, edad y dni, busca la persona de una lista con edad más parecida a la suya,...

CLASE

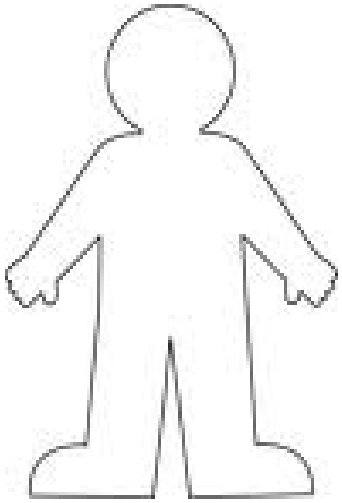
- Modelo abstracto para construir objetos
- Describe el estado mediante CAMPOS (o atributos)
- Describe comportamiento de los objetos mediante los MÉTODOS

Introducción

- OBJETO o Instancia
  - Un **objeto o instancia** es un *ejemplo* de un clase
  - Juan con 22 años y dni 3536353 es un *objeto* de la *clase Persona*

clase

Nombre
Edad
Dni



objeto

Juan
22
3536353

TEMA 3 : Clases y Objetos

Introducción

OBJETO o Instancia

- Los objetos se crean a partir de clases
- Se interactúa con los objetos invocando sus métodos
- Los objetos se caracterizan por datos denominados campos o atributos
- Los métodos y los campos son comunes a todos los objetos de la misma clase
- los valores almacenados en los campos puede ser diferente para cada objeto

TEMA 3 : Clases y Objetos

Introducción

Coche

Matricula Marca modelo, color,...

coche _1

coche _2

coche _3

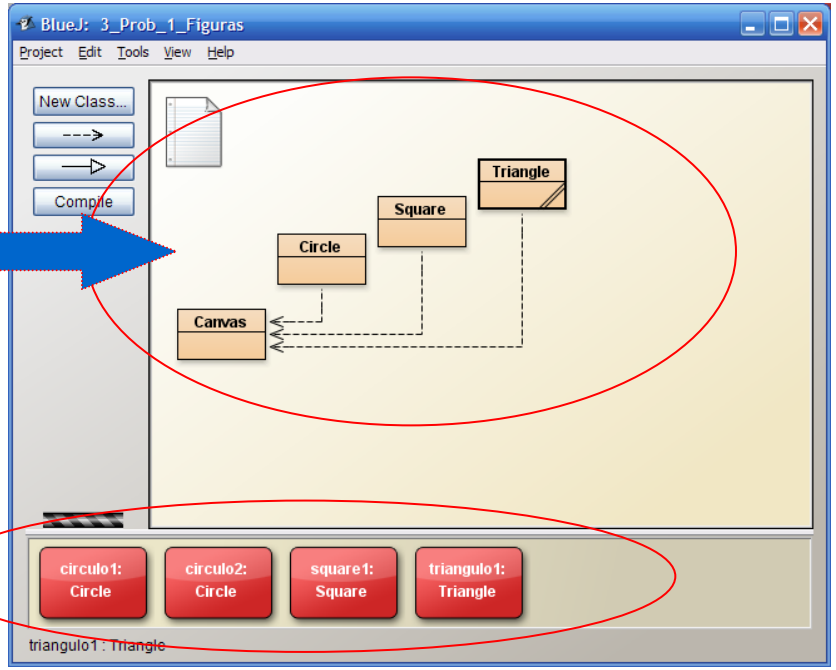
coche _4

Introducción

- En BlueJ se representan...

CLASES

OBJETOS



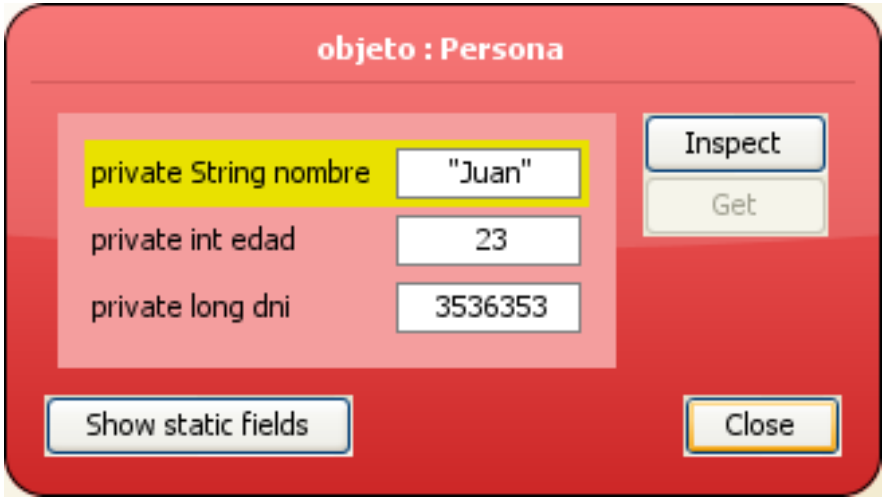
TEMA 3 : Clases y Objetos

Introducción

- El **estado** de un objeto es el valor que tienen su campos en un momento dado

objeto

Juan
23
3536353



TEMA 3 : Clases y Objetos

¿Qué es un programa?

Un **Programa** es...

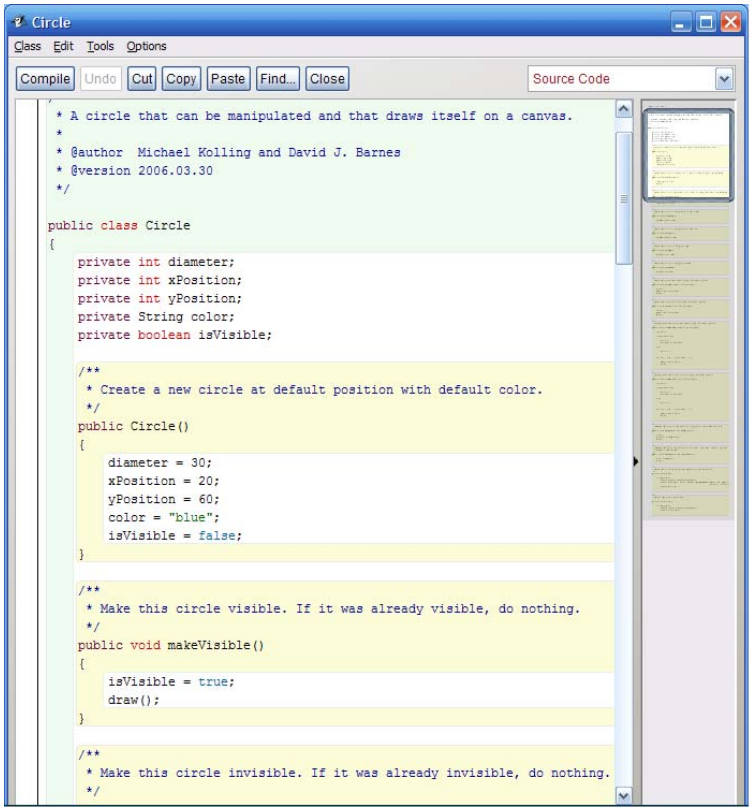
- Un conjunto de objetos de distintas clases enviándose mensajes y comunicándose entre sí.

TEMA 3 : Clases y Objetos

¿Qué es un programa?

Cualidades de un programa

- Eficacia
- Eficiencia
- Legibilidad
- Modularidad
- Reusabilidad
- Mantenible



Componentes básicos de una clase

- **Campos**: atributos de la clase.
  - Una persona es: nombre, edad, dni
  - Un punto del plano: coordenadas x e y
  - Un círculo: su punto central y radio

- **Constructores**: modo de construir objetos de las clases

- **Métodos**: comportamientos de las clases
  - Una persona conoce su nombre, edad y dni
  - A un punto se le puede sumar otro punto
  - Un círculo sabe calcular su área

Campos en java

```
/**
 * Clase para representar personas
 */
public class Persona {
    private String nombre;
    private int edad;
    private long dni;
    ..........
}
```

Declaración de clase

CAMPOS

Campos en java

```
/**
   Clase para representar puntos del plano
 */
public class Punto {
    private double x;
    private double y;
    ..........
}
```

Comentario: empieza con /** y finaliza en */

Campos en java

```
/**
 * Clase para representar círculos
 */
class Circulo {
    private Punto centro;
    private double radio;
    ..........
}
```

- También podemos declarar clases no públicas.
- En este curso, casi siempre serán públicas.
- Significado de `public`: en el tema siguiente.

Campos en java

Sintaxis: Declaración campo

tipo_visibilidad tipo_dato identificador;

tipo_visibilidad tipo_dato identificador = valor_inicial;

- Ejemplos:

```
private int edad = 3;
private double radio;
private String apellido = “Pérez”;
```

Campos en java

Visibilidad del campo

- Visibilidad de *casi todos* los campos será `private`. En el tema siguiente se explicarán los tipos de visibilidad.
- Hay cuatro tipos de visibilidad:
  - private `private int peso;`
  - public `public int peso;`
  - protected `protected int peso;`
  - sin nada `int peso;`

Tipos de datos

- Dos grupos de tipos de datos:
  - tipos de datos **primitivos**: para representar enteros, reales, booleanos y caracteres.
  - **clases**: hechas por el programador (Punto) o por otros programadores (String)

Tipos de datos

Tipos de datos primitivos

- Hay cuatro grupos de tipos de datos primitivos en Java
  - **Enteros**: para representar números enteros
  - **Reales**: representar números reales
  - **Booleanos**: valores de cierto o falso
  - **Caracteres**: representa un carácter
- Para cada tipo de dato primitivo nos preguntaremos
  - palabras reservadas: int, double, etc
  - rango de valores: ¿con un int puedo representar un número de la seguridad social?
  - literales: una número real se escribe 1,7 ó 1.7
  - operaciones: ¿puedo sumar enteros? ¿Y caracteres?

TEMA 3 : Clases y Objetos

Tipos de datos. Enteros

- Distintos tipos de enteros:

  byte, short, int, long

- Literales de tipo entero:
  - en base 10: 12 -345
  - en base 8: 023 -0678 (iniciados con un cero)
  - en base 16: 0x12 -0X23 (iniciados por cero y x)

- Ejemplos:

```
short num = 6;
long valor = -17;
```

Tipos de datos. Enteros

Rango de valores

Tipo dato	Tamaño	Valor mínimo	Valor máximo
byte	1 byte	-128	127
short	2 byte	-2^{15}	$2^{15} - 1$
int	4 bytes	-2^{31}	$2^{31} - 1$
long	8 bytes	$-2^{63} - 1$	$2^{63} - 1$

Tipos de datos. Enteros

Operaciones

Símbolo	resultado numérico
+	sumar
-	restar
*	multiplicar
/	dividir
%	módulo
+=	sumar igual
-=	restar igual
*=	multiplicar igual
/=	dividir igual
%=	módulo igual
++	sumar uno
--	restar uno

Símbolo	resultado boolean
> =	mayor o igual
>	mayor
<=	menor o igual
<	menor
==	igual
!=	distinto

Tipos de datos. Enteros

Operaciones

- Ejemplos:

```
int a = 5;
int b = a + 3;  // b es 8
b += 7; //equiv b = b + 7, b es 15
b = a % 3; // resto división a entre 3, b es 2
b++; // equiv b = b + 1, b es 3
++b; // sumar uno a b, b es 4
```

Otro tipo comentario: empieza con // y acaba al final de línea

Tipos de datos. Reales

- Distintos tipos de reales:
  `float` y **`double`**

- Literales de tipo real:
  - `1.78 -2.987`
  - Notación científica: `2.3E10 -1.4e-4`
  - `3.4f 7.3d 3d 0f`

- Ejemplos:

```
float radio = 6.7f;
double epsilon = 2.3e-4
```

Tipos de datos. Reales

Rango de valores

Tipo dato	Tamaño	Valor mínimo	Valor máximo
float	4 bytes	IEEE754	IEEE754
double	8 bytes	IEEE754	IEEE754

Tipos de datos. Reales

Operaciones

- Operaciones aritméticas binarias e unarias:

  + - * / % += -= *= /= %= ++ --

- Operaciones relacionales:

  >= > < <= == !=

- Las mismas que para enteros con el mismo significado
- Cuidado: al dividir números enteros, el resultado es entero pero si al menos un operando es real, es real

```
int a = 3;
int b = 2;
int cociente = a / b; //correcto, resultado entero
int cociente = a / 1.3; //error
double cociente = a / 1.3; //correcto, resultado real
```

TEMA 3 : Clases y Objetos

Tipos de datos. Operaciones de librería

- Para realizar operaciones matemáticas “complejas” se usa

```
double r1 = Math.sqrt(20.2); //raíz cuadrada
double r2 = Math.sin(r1);//seno
double r3 = Math.cos(r1*r2);//coseno
double r4 = Math.tan(67);//tangente
double r4 = Math.abs(r1);//valor absoluto
double r5 = Math.log(r4);//logaritmo en base e
double r6 = Math.log10(674);//logaritmo en base 10
```

- Y más que se pueden consultar en documentación

http://download.oracle.com/javase/6/docs/api/java/lang/Math.html

Tipos de datos. Booleanos

- Solo un tipo de booleano:

  `boolean`

- Literales de tipo booleano:
  - `true` `false`

- Ejemplos:

```
boolean esRedondo = true;
boolean esBlanco = false;
```

Tipos de datos. Booleanos

Operaciones

- Operaciones booleanos: operadores binarios *y, o* lógicos; operador unario *negación*.

&&	true	false
true	false	false
false	false	false

||	true	false
True	true	true
false	true	false

!	true	false
	false	true

Tipos de datos. Booleanos

Ejemplos

```
double a = 1.2;
int n = 5;
boolean esBlanco = false;
```

`(n > 5) && (a < 2)`	resultado false
`((a + 5) >= 3) || esBlanco`	resultado true
`((n * a) <= (n + a)) && !esBlanco`	resultado true

Tipos de datos. Carácter

- Solo un tipo de caracter:

  char

- Un carácter en Java ocupa dos bytes.
- A cada carácter se le asocia un número: código UNICODE
- Los 256 primeros caracteres del UNICODE se conocen también como código ASCII.
- UNICODE es una extensión de ASCII.

TEMA 3 : Clases y Objetos

Tipos de datos. Carácter (Códigos)

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Tipos de datos. Carácter

Literales de tipo **carácter**

- Un carácter entre comillas simples:

```
char car = 'A';
char oper =  '+';
```

- Código UNICODE entre comillas simples:

```
char car = '\u0041';
char oper =  '\u002B';
```

- El código UNICODE en Java empieza por **\u** y se compone de exactamente cuatro cifras en hexadecimal

Tipos de datos. Carácter

Operaciones con caracteres

- Operador incremento y decremento: Determinan respectivamente el siguiente y el anterior en la tabla de códigos

  ++ --

```
char car = 'A';
car++;
```

- Operadores relacionales: **>= > <= < == !=**

```
char car1 = '?';
char car2 = '9';
```

- ¿Es car1 menor o mayor que car2?
  - Cierto o falso según sea menor o mayor el código de un carácter respecto del otro.
  - En este caso, car1 es mayor que car2.

TEMA 3 : Clases y Objetos

Identificadores en Java

- Son ***nombres*** para las entidades que creamos en un programa tales como campos, clases, métodos, etc.

- Reglas para construir identificadores correctos:
  1. El primer carácter es una letra, **$** , **_**
  2. Los siguientes caracteres son *letras*, *dígitos*, **$**, **_**
  3. Un identificador no puede ser una palabra clave

- Estas reglas son de *obligado cumplimiento*
- Existen convenciones para generar identificadores que veremos a lo largo del curso

Componentes básicos de una clase

- **Campos**: atributos de la clase.
  - Una persona es: nombre, edad, dni
  - Un punto del plano: coordenadas x e y
  - Un círculo: su punto central y radio

- **Constructores**: modo de construir objetos de las clases

- **Métodos**: comportamientos de las clases
  - Una persona conoce su nombre, edad y dni
  - A un punto se le puede sumar otro punto
  - Un círculo sabe calcular su área

Constructores

- Los constructores se usan para crear los objetos de la clase

```
public class Persona {
    private String nombre;
    private int edad;
    private long dni;

    public Persona(String n, int e, long d) {
       nombre = n;
       edad = edad;
       dni = d;
    }
}
```

Constructores: Sintaxis

tipo_visibilidad NombreClase (lista_parametros)

donde

lista_parametros = tipo_dato identificador, tipo_dato identificador, …

- En la clase Persona

```
public Persona(String n, int e, long d){
    . . .
}
```

- En la clase Punto

```
public Punto(double nx, double ny) {
    . . .
}
```

Constructores: Código

- En los constructores se suele dar valores a los campos con los parámetros.

```
public class Punto {
   double x;
   double y;
   /**
    * Construye un punto dadas unas coordenadas.
    * @param nx coordenada x del punto.
    * @param ny coordenada y del punto.
    */
   public Punto(double nx, double ny) {
    x = nx;
    y = ny;
   }
    . . . . . .
}
```

Comentario con documentación de constructor

Constructores. Varios

- Una clase puede tener implementada tantos constructores como se quiera.

```
/**
  * Constructor por defecto que construye el punto (0,0)
  */
 public Punto() {
       x = 0;
       y = 0;
 }

 public Punto(double nx, double ny) {
    x = nx;
    y = ny;
   }
```

- El **constructor por defecto** es el que no tiene parámetros.

Constructores. Sin constructor

- Una clase puede no tener implementado ningún constructor.
- En ese caso, se genera automáticamente un constructor por defecto.
- El constructor por defecto que se crea da valores por defecto a todos los campos en función de su tipo

Tipo dato	**Valor por defecto**
entero	0
real	0.0
booleano	False
carácter	'\u0000'
clase	null

Constructores. Uso

Sintáxis:

new llamada_constructor

- Para construir un objeto se usa el operador ***new*** y un constructor.

```
Punto p1 = new Punto(2,3);//uso constructor dos parámetros
Punto p2 = new Punto(); //uso constructor por defecto
Persona estudiante = new Persona("Juan", 23, 38867876);
```

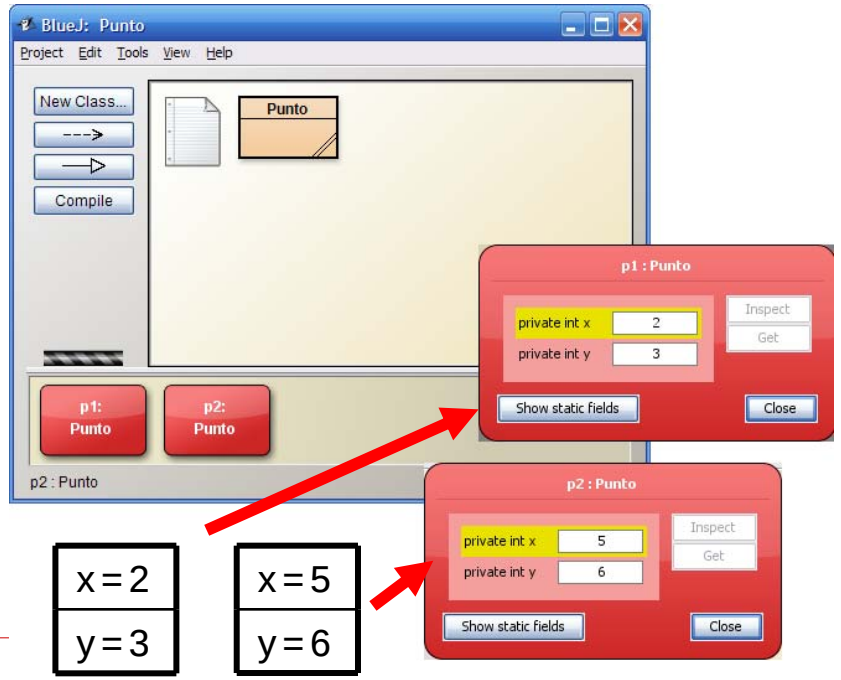
- p1 es una variable que hace referencia al objeto creado por `new Punto(2,3)`

Constructores. Llamadas

•La llamada a un constructor se resuelve dando valores a los parámetros por orden de aparición.

```
public Punto(double nx, double ny) {
    x = nx;
    y = ny;
}
```

```
Punto p1 = new Punto(2, 3);
Punto p2 = new Punto(5, 6);
```



•Se crean dos objetos de clase Punto cada uno con sus propios campos x, y

TEMA 3 : Clases y Objetos

Constructores. Llamadas

- Los parámetros se sustituyen ordenadamente y tienen que ser del mismo tipo que en la declaración.

```
public Persona(String n, int e, long d) {
    nombre = n;
    edad = e;
    dni = d
}
```

```
//error: primer parámetro no es de tipo String
//segundo parámetro no es de tipo int
Persona e = new Persona(23, “Juan”, 388833);
```

```
//correcto
Persona e = new Persona(“Juan”, 23, 388833);
```

Componentes básicos de una clase

- **Campos**: atributos de la clase.
  - Una persona es: nombre, edad, dni
  - Un punto del plano: coordenadas x e y
  - Un círculo: su punto central y radio

- **Constructores**: modo de construir objetos de las clases

- **Métodos**: comportamientos de las clases
  - Una persona conoce su nombre, edad y dni
  - A un punto se le puede sumar otro punto
  - Un círculo sabe calcular su área

Métodos

- Los métodos nos proporcionan el comportamiento de la clase

```
public class Punto {
    private double x;
    private double y;

    public Punto(double nx, double ny) {
            x = nx;
            y = ny;
    }

    public double getX() {
        return x;
    }

    public double getY() {
        return x;
    }
```

```
 public void setX(double nuevax) {
     x = nuevax;
 }

 public void setY(double nuevay) {
     y = nuevay;
 }

}//cierra clase Punto
```

Métodos. Sintaxis

tipo_visibilidad tipo_dato_retorno identificador(lista_parametros)

lista_parametros = tipo_dato identificador, tipo_dato identificador, …

```
public int getX(){
    . . .
}
```

Sin parámetros

se llama getX

retorna un int

```
public void setX(double nuevax) {
    . . .
}
```

Un método que no retorna nada, devuelve void

Métodos. Retornar datos

- Para retornar de un método se usa la palabra clave **`return`**

```
public double getX() {
    return x;
}

public void setX(double nuevax) {
    x = nuevax;
    return; //no es necesario, no se suele poner
}
```

El Tipo de dato x debe ser double

Métodos. Acceso a campos

```
public class Punto {
    private double x;
    private double y;

    public Punto(double nx, double ny) {
        x = nx;
        y = ny;
    }
    //crea un punto copia de p
    public Punto (Punto p)  {
        x = p.x;
        y = p.y;
    }
   . . . . .
}
```

Campos del objeto ya creado p

Campos del objeto que se está creando

- Para acceder al campo de un objeto

  nombre_objeto.nombre_campo

```
Punto p1 = new Punto(4, 5);
Punto p2 = new Punto(p1);
```

- El campo x de p1: p1.x
- El campo y de p2: p2.y
- CUIDADO!! Siguiente tema veremos dónde podemos escribir p1.y, p2.x

Métodos. Instrucciones básicas

Tipos de Sentencias:

- Asignación
  - Evalúa la expresión de la derecha
  - Almacenan el valor resultante en el campo, parámetro y variable de la izquierda

nombre_Campo = Expresión_a_evaluar;

nombre_Parámetro = Expresión_a_evaluar;

nombre_Variable = Expresión_a_evaluar;

```
edad = edad*2 + 4;
```

  - El resultado de la expresión debe ser del mismo tipo que el término de la izda

TEMA 3 : Clases y Objetos

Métodos. Instrucciones básicas

Sacar datos a la salida estándar (la pantalla)

- System.out.print(String) System.out.println(String)

System.out.print(String)
Muestra por pantalla el String

System.out.print(String)
Muestra por pantalla el String. La siguiente salida se mostrará en la siguiente línea.

- Se pueden concatenar Strings con operador +

  System.out.print(“Tienes “ + edad + “ años”);

Métodos. Instrucciones básicas

```
public class Punto {
  private  double x;
  private  double y;
 . . . .

  public void mostrar1() {
    System.out.print(“Valor x: “ + x);
    System.out.print(“Valor y: “ + y);
  }

  public void mostrar2() {
    System.out.println(“Valor x: “ + x);
    System.out.println(“Valor y: “ + y);
  }
  . . . .
}
```

Salida por pantalla con mostrar1

```
Valor x: 5Valor y: 4
```

Salida por pantalla con mostrar2

```
Valor x: 5
Valor y: 4
```

Métodos. Instrucciones básicas

Tipos de Sentencias:

- Condicional
  - Realiza una de dos acciones en función del resultado de una prueba

```
if (expresionBooleana)   {

   // acciones a realizar cuando es true
} else {

   // acciones a realizar cuando es false
}
```

  - Una expresión booleana tienen como resultado dos valores: cierto o falso.

TEMA 3 : Clases y Objetos

Métodos. Instrucciones básicas

```
public class Punto {
   private  double x, y;
 . . . .
public boolean igual(Punto p) {
  boolean esIgual = false;
  if (p.x == x && p.y == y) {
     esIgual = true;
  }
  return esIgual;
}

public void sumarORestar(int a) {
  if (a > 0) {
      x += a; y += b;
  else {
      x -= a; y -= b;
} }
```

```
/**
  * Retorna 1, -1, 0 si el punto
  * está a la arriba, abajo,
  * o en la recta x=abs
  */
public int comparaX(double abs) {
   int arriba = 0;
   if (x < abs) {
      arriba = -1;
   } else if (x > abs) {
      arriba = 1;
   }
   return arriba;
}
. . .
}
```

Métodos. Instrucciones básicas

Tipos de Sentencias:

- Condicional múltiple
  - Realiza acciones según valor de una variable de tipo entero o carácter.

```
switch (variable){
  case const11: case const12: …
       acciones1
       break;
  case const21: case const22: …
        acciones2
        break;
  . . . .
  default: //no es obligario el caso default
       accionesDefault
       break;
}
```

  - La constantes son literales del mismo tipo que la variable: enteras o caracteres.

Métodos. Instrucciones básicas

```
//a y b son variables
//enteras definidas
switch (a) {
case 1:
    b += 5;
    break;
case 2: case 3:
    b += 7;
    break;
default:
    b += 8;
    break;
}
```

```
//equivalente con if
if (a == 1) {
    b += 5;
else if (a == 2 || a == 3) {
    b += 7;
else {
    b += 8;
}
```

Métodos *especiales*

- Métodos ***get*** (u observadores): retornan el valor de un campo.
- Métodos ***set*** (o modificadores): modifican el valor de un campo.
- Método ***toString***: retorna la representación del objeto mediante un String.
- No todas las clases han de implementar estos métodos.
- Depende de cómo se diseñe la clase.

Métodos get (observadores)

- Los métodos que retornan el valor que tiene un campo se llaman métodos ***get*** u observadores.
- Los métodos *get* no tienen parámetros y retornar el tipo de dato del campo que observan.
- Ejemplo: si tenemos una clase con el campo *edad*

```
/**
 * Devuelve la edad.
 * @return edad de la persona.
 */
public int getEdad() {
    return edad;
}
```

TEMA 3 : Clases y Objetos

Métodos set (modificadores)

- Los métodos que modifican el valor que tiene un campo se llaman métodos ***set*** o modificadores.
- Los métodos s*et* tienen un parámetro del mismo tipo de dato que el campo que modifican y no retornan nada.
- Ejemplo: si tenemos una clase con el campo *edad*

```
/**
 * Modifica la edad.
 * @param nedad nueva edad de la persona.
 */
public void setEdad(int nedad) {
    edad = nedad;
}
```

Método toString

- El método *toString()* está definido implícitamente en todas las clases.
- Se utiliza básicamente para sacar por pantalla la representación del objeto.
- Ejemplo, si definimos una clase Punto y p es una variable de tipo Punto, es correcto el código

  `System.out.print(p);`

- Muestra por pantalla un número seguido nombre de clase: 123f@Punto
- Si introducimos en la clase punto el método

```
public String toString() {
     String res = “(“ + x + “,” + y + “)”;
     return res;
 }
```

  `System.out.print(p);`

- Muestra por pantalla: (3,4)

TEMA 3 : Clases y Objetos

Métodos. Variables locales.

- Las variables locales son variables que se declaran y usan dentro de un único método.

- El alcance de una variable es el código donde la variable puede ser accedida.
- El tiempo de vida describe el tiempo durante el cual la variable sigue existiendo antes de ser destruida

- El alcance y existencia de las variables locales se limitan al método donde se definen

- Las variables locales no tiene visibilidad. Solo los campos.

Métodos. Variables locales.

```
public class Punto {
    private double x;
    private double y;
    . . . . .
    public boolean estaEnRectaFormadaPor(Punto p1, Punto p2) {
       //hallar coeficientes a, b, c de la recta
       //formada por p1 y p2
       double a = -(p1.y – p2.y); //error private double a = …
       double b = p1.x – p2.x;
       double c = -a*p1.x – b*p1.y;
       boolean esta = (a*x + b*y + c == 0);
       return esta;
    }
    . . . .
}
```

Variables locales al método

Variables

Distinguiremos tres tipos de variables:

- campos (o variables de objeto)
- variables locales
- variable de clase (o estáticas, tema siguiente)

Variables: Declarar, inicializar, crear

- Declarar una variable es: indicar tipo y nombre de variable

```
int valor;   Punto p;
```

- Inicializar variable: darle un valor por primera vez

```
valor = 3;
```

- Las variables que son objetos de una clase: se crea el objeto.

```
p = new Punto(3, 4);
```

- Se puede declarar e inicializar al mismo tiempo.

```
int valor = 3;
Punto p = new Punto(3, 4);
Punto p1 = null;
```

- Buena costumbre del programador: declarar e inicializar al mismo tiempo las variables locales.
- Podemos inicializar a `null` (objeto nulo) cualquier variable tipo clase

Paquetes

- Todas las clases pertenecen a un *paquete*.
- Un paquete es un conjunto de clases bajo un nombre
- Ejemplos
  - **`java.lang`** contiene clases e interfaces fundamentales. Se importa automáticamente e implícitamente
  - **`java.util`** contiene clases e interfaces útiles y diversas
  - **`java.io`** contiene clases que permiten operaciones de entrada y salida
  - **`java.net`** contiene clases e interfaces que soportan aplicaciones para trabajar en red. (no las veremos)
  - ...

Paquetes

- Ejemplos:
  - la clase ***String*** pertenece al paquete java.lang
    - java.lang.String
  - la clase ***Math*** también pertenece al paquete java.lang
    - java.lang.Math
  - la clase ***Random*** pertenece al paquete java.util
    - java.util.Random
- Cuando queremos usar una clase de un paquete podemos incluir la sentencia:

```
import java.util.Random;
```

- También podemos importar cualquier clase de un paquete con

```
import java.util.*;
```

TEMA 3 : Clases y Objetos

Paquetes

```
import java.util.*;
public class Dado {
    int valor;
    Random gen = new Random();

     /**
     * Constuye un dado y le da un
     * valor inicial aleatorio
     */
    public Dado() {
       valor = gen.nextInt(6) + 1;
    }
```

```
   /**
    * Retorna valor del dado.
    * @return valor del dado.
    */
    public int getValor() {
       return valor;
    }

    /**
     * Lanza el dado de nuevo
     */
     public void tirarDado() {
        valor = gen.nextInt(6) + 1;
     }
} //fin clase Dado
```

Paquetes

```
import java.util.Random;
public class Dado {
    int valor;
    Random gen = new Random();

     /**
     * Constuye un dado y le da un
     * valor inicial aleatorio
     */
    public Dado() {
       valor = gen.nextInt(6) + 1;
    }
```

```
   /**
    * Retorna valor del dado.
    * @return valor del dado.
    */
    public int getValor() {
       return valor;
    }

    /**
     * Lanza el dado de nuevo
     */
     public void tirarDado() {
        valor = gen.nextInt(6) + 1;
     }
} //fin clase Dado
```

TEMA 3 : Clases y Objetos

Paquetes

```
//sin importar nada
public class Dado {
    int valor;
    java.util.Random gen =
         new java.util.Random();

     /**
     * Constuye un dado y le da un
     * valor inicial aleatorio
     */
    public Dado() {
       valor = gen.nextInt(6) + 1;
    }
```

```
   /**
    * Retorna valor del dado.
    * @return valor del dado.
    */
    public int getValor() {
       return valor;
    }

    /**
     * Lanza el dado de nuevo
     */
     public void tirarDado() {
        valor = gen.nextInt(6) + 1;
     }
} //fin clase Dado
```

Paquetes

- La clases que creamos también pertenecen a un paquete
- Para indicar a qué paquete pertenece la primera instrucción debe ser

```
package nombre_paquete;
```

- El nombre del paquete son identificadores serparados por puntos

```
package utiles.aleatorio;
package miscosas;
```

- Si no indicamos paquete, se crea un paquete sin nombre en el directorio de trabajo.

Paquetes

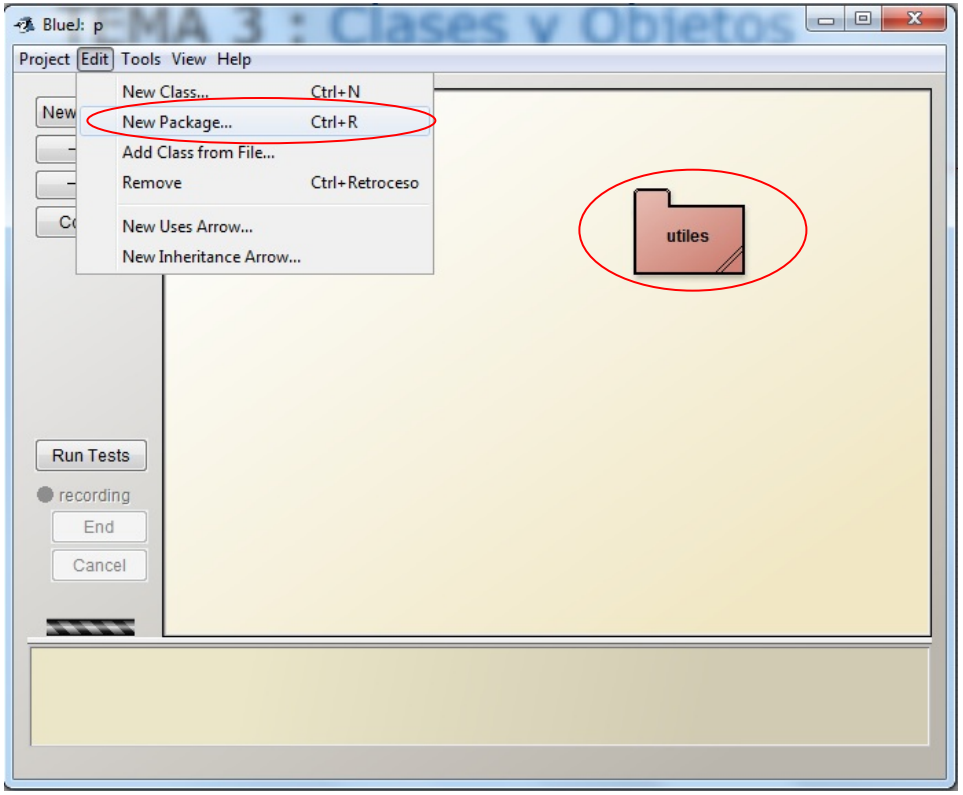
```
package utiles;

import java.util.*;

public class Dado {
    . . . . . .
}
```

Paquetes

- Crear paquetes con BlueJ

- Cualquier clase que se cree dentro de *utiles* automáticamente BlueJ introduce como primera línea de código

```
package utiles;
```



Paquetes

- Para crear el paquete *utiles.aleatorio* debemos crear dentro del paquete *utiles* el paquete *aleatorio*

- Si creamos una clase en *aleatorio*, automáticamente BlueJ introduce como primera línea de código

```
package utiles.aleatorio;
```