Programación Interactiva
Introducción a Java

Escuela de Ingeniería de Sistemas y Computación
Facultad de Ingeniería
Universidad del Valle

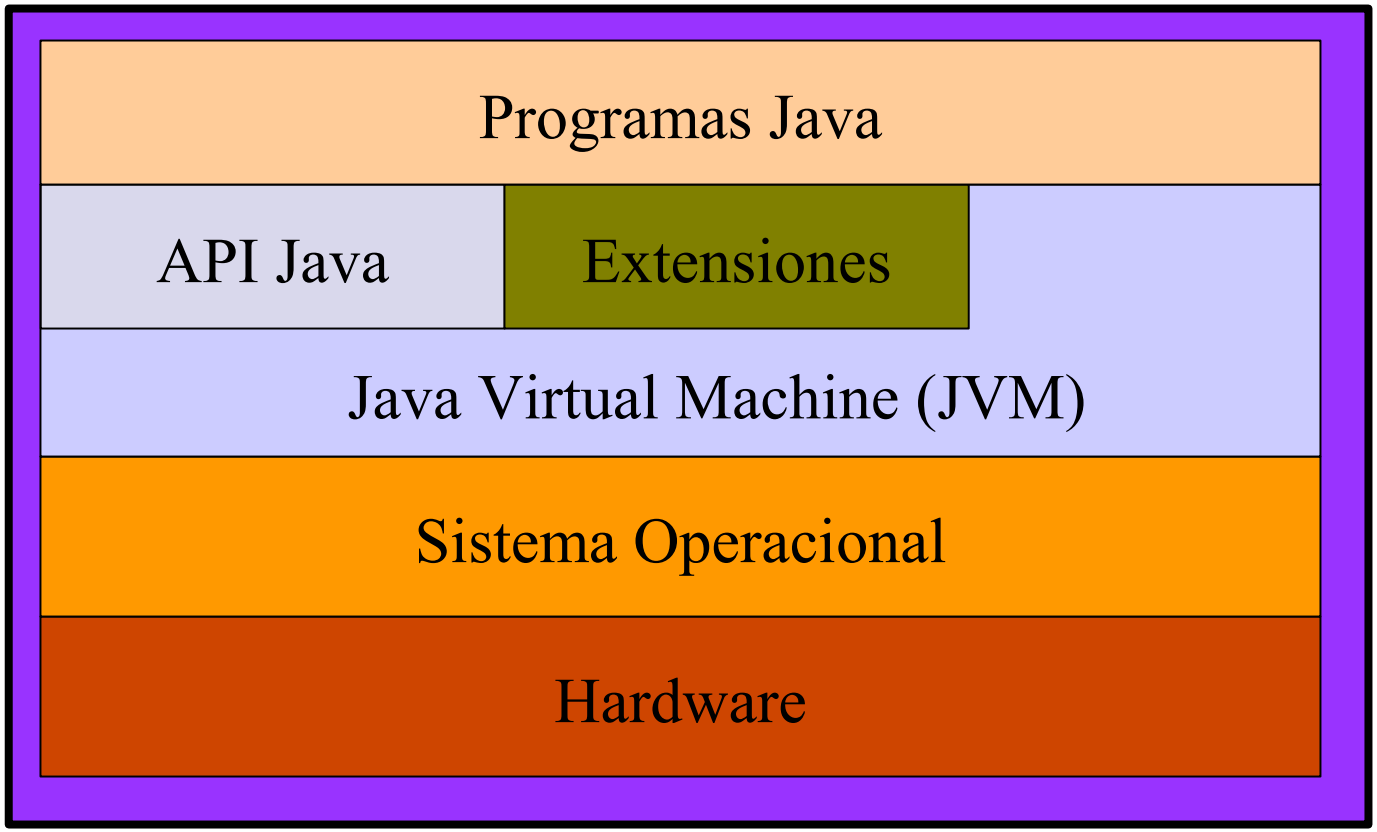
¿ Qué es Java ?

- Java es un lenguaje de programación de propósito general, fuertemente tipado, robusto, basado en clases y objetos, de nivel medio/alto.

- Es un lenguaje interpretado, es decir no es el SO el que ejecuta los programas Java sino una máquina virtual conocida como JVM (Java Virtual Machine).

- Desarrollado por Sun Microsystems (www.sun.com)

Arquitectura de Java



Tomado de las clases de Simena Dinas

¿ Cómo comienza la vida de un programa Java ?

Código Fuente (Archivo.java)

Compilador (javac)

Archivo compilado Bytecode (Archivo.class)

JVM (java)

Programa en ejecución

¿ Qué es el bytecode ?

- Es el tipo equivalente a un archivo .o en C++
- Cuando el javac compila un archivo de código fuente “Numeros.java” genera un archivo bytecode “Números.class” que corresponde a una versión del archivo .java que la JVM puede entender
- La utilidad de este proceso (en vez de la generación de un ejecutable) radica en la portabilidad del bytecode, ya que este archivo puede ejecutarse en cualquier SO que tenga una implementación de la JVM

Ventajas y Desventajas

- Retira parte de la responsabilidad sobre punteros (ventajas? desventajas?)
- Multiplataforma gracias a que es interpretado por la JVM (ventajas? desventajas?)
- Facilidad de uso
- Velocidad y uso de recursos (principales desventajas)
- Paso por parámetro, por valor ?
- Java API

Punteros en C++ y Java

- En C++ el usuario puede crear punteros a objetos u objetos, en Java, solo lo primero es posible (se les denomina referencias)

- En C++ la responsabilidad de liberar un puntero es del usuario, mientras en Java la JVM se encarga de esto a través de un proceso llamado “Garbage Collection”

Ventajas y desventajas de los lenguajes Interpretados

- Java, PHP, Scheme, Perl, entre otros, son lenguajes interpretados, es decir, no es el SO quien ejecuta el programa sino una máquina virtual (otro programa) que está siendo ejecutado por el SO
- Problemas de desempeño (gran problema)
- Mayor posibilidad de robo de código
- Multiplataforma (mayor ventaja), aunque esto depende solo aplica para las plataformas que tienen una implementación de la JVM (gran ventaja)

Facilidad De Uso De Java

- Gracias a que en Java un programador puede concentrarse más en lo que debe hacer su programa, y no tanto en el manejo de punteros y otros asuntos de bajo nivel (Java API)

- Una extensa librería de clases que permiten hacer casi todo lo que se puede en C++ (Java API)

- El código fuente queda menos plagado de pormenores que en C++

Ventajas De Java Sobre C++

- Gracias al API de Java los programadores reescriben menos código, los programas son más estandarizados en cuanto a métodos que utilizan
- El Garbage Collector despreocupa al programador sobre el desperdicio de memoria (en parte)
- Independiente de la plataforma

Desventajas de Java contra C++

- Los programas en C++ se ejecutan más rápidamente
- Mayor eficiencia en consumo de recursos
- La capacidad de crear **objetos reales** en C++ es útil cuando se requiere velocidad extra.
- Incapacidad de pasar objetos por valor (imitado por métodos de clonación)
- Carencia de herencia múltiple, imitado por la herencia múltiple de interfases en Java, aunque no totalmente.

Desventajas de Java contra C++

- Los programas en C++ se ejecutan más rapidamente
- Mayor eficiencia en consumo de recursos
- La capacidad de crear **objetos reales** en C++ es útil cuando se requiere velocidad extra.
- Incapacidad de pasar objetos por valor (imitado por métodos de clonación)
- Carencia de herencia múltiple, imitado por la herencia múltiple de interfases en Java, aunque no totalmente.
- Instrucción goto (reemplazada por bloques con nombre)

Java Development Kit (JDK)
Java Runtime Environment (JRE)

- El JRE es un paquete de software desarrollado por Sun que contiene la JVM y todo lo necesario para ejecutar programas Java

- El JDK es otro paquete que incluye todo lo necesario para crear y ejecutar programas Java (incluye el JRE, compilador, debugger, visor de applets, Java API, entre otras herramientas para el programador).

- Puede descargar las últimas versiones en www.sun.com (1.5.0 en este momento)

Java API

- El Java API (Application Program Interfase) es un conjunto de librerías que permiten el desarrollo de aplicaciones en Java, brinda funciones de uso común para el programador como por ejemplo:
  - Creación y manejo de elementos de GUI
  - Manejo de archivos
  - Funciones de red
  - Comunicación entre programas

Java API

- Existen dentro de su librería clases gráficas (awt y swing), las cuales permiten crear objetos gráficos comunes altamente configurables y con una arquitectura independiente de la plataforma.

- Hay gran cantidad de herramientas para generar interfaces gráficas como:
  - JBuilder
  - NetBeans
  - Fote4J
  - Jdeveloper
  - Eclipse

Java API

- Se puede acceder a bases de datos fácilmente con ***JDBC***, independientemente de la plataforma utilizada.

- Existen clases JDBC para las Bases de Datos más comunes, entre ellas:
  - Oracle
  - PostgreSQL
  - MySQL

Herramientas del JDK

- **javac:** El compilador Java, convierte código fuente (.java) en bytecodes (.class)
- **java:** El interprete de Java este programa es el que ejecuta los bytecodes
- **appletviewer:** Un interprete Java que ejecuta applets desde un archivo HTML, tal como lo haría un navegador web
- **javadoc:** Genera documentación de código fuente en formato HTML
- **javap:** El desensamblador Java, puede obtener el código fuente a partir de lso bytecodes
- **jdb:** El depurador (debugger), permite supervisar la ejecución de los programas Java
- **jar:** Permite almacenar un programa en Java que comprende muchos .class en un solo archivo .jar (en realidad, es un .zip)

Compilación y Ejecución

- Por ahora, digamos que cada clase debe estar definida en un archivo que debe llamarse igual que la clase, con extensión .java

- Es decir, la clase FiguraGeometrica debe declararse en un archivo llamado FiguraGeometrica.java

Compilación y Ejecución Aplicaciones

- Por ejemplo:

```
/******************** HolaMundo.java********************/

class HolaMundo
{
  public static void main (String args[])
  {
    System.out.println("Hola Mundo !");
  }
}
```

Compilación y Ejecución Applets

- Por ejemplo:

```
/* HolaMundoApplet.java */
import java.applet.Applet;
import java.awt.*;

class HolaMundoApplet  extends Applet
{
    public void paint (Graphics g)
    {
        g.drawString ("Hola Mundo !", 0, 20);
    }
}
```

```
/* HolaMundoApplet.java */
<HTML>
<HEAD>
<TITLE>Hello world</TITLE>
</HEAD>
<BODY>
<APPLET CODE="HolaMundoApplet.class"
        WIDTH=250 HEIGHT=100>
</APPLET>
</BODY>
</HTML>
```

Compilación y Ejecución

```
/********************** FiguraGeometrica.java **********************/
public class FiguraGeometrica {
   protected int numeroLados;
   protected String nombreFigura;

   public FiguraGeometrica(int nLados) {
      numeroLados = nLados;
   }
   public int getNumeroLados() {
      return numeroLados;
   }
   public String getNombreFigura() {
      return nombreFigura;
   }
}
```

Compilación y Ejecución

- Cuando esta clase se compile, se creará un archivo llamado FiguraGeometrica.class

- Si el nombre del archivo .java no coincidiera con el nombre de la clase, el compilador de Java no mostraría un mensaje de error (no en todos los casos, más adelante veremos):

```
OtroNombre.java:2: class FiguraGeometrica is public, should be
declared in a file named FiguraGeometrica.java
```

- Ahora veamos un par de ejemplos, una con salida por consola y otro mostrando un cuadro de diálogo:

Compilación y Ejecución

```
import javax.swing.*;

public class FiguraGeometricaFrame
{
   public static void main(String[] args)
   {
        FiguraGeometrica figura = new FiguraGeometrica(6);
        JOptionPane.showMessageDialog(null, "El numero de lados”+
                “de la figura geometrica es: "+
                figura.getNumeroLados());
        System.exit(0);
   }
}
```

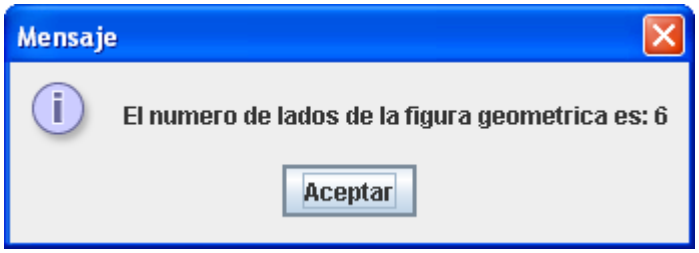
Compilación y Ejecución

Recuerde que la compilación se hace con:
javac FiguraGeometrica.java

Y la ejecución se hace con:
java FiguraGeometrica

Note que en la ejecución solo se coloca el nombre de la clase, sin la extensión

El resultado de la ejecución debe ser:



Material

- Descargar de

  http://eisc.univalle.edu.co/~dwilches/

  el material correspondiente a la clase 1:
  - JDK 1.5.0
  - Documentación del API de Java
  - Manual introductorio de Java
  - JCreator LE (entorno de desarrollo)

- Leer el tutorial que cubre las bases de programación:

  http://www.programacion.com/java/tutorial/java_basico

- Para los más avanzados:
  - Thinkin in Java de Bruce Eckel

IDE's
Entornos Gráficos de Desarrollo

- Otros enlaces:
  - Eclipse (http://www.eclipse.org) Open-Source
  - NetBeans (http://www.netbeans.org) Open-Source
  - JBuilder (http://www.borland.com/jbuilder) de Borland
  - Visual Age de IBM
    (http://www-4.ibm.com/software/ad/vajava/)
  - JDeveloper de Oracle
    (http://otn.oracle.com/products/jdev/content.html)