Programación Orientada a Objetos

Dra. Maricela Claudia Bravo Contreras

mari_clau_18@hotmail.com

UNIVERSIDAD AUTONOMA METROPOLITANA
Casa abierta al tiempo Azcapotzalco

Elementos del Lenguaje Java

Justificación

- Java es el lenguaje de programación que más impacto ha tenido en los últimos años, especialmente en el mundo de desarrollo para la Web.
- La expansión de Java va en aumento no sólo en el desarrollo de aplicaciones Web, sino en el desarrollo de nuevas tecnologías como son: servicios Web y la programación para dispositivos electrónicos.

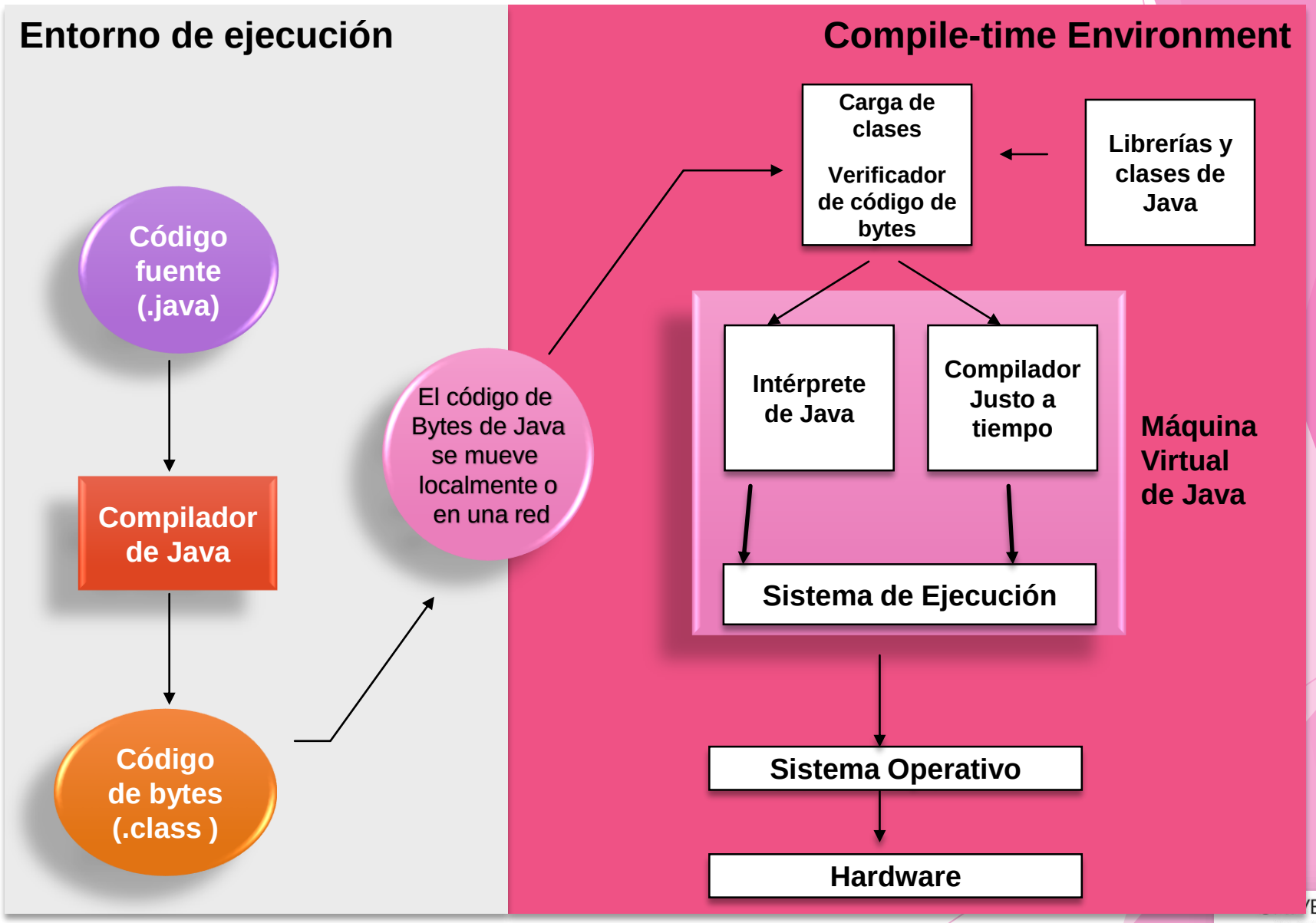
Características de Java

- Lenguaje totalmente orientado a objetos
  - Encapsulación, herencia, polimorfismo, etc.
- Dispone de un amplio conjunto de librerías
  - Oracle pone a disposición un amplio conjunto de clases para la creación de interfaces gráficas, gestión de redes, multitarea, acceso a datos, etc.

Características de Java

- Aplicaciones multiplataforma
  - Una vez que se ha compilado el programa, éste puede ser ejecutado en diferentes sistemas operativos sin necesidad de volver a compilar el programa.
- Ejecución segura de aplicaciones
  - El lenguaje carece de instrucciones que pueden provocar accesos descontrolados a la memoria (apuntadores en C++), la máquina virtual de Java impone ciertas restricciones a las aplicaciones para garantizar una ejecución segura.

¿Cómo funciona Java?



¿Cómo funciona?

- Java es independiente de plataforma:
  - Solamente depende de la Máquina Virtual de Java (JVM).
  - El código fuente se compila a *bytecode*, el cual es interpretado por la JVM residente en la máquina.
  - JIT (just in time) los compiladores intentan incrementar la velocidad de ejecución.

Ventajas de Java

- Portable
  - Escribe una vez y ejecuta en todas partes
- Seguridad
- Administración robusta de memoria
- Diseñado para la programación de redes
- Multihilos (múltiples tareas simultáneas)
- Dinámico y extensible
  - Clases almacenadas en archivos separados
  - Cargadas solamente cuando es necesario

Ediciones de Java

- Java 2 Estándar Edition (J2SE)
  - Paquetes para el tratamiento de cadenas, colecciones, acceso a datos, creación de entornos gráficos y applets.
- Java 2 Enterprise Edition (J2EE)
  - Paquetes y tecnologías necesarias para la creación de aplicaciones empresariales multicapa, por ejemplo aplicaciones Web.
- Java 2 Micro Edition (J2ME)
  - Paquetes para la creación de aplicaciones para dispositivos electrónicos como son: PDA´s, teléfonos móviles, agendas electrónicas, etc.

JDK - Java Development Kit

- javac – Compilador de java
- java - Interprete de java
- jdb - Debugger de java
- appletviewer –Herramienta para ejecutar applets
- javap – para imprimir los Java bytecodes
- javaprof - Java profiler
- javadoc – generador de documentación

Entornos de Desarrollo IDE

- NetBeansOracle *http://www.netbeans.org*
- Jbuilder Borland *http://www.borland.com*
- Jdeveloper Oracle *http://www.oracle.com*
- Eclipse Eclipse Foundation *http://www.eclipse.org*

Estructura de un Programa en Java

```
public class miClase
{
    public static void main(String[] args)
    {
        System.out.println(“Primera clase”);
    }
}
```

Tarea

- Cómo se configuran las variables de entorno
  - CLASSPATH
  - JAVAHOME
  - PATH
- En Windows y Linux, en línea de comandos, archivos por lotes y con herramientas del sistema operativo.

Elementos del Lenguaje de Programación Java

UNIVERSIDAD AUTONOMA METROPOLITANA
Casa abierta al tiempo Azcapotzalco

Tipos de Datos Primitivos

- Como tipos primitivos entendemos aquellos tipos de información más usuales y básicos. Son los habituales de otros lenguajes de programación.
- **boolean**: No es un valor numérico, solo admite los valores true o false.
- **char**: Cada carácter ocupa 16 bits.
- **Enteros**: Difieren en las precisiones y pueden ser positivos o negativos.
  - `byte: 1 byte.`
  - `short: 2 bytes.`
  - `int: 4 bytes.`
  - `long: 8 bytes.`
- **Reales en punto flotante**: igual que los enteros también difieren en las precisiones y pueden ser positivos o negativos.
  - `float: 4 bytes.`
  - `double: 8 bytes.`

Tipos de Datos Primitivos y Variables

- boolean, char, byte, short, int, long, float, double etc.
- Estos tipos de datos básicos o primitivos no son objetos.
- Esto significa que no se puede utilizar el operador new para crear una instancia de objeto para una variable.
- Declaración de variables con tipos primitivos:
  - `float valorInicial;`
  - `int valorRetorno, indice = 2;`
  - `double gamma = 1.2;`
  - `boolean valueOk = false;`

Variables

- Una variable es un espacio físico de memoria donde un programa puede almacenar un datos para su posterior utilización.
- Tipos de datos de una variable
  - Tipos primitivos
  - Tipo objeto

Inicialización de Variables

- Si no se asigna ningún valor previamente al uso de la variable, el compilador causará error.
- Java asigna variables primitivas a cero o falso en el caso de tipos boleanos.
- Todas las referencias a objetos apuntan a null inicialmente.
- Un arreglo es un objeto que se inicia en null al declararse, y sus elementos se inicializan a cero durante la creación.

Declaraciones

```
int a ; // declaración de una variable 'a' inicializada a 0

int b = 8; // declaración de una variable 'b' inicializada a 8.

NombreClase referencia; // declaración de una variable 'referencia' preparada
para un objeto de la clase 'NombreClase'.

NombreClase referencia2; // lo mismo que en la variable anterior.

Referencia = new NombreClase; // se crea un nuevo objeto de la clase
'NombreClase',y es asignado a la variable 'Referencia'

Referencia2 = referencia; // Ahora también 'referencia2' tiene el mismo objeto a
su cargo que 'referencia'
```

Asignaciones

- Todas las asignaciones en Java son de derecha a izquierda

  int a = 1, b = 2, c = 5

  a = b = c

  System.out.print(“a= “ + a + “b= “ + b + “c= “ + c)

- ¿Cuál es el valor de a, b y c?

- De derecha a izquierda: a = (b = c);

Operadores Aritméticos

`*` `/` `%` `+` – operadores básicos

`*` `/` `%` tienen mayor prioridad que `+` o `-`

```
double miVal = a + b % d - c * d / b;
```

Es lo mismo que:

```
double miVal = (a + (b % d)) - ((c * d) / b);
```

Operadores de Asignación

Expresión	Equivalente
a += b;	a = a + b;
a -= b;	a = a – b;
a *= b;	a = a * b;
a /= b;	a = a / b;
a %= b;	a = a % b;

Ejemplo

```
int a = 2, b = 3;
a += b;
b*=5;
a=++b;
b+=--a+5;
```

Operadores Relacionales

== Igual

!= Diferente

>= Mayor o igual que

<= Menor o igual que

> Mayor que

< Menor que

Ejercicio. Determinar si las siguientes relaciones son true o false

```
7 <= 5

2 > 6

3 != 5

a == a + 1

b ++ != b++
```

El Recolector de Basura

- Cuando se utilizan los métodos constructores para inicializar variables de instancia de objetos, éstos ocupan recursos del sistema, como por ejemplo memoria.
- Se necesita una forma disciplinada de devolver estos recursos al sistema cuando ya no son necesarios.
- Para evitar la fuga de recursos Java realiza automáticamente la recolección de basura.

El Recolector de Basura

- El recolector de basura ayuda a regresar al sistema la memoria ocupada por los objetos que ya no se utilizan.
- Cuando ya no existen referencias a un objeto, éste queda marcado para la recolección de basura.
- La memoria de ese objeto podrá reclamarse al momento en que se ejecute el recolector de basura.

Estructuras de Control

Una instrucción sencilla es un comando terminado por ;

```
nombre = “Fred”;
```

Un bloque de instrucciones es un conjunto de comandos encerrados por llaves

```
{
    nombre1 = “Fred”;
    nombre2 = “Bill”;
}
```

Los bloques de instrucciones pueden contener otros bloques de instrucciones.

Flujo de control

- Java ejecuta una instrucción después de la otra en el orden en que fueron escritos.
- Muchas instrucciones de Java son flujos de control

  Alternativas: if, if else, switch

  Ciclos: for, while, do while

  Escapes: break, continue, return

If – La instrucción condicional

- La instrucción condicional evalua una expresión y si el resultado de la evaluación es verdadero, entonces se ejecuta la acción

```
if ( x < 10 ) x = 10;
```

- Si el valor de x es menor que 10, hacer que x sea igual a 10
- También se puede escribir como

```
if ( x < 10 )
x = 10;
```

- O alternativamente:

```
if ( x < 10 ) { x = 10; }
```

If… else

- La instrucción condicional if … else evalúa una expresión y realiza una acción si la evaluación resulta verdadera, o realiza otra acción si la evaluación resulta falsa.

```
if (x != anterior)
{
    System.out.print("x fue cambiada");
}
else
{
    System.out.print("x no fue cambiada");
}
```

if … else anidado

```
if ( miVal > 100 )
{
    if ( remainderOn == true)
  {
        miVal = miVal % 100;
     }
    else
  {
      miVal = miVal / 100.0;
     }
 }
 else
 {
      System.out.print(“miVal está en rango”);
 }
```

else if

Útil para escoger entre alternativas:

```
if ( n == 1 )
{
 // ejecuta el código del bloque #1
}
else if ( n == 2 )
{
    // ejecuta el código del bloque #2
}
else
{
    // si todas las evaluaciones anteriores han
 fallado, ejecuta el código del bloque #3
}
```

La sentencia switch

```
switch ( n )
{
   case 1:
     // ejecuta el código del bloque #1
     break;
   case 2:
     // ejecuta el código del bloque #2
     break;
   default:
     // si todas las pruebas anteriores
    // fallan entonces
    // ejecuta el código del bloque #4
     break;
 }
```

El ciclo for

- Repetición n veces

```
for ( i = 0; i < n; n++ ) {
    // este código se ejecutará n veces
    // i desde 0 hasta n-1
}
```

- For anidado:

```
for ( j = 0; j < 10; j++ ) {
    for ( i = 0; i < 20; i++ ){
      // este código se ejecutará 200 veces
    }
}
```

Ciclo while

```
while(response == 1)
{
  System.out.print( "ID =" +
  userID[n]);
  n++;
  response = readInt( "Enter ");
}
```

Ciclo do {… } while

```
do
{
   System.out.print( “ID =” + userID[n] );
   n++;
   response = readInt( “Enter ” );
}while (response == 1);
```

Break

- La instrucción break causa un exit desde el ciclo que lo llama.

```
for ( int i = 0; i < maxID, i++ )
{
  if ( userID[i] == targetID )
  {
    index = i;
    break;
  }
} // la ejecución del programa salta
  aquí después del break
```

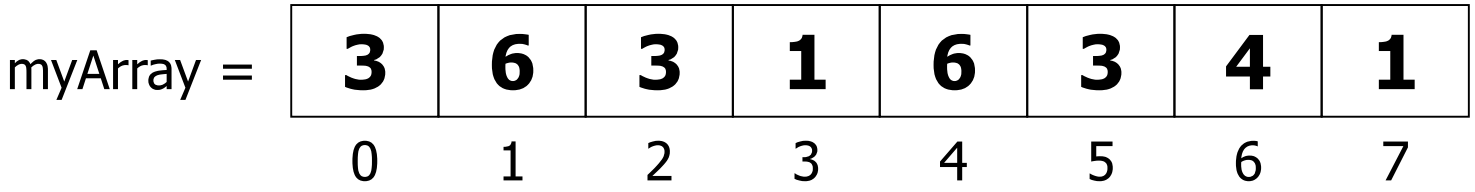
Continue

- Puede ser utilizado solamente con los ciclos while, do o for.
- La instrucción continue ocasiona que el ciclo continue en la siguiente iteración inmediatamente.

```
for ( int i = 0; i < maxID; i++ )
{
  if ( userID[i] != -1 ) continue;
  System.out.print( "UserID " + i + " :" +
   userID);
}
```

Arreglos

- Un arreglo es una lista de elementos similares
- Un arreglo tiene:
  - nombre
  - tipo
  - tamaño
- Estos deben ser declarados cuando el arreglo es creado.
- El tamaño del arreglo no puede ser cambiado durante la ejecución del programa.



myArray tiene espacio para 8 elementos

- Los elementos son accedidos por su índice
- en Java, los índices de los arreglos comienzan con 0.

Declaración de Arreglos

```
int miArray[];
```

declara *miArray* como un arreglo de enteros

```
miArray = new int[8];
```

reserva 8 espacios de enteros en memoria, etiquetados de *miArray[0]* - *miArray[7]*

```
int miArray[] = new int[8];
```

combina las dos instrucciones anteriores.

Asignación de valores

- Se hace referencia a los elementos del arreglo mediante su índice.

```
miArray[0] = 3;

miArray[1] = 6;

miArray[2] = 3;   ...
```

- Se puede inicializar en un solo paso.

```
int miArray[] = {3, 6, 3, 1, 6, 3, 4, 1};
```

Iteración con arreglos

- Los ciclos *for* loops son los más usuales para trabajar con arreglos:

```
for (int i = 0; i < myArray.length; i++)
{
  myArray[i] = getsomevalue();
}
```

Arreglos de objetos

- Hasta ahora solo se han visto arreglos de tipos de datos primitivos.
  - integers
  - doubles, floats, characters…
- Frecuentemente se necesitan arreglos de objetos
  - Estudiantes, Libros, Créditos ……
- Se necesitan seguir tres pasos.

Declaración del arreglo

1. Declarar el arreglo

```
private Student listaAlumnos[];
```

- Esto declara listaAlumnos

2 .Crear el arreglo

```
listaAlumnos = new Alumno[10];
```

- reserva 10 espacios en memoria que pueden almacenar referencias a los objetos de Alumno

3. Crear el objeto de estudiante y añadirlo al arreglo

```
listaAlumnos[0] = new Alumno("Cesar", "Computacion");
```

Dudas o comentarios???