UNIVERSIDAD TECNICA
FEDERICO SANTA MARIA

Estructuras fundamentales de la programación en Java

ELO-329: Diseño y programación orientados a objetos

Agustín J. González

Primer programa en Java

- Todo programa debe tener al menos una clase.
- Toda aplicación Java debe tener el método main como el mostrado.
- System.out es un objeto al cual le invocamos el método println.

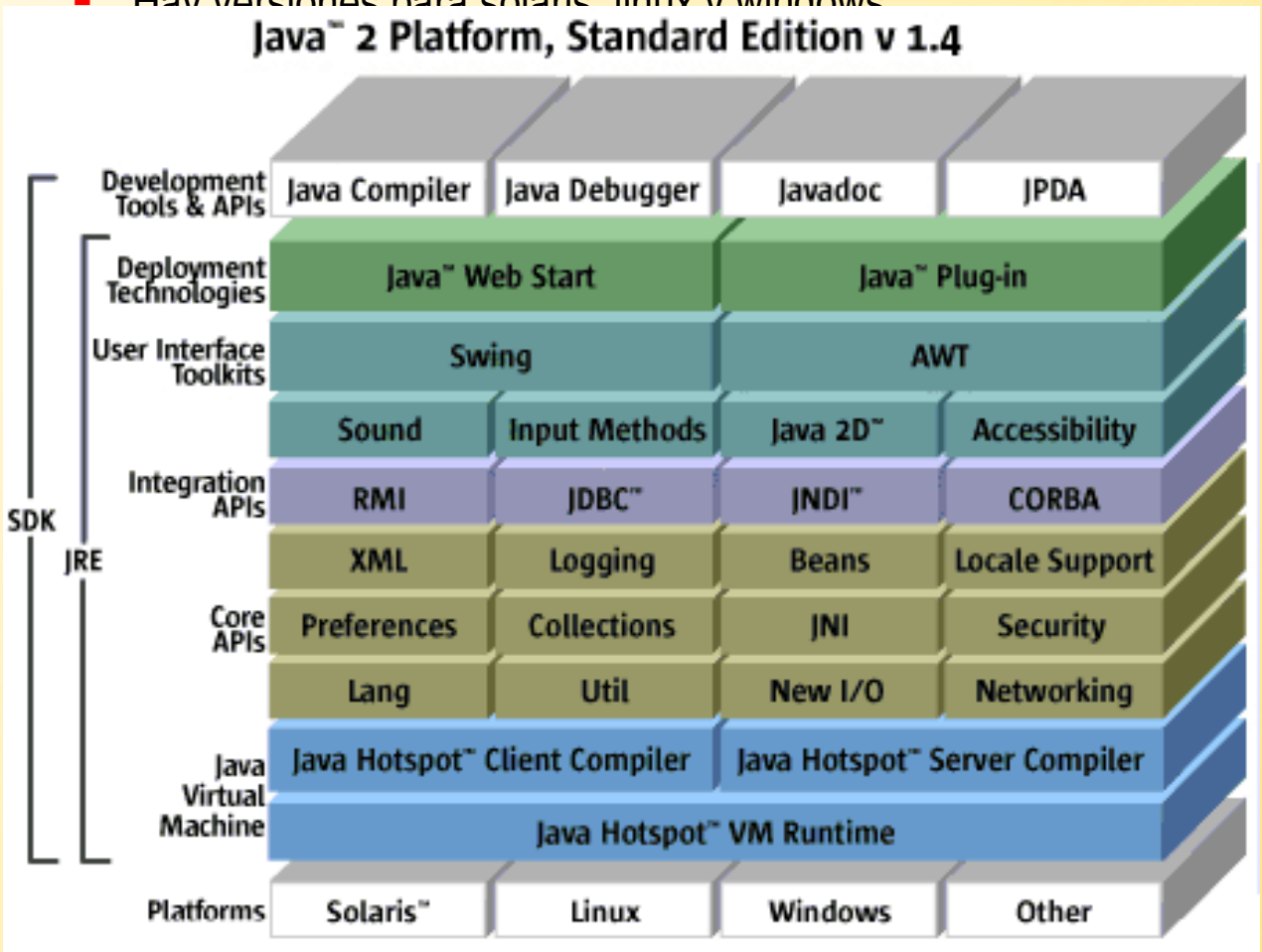
Nombre de archivo = FirstSample.java

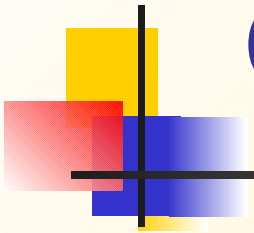
```
public class FirstSample
{
 public static void main(String[ ] args)
 {
  System.out.println("We will not use 'Hello, Sansanos!'");
 }
}
```

- FirstSample.java

Trabajando con Java: Instalación

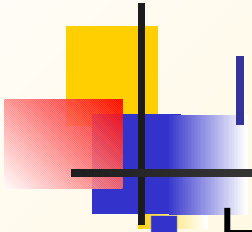
- Desde el CD de paño, o desde la red: http://www.elo.utfsm.cl/~install/index.php?dir=WINDOWS/Lenguajes/java/jdk_1.5/
- Desde http://java.sun.com/j2se/1.5
  - Hay versiones para solaris, linux y windows





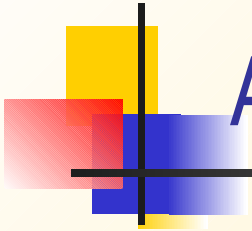
Compilador Java

- Ver CD en pañol con las versiones de Java para Windows y Linux. (estará actualizado próximamente)
- El CD también incluye ambientes de desarrollo.
- Manual en línea en manuales.elo.utfsm.cl



Instalación

- Hay otras versiones: Enterprice Edition (J2EE) y la Micro Edition (J2ME).
- Instalación en UNIX:
  - Incorporar el el path del compilador en el entorno al final de .bashrc o .bashrc_profile.
  - export PATH=/usr/local/jdk/bin:$PATH
- En Windows hacer lo equivalente (depende de su OS)
- Control Panel -> System -> Environment. Avanzar hasta las variables de usuario y buscar la variable PATH. Agregar el directorio jdk\bin al comienzo. Ej c:\jdk\bin; otras rutas.



Ambientes de desarrollo

- Hay varios. Lo más básico es usar un editor de texto, escribir los programas, compilar y ejecutar en la línea de comandos. En esta opción yo uso emacs o xemacs como editor.
- Jgrasp: http://www.jgrasp.org/ Ambiente desarrollado en Java para desarrollo de programas.
- Otros: kate en linux, netbean de Sun.
- Jedit: http://www.jedit.org/ También escrito en Java.
- Jbuilder : http://www.borland.com/jbuilder/

Tipos primitivos (no son objetos)

- Booleano
  - boolean
    - true and false
- Enteros
  - int 4 bytes Ej: 24, 0xFA, 015
  - short 2 bytes
  - long 8 bytes Ej: 400L
  - byte 1 byte
- Punto flotante
  - float 4 bytes Ej: 3.14F (6-7 dígitos signif.)
  - double 8 bytes Ej: 3.14D (15 dígitos signif.)

Tipos primitivos (no son objetos)

- Carácter: `char`
  - Unicode
    - Usa *dos* bytes
    - Diseñado para internacionalización
    - Comillas simples: ‘a’, ‘A’, ‘!’, ‘1’, ...
    - Forma hexadecimal ‘\u0008’ (Unicode backspace)
    - El byte menos significativo corresponde al ”ASCII” de 8 bits.
    - No visibles : Ej:
      - ‘\b’ backspace ‘\t’ tab
      - ‘\n’ linefeed ‘\r’ return
      - ‘\”’ double quote ‘\’’ single quote
      - ‘\\’ el mismo backslash!

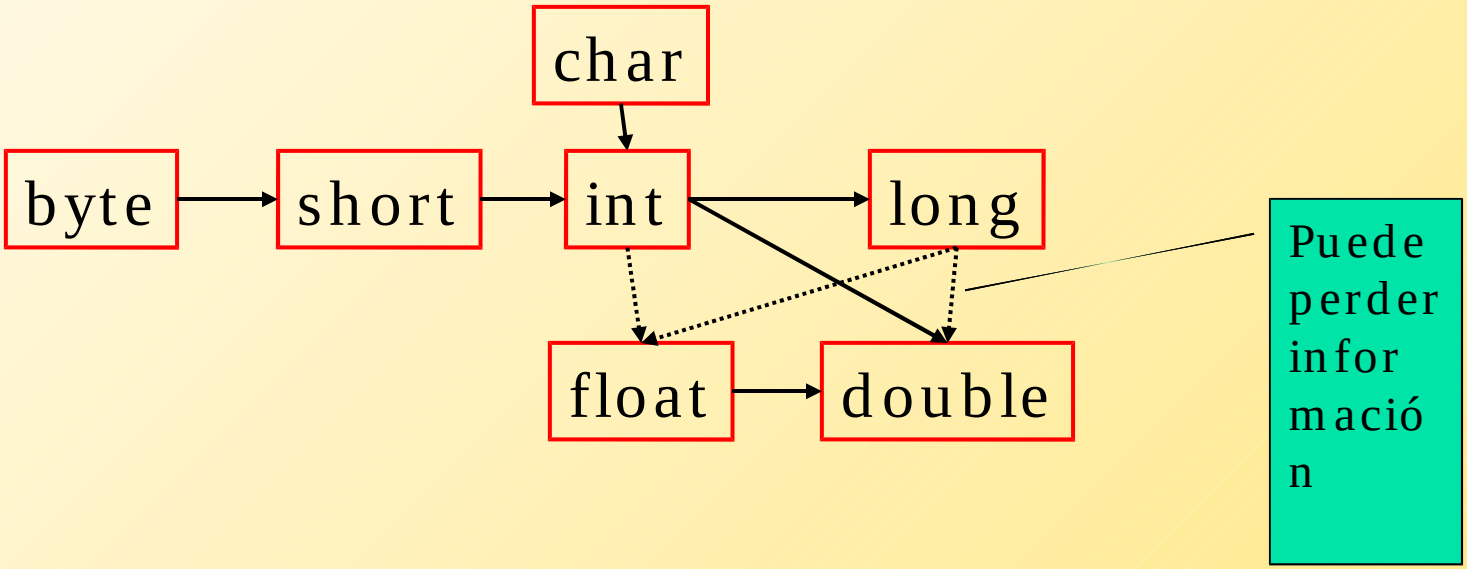
Constantes

- Se usa la palabra reservada *final*
- Ej: public final float CM_PER_INCH=2.54;
- Si deseamos crear sólo una instancia de esta constante para todos los objetos de una clase, usamos:

```
public class Constante
{
public static final float MC_PER_INCH=2.54;
...}
```

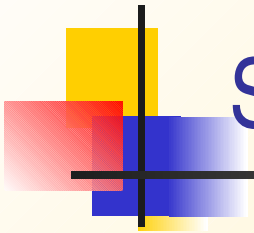
  - El valor se accede: Constante.CM_PER_INCH

Cambios de tipo automáticos



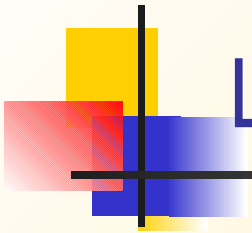
Operadores y su precedencia

[] . ( ) *(invocación)*	→
! ~ ++ -- +*(unario)* - *(unario)* ( ) *(cast)* new	←
* / %	→
+ -	→
<< >> >>>	→
< <= > >= instance of	→
== !=	→
&	→
^	→
|	→
&&	→
||	→
? :	←
= += -= *= /= %= &= |= ^= <<= >>= >>>=	←



String

- Java tiene una clase pre-definida llamada String.
- Todos los string son objetos y su comportamiento está dado por la clase (ver documentación).
- El operador + concatena strings. Si uno de los operandos no es string, Java lo convierte string y luego lo concatena.
  Ej: int nCanal=13;
  String estacion = “Canal”+nCanal;
- Para comparar dos strings, usar el método equals.



Lectura de entrada

- Es más complicado que en C, porque el modelado de la estrada de datos en más completo.
- Una forma simple de ingresar datos es vía JoptionPane.showInputDialog(promptString); este llamado retorna el string ingresado por el usuario.
- Ver ejemplo: InputTest

Sentencias

- IF
- ```
  if( exp ) statement1;
  else statement2;
  ```

- ```
  if (a>b) x = a;
     else      x = b;
  ```

  ```
  else // es opcional
     if ( x[i] > max ) max = x[i];
  ```

Sentencias - Lazos

- while

```
while( exp ) statement1;
while( exp ) { statements; }
```

-
```
while (a>b) a = x[i++];
while ( x < 0 ) {
      x = getX( ... );
      y = y + x;
      }
```

- `while` permite evitar el viaje al bloque interno

Sentencias - Lazos

- do

```
do statement; while( exp );
do { statements; } while( exp );
```

-
```
do a = x[i++]; while( a>z );
```

-
```
do {
        x = getX( ... );
        y = y + x;
        } while ( x > 0 );
```

- `do` implica al menos un viaje

Sentencias - Lazos

- for

```
for( exp1; exp2; exp3 ) { s; }
```

- *equivalente a:*

```
exp1;
while ( exp2 )
        { s; exp3; }
```

- `for( k=0; k<n; k++ ) { s; }`
  `equivale a:`

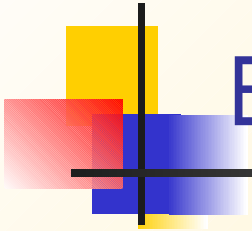
```
k=0;
while( k<n ) { s; k++; }
```

- *Patrón estándar para* `n` *iteraciones!*

Sentencias - switch

```
switch( exp1 ) {
    case x1: s1; break;
    case x2: s2; break;
    default: s3;
    }
```

```
switch( x ) {
  case 1: y = a; break;
  case 2: y = b; break;
  default: y = c;
  }
```

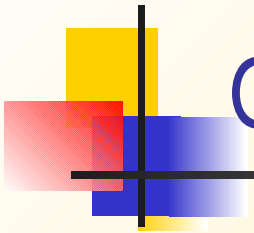


Break y continue

- La sentencia break permite salir fuera del lazo de repetición sin terminarlo (además de su uso en switch).
- También puede ser usada en conjunto con un rótulo para hacerla salir fuera de cualquier bloque. El rótulo va inmediatamente antes del bloque en cuestión.
- La sentencia continue transfiere el control de flujo a al encabezado del lazo más interno.

Clases para tipos de datos primitivos

- Envoltorios (Wrappers)
  - Crean objetos para los tipos estándares.
  - `java.lang`
    - `Boolean`
    - `Integer`
    - `Long`
    - `Character`
    - `Float`
    - `Double`
  - Un método importante en estas clases nos permite transformar un string que contiene números en un tipo básico. Ej: int ***a*** = Integer.parseInt("3425");
    hace que ***a*** adopte 3425.



Otros ejemplos

- InputTest.java
- LotteryOdds.java
- BigIntegerTest.java