UT2 Objetos y clases

Módulo – Programación (1º)
Ciclos – Desarrollo de Aplicaciones Multiplataforma | Desarrollo de Aplicaciones Web
CI María Ana Sanz

Contenidos

- Objetos y clases
- Métodos
- Parámetros
- Tipos de datos
- Estado de un objeto
- Interacción de objetos
- Código fuente
- Valor de retorno de un método
- Objetos como parámetros

- Tipos de datos
  - tipos primitivos
  - tipos referencia
- Expresiones en Java

Objetos y clases

- Leer punto 2.1
- Clase
  - abstracción que describe a un tipo particular de objetos
  - Plantilla para crear objetos
  - describe un tipo de objeto
  - Persona / Estudiante / Profesor
  - Coche
  - Figura / Círculo

Objetos y clases

- Objeto
  - instancia individual de una clase
  - se crean a partir de las clases
  - representación detallada y particular de algo de la realidad
  - un objeto tiene estado, comportamiento e identidad
  - la estructura y el comportamiento de objetos similares están definidas en su clase común
  - *miCoche, tuCoche* – objetos de la clase Coche
  - *juan, pedro – objetos* de la clase Persona

Objetos y clases

Clase

Persona
características / estado /atributos
comportamiento

Objetos

Llamando a métodos

- A través de los métodos nos comunicamos con los objetos
- Los objetos hacen algo cuando invocamos a sus métodos
- Los métodos definen el comportamiento de un objeto
- Cuando invocamos un método sobre un objeto estamos enviando un mensaje al objeto.
- POO - conjunto de objetos que interactúan entre sí a través de mensajes.
- Algunos mensajes se los envía un objeto a sí mismo.
- Otros mensajes los envía un objeto a otro objeto
  - circulo1.moverAbajo()

Parámetros

- Los métodos pueden tener o devolver parámetros.
- Los parámetros proporcionan **información adicional** a los métodos para trabajar en función a los parámetros introducidos y/o devueltos.
- Cuando un método necesita un parámetro indica qué **tipo** de parámetro requiere.
- Podemos tener método **sin parámetros** - nombre del método y paréntesis vacíos.

Signaturas de métodos

- int, void, class – palabras reservadas del lenguaje Java
- Buenas prácticas a tener en cuenta:
  - nombres descriptivos (**legibilidad**).
  - nombres de **clases** empiezan en **mayúsculas.**
  - nombres de **métodos y variables** empiezan en **minúsculas**.
  - notación *camelCase*:
    - métodos y variables: tipoFlor, distancia, esVisible
    - clases: Estudiante, CentroComercial, CentroCivico

Ejemplo de signaturas de métodos

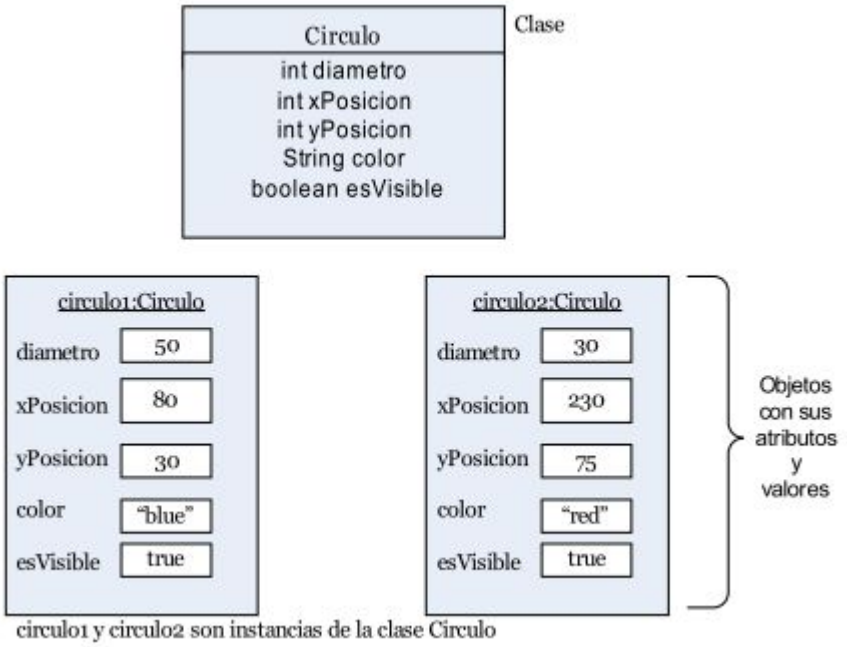
- void cambiarTamano(int nuevoAlto, int nuevoAncho)
- void hacerVisible()
- void enviarFlores(int cantidad, String tipoFlor, String aQuien)
- int calcularAreaCuadrada(int altura, int anchura)
- String saludar(String nombre)
- void guardarDinero(int dinero)
- int obtenerSaldo()

Estado de un objeto o instancia

- Todo objeto o instancia tiene un estado.
- El estado de un objeto o instancia está definido por los valores de sus atributos (campos).
  - *diámetro, xPosicion, yPosicion, color, esVisible*
- Los atributos también tienen un tipo
- Algunos métodos
  - modifican el estado de un objeto o instancia (mutadores)
    - void moverIzquierda() - modifica el atributo *xPosicion*
  - otros consultan su estado (accesores)
    - boolean esVisible() - consulta el estado del atributo *esVisible*

Estado de un objeto o instancia

- Objetos o instancias de la misma clase tienen todos los mismos atributos (¡Pero no los mismos valores!)
  - El nombre, tipo y nº de los campos es el mismo mientras que el valor actual de cada atributo en cada objeto puede variar.



Estado de un objeto / Comportamiento de un objeto

- Cuando se crea un objeto o instancia de una clase automáticamente se inicializan los atributos definidos de esa clase.
  - Los valores de esos atributos se almacenan en la instancia, no en la clase.
- Los métodos se definen también en la clase.
  - Todos los objetos o instancias de una misma clase tienen los mismos métodos (el mismo comportamiento).

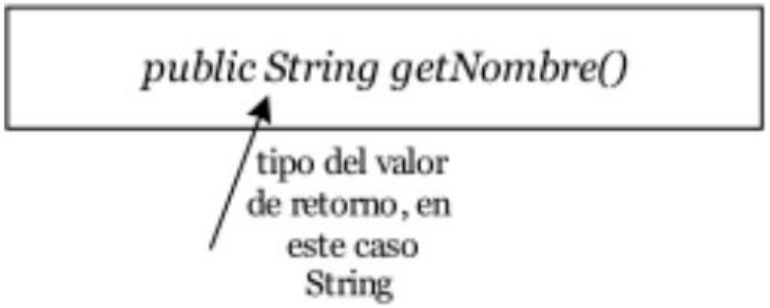
Los métodos se invocan sobre los objetos o instancias, no sobre las clases.

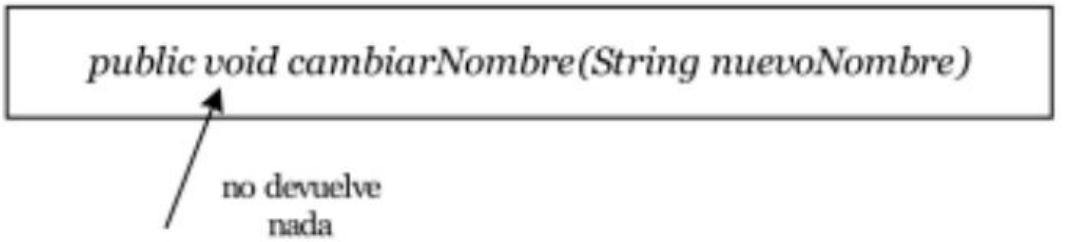
Interacción de objetos

- Los objetos o instancias pueden crear otros objetos
- Los objetos o instancias interactúan entre ellos a través de llamadas a métodos – ***paso de mensajes***
- Un programa es un conjunto de objetos o instancias
  - El usuario de un programa inicia el programa (normalmente creando un primer objeto) y todos los demás objetos se crean (directa o indirectamente) a partir de éste.

Valores de retorno

- Un método puede devolver un valor (valor de retorno o parámetro de salida), es decir, un resultado.





- public void moverHorizontal(int distancia)

  public String getNombre()

  public boolean estaVisible()

- Los métodos que devuelven valores nos permiten consultar el estado de un objeto.

Objetos como parámetros

- Los objetos o instancias pueden ser pasados como parámetros a métodos de otros objetos.
  - void matricularEstudiante(Estudiante nuevoEstudiante)

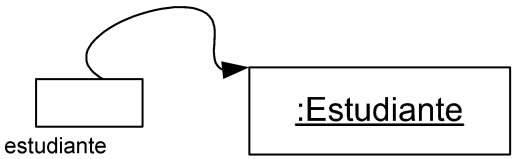
Tipos de datos

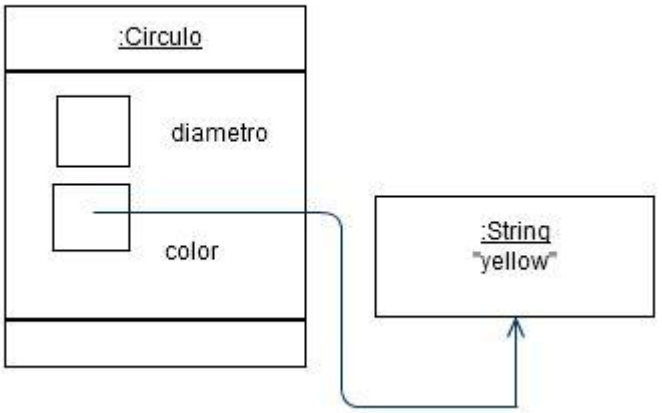
- **Tipo de datos**
  - conjunto de valores que una variable (atributo, parámetro, variable local) puede tomar
  - así sabe el compilador la cantidad de memoria que un atributo ocupará y la JVM reservará memoria para él

- Tipos de datos en Java
  - **primitivos** – simples (los que veremos de momento)
  - **referencia** – (apuntan a objetos o instancias)

Tipos de datos

- Tipos de datos en Java
  - *primitivos* – El atributo o parámetro (la variable, en general) guarda directamente el valor de ese tipo.
  - *referencia* - almacenan una referencia (puntero) o dirección de memoria a un objeto.





Tipos primitivos en Java – Tipos enteros

Nombre del tipo	Precisión	Rango	Ejemplos
Tipos numéricos enteros			
byte	8 bits	-128 a 127	24 -2 123
short	16 bits	-32768 a 32767	1234 -23456
int	32 bits	-2^{31} a $2^{31}-1$	-2003 5409
long	64 bits	-2^{63} a $2^{63}-1$	4233781L 55L

byte – 8 bits (un octeto)

01111111 +127

10000000 -128

complemento a 2

-2^7 a $2^7 - 1$ = -128 a 128-1 (127)

short – 16 bits (2 octetos)

int – 32 bits (4 octetos)

long – 64 bits (8 octetos)

Precisión – el nº de bits que utiliza la máquina virtual de Java para guardar el valor

Rango – conjunto de valores que comprende el tipo (valores que se pueden almacenar)

Tipos primitivos en Java – Tipos reales

Nombre del tipo	Precisión	Rango	Ejemplos
Tipos numéricos reales			
float	32 bits	-3.4E38 a 3.4E38	43.889F
double	64 bits	-1.7E308 a 1.7E308	45.63 2.4e5 45.64

float – 32 bits (4 octetos) – 43.89F

double – 64 bits (8 octetos) – 43.89

$24.5\ e^{-3} == 24.5 \times 10^{-3} == 24.5 / 10^{3}$

$43.89 == 4389 \times 10^{-2} == 4389 / 10^{2}$

notación exponencial

Tipos primitivos en Java – Tipos char y boolean

Nombre del tipo	Precisión	Rango	Ejemplos
Otros tipos			
char	16 bits	Un carácter Unicode (UTF-16)	'm' '?' '\u00F6'
boolean	Java no lo especifica	Valor booleano *true* o *false*	true false

- Ejemplos
  - enteros – *edad, nota, numeroPuertas*
  - reales – *peso, media, iva*
  - char – *estadoCivil* ('s' 'c')
  - boolean – *estaPrestado, estaEncendida*

Tipos primitivos en Java – Tipos char y boolean

- Ejemplos
  - char – un carácter Unicode
    - private char letra;
    - letra = 'A'; letra = '\u0041'; letra = 65;
  - secuencias de escape - '\n' '\\' '\t' '\"' (entre otros)

Carácter	Significado
\n	Salto de línea
\t	Tabulador
\"	Comillas dobles
\'	Comillas simples
\\	Barra invertida

Operadores y expresiones Java.

- Sobre cada uno de los tipos anteriores se pueden utilizar un conjunto de operadores para formar expresiones
- Una **expresión**
  - se construye agrupando operadores y operandos
  - se evalúa y produce un resultado de un determinado tipo.
- Expresiones
  - **aritméticas** – se construyen con operadores aritméticos
  - **lógicas** - se construyen con operadores lógicos y/o relacionales

Expresiones aritméticas

- Se construyen con **Operadores aritméticos: + - * / %**
- Los operandos son de un tipo primitivo numérico, entero o real
- Al evaluarlas producen un resultado numérico
- **/ %** resultado depende de los operandos, enteros o reales
- *Reglas de precedencia* de los operadores
  - se aplican cuando hay varios en una expresión
  - ( ) para cambiar la prioridad
  - a igual prioridad de izquierda a derecha
- Construiremos expresiones correctas en cuanto al estilo
  - blancos de separación entre operadores y operandos
  - ( ) para aclarar la expresión

Precedencia de los operadores

Operadores	
( )	Paréntesis
++ --	Incremento / Decremento
+ - !	Suma / Resta (Unario)
* / %	Producto / División / Resto
+ -	Suma / Resta
< <= > >=	Comparación
== !=	Igual / Distinto
&&	boolean (y /and)
||	boolean (or / o)
= += -= *= /= %=	Operadores de asignación

Expresiones aritméticas. Ejemplos

Ejemplos	51 * 3 - 53 ⇒ 100 154 - 2 * 27 ⇒ 100 (200 - 5) / 2 ⇒ 97 2 * (47 + 3) ⇒ 100

Ejemplos	5 + 3 ⇒ 8 5 / 3 ⇒ 1 5.0 / 3 ⇒ 1.66666	5 / 2 ⇒ 2 7 % 3 ⇒ 1

Expresiones booleanas

- Se construyen con operadores relacionales y/o lógicos
- Al evaluarlas producen un resultado lógico (booleano), *true* (cierto) o *false* (falso)
- Operadores relacionales usualmente se combinan con operandos y operadores aritméticos

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

- **Operadores lógicos** `&&` `||` `!`
  - se evalúan según las tablas de verdad

Tablas de verdad

a	b	a && b	a || b	!a
false	*false*	*false*	*false*	*true*
false	*true*	*false*	*true*	*true*
true	*false*	*false*	*true*	*false*
true	*true*	*true*	*true*	*false*

a,b: expresiones booleanas

- Ejemplos

Expresión	Resultado al evaluarla
4 > 5	false
4 != 5	true
(4 != 5) && (4 > 2)	true
(4 != 5) && (4 < 2)	false

Expresiones booleanas

- Evaluación en cortocircuito
  - Tan pronto como se conoce el resultado final de la expresión no se evalúan el resto de condiciones.

edad >= 65 && sexo == 'M'	si *edad* no es mayor o igual a 65 la condición es *false* y el resultado final de la expresión será también *false*, no se evalúa la segunda condición
notaTeoria >= 5 || notaPractica > 7	si *notaTeoria* es mayor o igual a 5 la condición es *true* y el resultado final de la expresión será también *true*, no se evalúa la segunda condición