UNIVERSIDAD POLITÉCNICA DE SAN LUIS POTOSÍ


# Programación III
# POO JAVA

## Introducción al lenguaje

MTI. Guadalupe Ledesma Ramos guadalupe.ledesma@upslp.edu.mx

# Lenguaje OAK (Roble)

- Concebido por James Gosling, Patrick Naughton, Chris Warth, Ed Frank y Mike Sheridan de Sun Mycrosystems, 1991.
- Orientado a aparatos electrodomésticos.
- Encontró su nicho en el desarrollo Web
- Inició con aplicaciones llamadas Applets
- Recibe el nombre de Java en 1995

# Entorno de desarrollo Java

- Diseñado para ser:
  - Simple:
    - Solo se manipulan objetos.
    - Optimizar el Código
  - Orientado a objetos:
    - Su principal objetivo es la creación de objetos, piezas de código que puedan interactuar con otros objetos para resolver problemas.

# Entorno de desarrollo Java

- Robusto: La robustez es la fiabilidad de un programa.
  - No tiene apuntadores o punteros. En Java hay referencias en lugar de apuntadores, y no se pueden mover una referencia para manipular directamente espacios de memoria.
  - Java tiene un colector de basura (garbage collector), el colector libera memoria automáticamente.
- Multithreaded (Multihilo)
  - Desempeña varios tareas a la vez, por ejemplo consultar a una base de datos y desplegar un formulario en pantalla.

# Entorno de desarrollo Java

- Arquitectura neutra. Independiente de la plataforma
  - Linker: liga un programa compilado con otras librerías para generar un ejecutable.
  - Compilador: convierte un programa a código llamado Código Máquina
- Interpretado:
  - Sus códigos de programas fuente en lugar de ser compilados y traducidos en ejecutables nativos, son traducidos en códigos de bytes (byte code) no asociados a una plataforma específica.

# Entorno de desarrollo Java

- Distribuido:
  - Puede acceder a objetos distribuidos en distintas computadoras.
    - Utiliza los protocolos estándar basados en TCP/IP como HTTP
    - RMI (que son invocaciones a métodos remotos).
- Seguro:
  - Prohíbe la manipulación de la memoria usando apuntadores.
  - Verifica que el código sea válido antes de ejecutarse.

# Compilación y Ejecución

- Fichero fuente Java: un fichero fuente contiene texto, escrito en el lenguaje de programación Java. Se puede usar cualquier editor de texto para crear y editar ficheros fuente.

- Fichero byte-code: el compilador de Java, javac, toma el fichero fuente y lo traduce en instrucciones que la Máquina Virtual Java (JVM) puede entender.

- Ejecutar fichero con bytecodes: la máquina virtual Java está implementada por un intérprete java. Este intérprete toma el fichero con byte-code y lo traduce a instrucciones que la máquina anfitriona pueda entender.

# Entorno de desarrollo Java

- Los programas en Java constan de varias piezas llamadas **clases**. Estas clases incluyen piezas llamadas **métodos**, los cuales realizan tareas y devuelven información cuando completan esas tareas.
- **Bibliotecas de clases de Java**, también se conocen como **APIs (Interfaces de programación de aplicaciones) de Java**.

- http://docs.oracle.com/javase/7/docs/api/

# Pilares de POO

- **Pilares,** aseguran la simplicidad de código y su reutilización.
  - **Abstracción**, permite identificar las características y comportamientos de un objeto y con los cuales se construirá la clase.
  - **Encapsulación**, implica el tratamiento de un grupo de propiedades (atributos o variables de instancia y métodos o acciones), como una única unidad u objeto, está relacionado con el acceso a un código desde el código de otra clase.
  - **Herencia,** describe la capacidad de crear clases nuevas a partir de una clase existente. La nueva clase hereda todas las propiedades, métodos y eventos de la clase base, y se puede personalizar con propiedades y métodos adicionales.
  - **Polimorfismo,** implica la posibilidad de tener varias clases que se pueden usar de forma intercambiable, incluso si cada clase implementa las mismas propiedades o métodos de formas distintas. El polimorfismo es esencial para la programación orientada a objetos, ya que permite usar elementos con los mismos nombres, sin importar qué tipo de objeto esté en uso en ese momento.

# Clases

- Son el núcleo de Java.
- La construcción lógica sobre la que se basa el lenguaje Java.
- Definen la forma y naturaleza de un objeto.
- Constituyen los fundamentos de la programación orientada a objetos en Java.
- El código de una clase define la interfaz con sus datos.
- Define un nuevo tipo de datos.

# Clases

- **Propiedades de una clase:**
  - Variables de instancia, se implementan mediante los procedimientos de las propiedades Get y Set, que proporcionan más control sobre el modo en que los valores se definen o se devuelven.
  - Métodos, representan acciones que puede realizar un objeto, definen el comportamiento de la clase.
  - Eventos, son notificaciones que un objeto recibe de otros objetos u otras aplicaciones o que transmite a ellos.

# Objetos

- Es una Instancia de una clase.
- Se crea mediante el operador new, este operador asigna memoria a un objeto en tiempo de ejecución y devuelve una referencia. Esta referencia se almacena en una variable.
- Un objeto tiene una realidad física. Es decir, ocupa espacio en memoria.
- Estructura con datos y métodos que manipulan los datos.

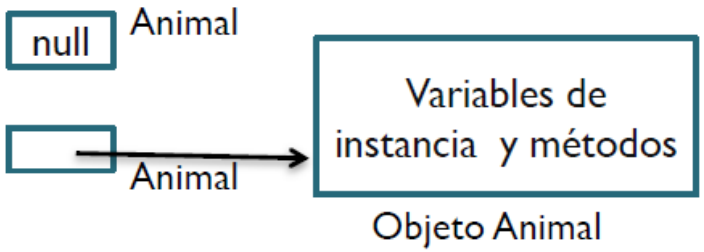
Miembros de una clase:

- Datos/Variables de instancia/atributos/características.
- Métodos o funciones miembro (acciones, definen el comportamiento de una clase).

```
class nombre de clase{
        tipo variable de instancia1;
        tipo variable de instancia2;
        //...
        tipo variable de instanciaN;
        tipo nombre del método1(lista de parámetros){
                //Cuerpo del método
        }
        tipo nombre del método2(lista de parámetros){
                //cuerpo del método
        }
}
```

# ¿Qué hace el siguiente fragmento de código?

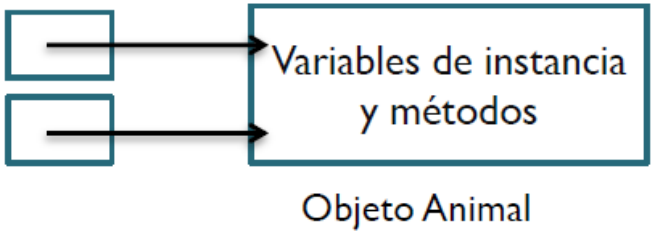
```
Animal b1;              //Declara la referencia a un objeto
b1= new Animal();       // reserva espacio para el objeto
```

| ***Sentencia*** | ***Efecto*** |
|---|---|
| Animal b1; |  |
| b1= new Animal(); | |



---

```
Animal b1 = new Animal();
Animal b2 = b1;
```





b1 y b2 hacen referencia al mismo objeto

b2 no reserva memoria

- Cada objeto de una clase (cada instancia de clase) contiene una copia de las variables de instancia.
- La variables de instancia son variables que contienen una referencia en memoria de un objeto.
- Los datos de un objeto están separados y son individuales de los datos de otro objeto.

De la clase Libro

- Libro milibro; //declara la referencia a un objeto de tipo Libro
- milibro = new Libro () // reserva espacio para el objeto

```
class MiClase {
        public void bienvenida() {
                System.out.println("Hola, Bienvenido a POO Java");
        }
 }
class Saludo{
        public static void main (String[] args) {
                MiClase hola = new MiClase();
                hola.bienvenida();
        }
 }
```

Recuerda que una clase define un nuevo tipo de datos. En el código MiClase y Saludo son dos nuevos tipos de datos.

# Llamada de métodos

- Forma general de un método:

```
tipo nombre_de_método(lista de parámetros){
    //cuerpo del método
}
```

Si el método no devuelve un valor, el tipo devuelto será void.

Los métodos que devuelven un tipo diferente al void devuelven el valor a la rutina que realiza la llamada mediante la sentencia return:

```
return valor;
```

# Tipos de datos en Java

| | NOMBRE | TIPO | OCUPA | RANGO APROXIMADO |
|---|---|---|---|---|
| **TIPOS PRIMITIVOS** (sin métodos; no son objetos; no necesitan una invocación para ser creados) | byte | Entero | 1 byte | -128 a 127 |
| | short | Entero | 2 bytes | -32768 a 32767 |
| | int | Entero | 4 bytes | $2^{*}10^{9}$ |
| | long | Entero | 8 bytes | Muy grande |
| | float | Decimal simple | 4 bytes | Muy grande |
| | double | Decimal doble | 8 bytes | Muy grande |
| | char | Carácter simple | 2 bytes | --- |
| | boolean | Valor true o false | 1 byte | --- |

# Tipos de datos en Java

| | | |
|---|---|---|
| **TIPOS OBJETO** (con métodos, necesitan una invocación para ser creados) | Tipos de la biblioteca estándar de Java | String (cadenas de texto)<br>Muchos otros (p.ej. Scanner, TreeSet, ArrayList…) |
| | Tipos definidos por el programador / usuario | Cualquiera que se nos ocurra, por ejemplo Taxi, Autobus, Tranvia |
| | arrays | Serie de elementos o formación tipo vector o matriz. Lo consideraremos un objeto especial que carece de métodos. |
| | Tipos envoltorio o wrapper (Equivalentes a los tipos primitivos pero como objetos.) | Byte |
| | | Short |
| | | Integer |
| | | Long |
| | | Float |
| | | Double |
| | | Character |
| | | Boolean |

# Operadores

| Operator | Description | Operator Type |
|---|---|---|
| ++,-- | Postfix increment, postfix decrement | Arithmetic |
| ++,-- | Prefix increment, prefix decrement | Arithmetic |
| ! | Boolean NOT | Logical |
| *,/,% | Multiplication, division, remainder (modulus) | Arithmetic |
| +,- | Addition, subtraction | Arithmetic |
| <, <=, >, >= | Less than, less than or equal to, greater than, greater than or equal to | Relational |
| ==, != | Value equality and inequality | Relational |
| ==, != | Reference equality and inequality | Relational |
| && | Conditional AND | Logical |
| \|\| | Conditional OR | Logical |
| =, +=, -= | Assignment and compound assignments (addition and subtraction) | Assignment |

# Indicadores

- // Comentario de una línea
- /* Comentario de múltiples líneas
- */ Cierra el comentario de varias líneas
- /** Es el comienzo de un comentario javadoc