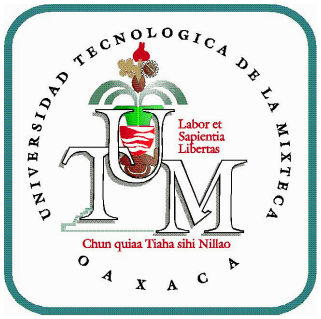

Java

una introducción.

Java: origen

- Diseñado dentro de Sun Microsystems por James Gosling
- El nombre original fue Oak
- Originalmente diseñado para usarse dentro de dispositivos electrodomésticos, que tuvieran la capacidad de comunicarse entre sí.
- Posteriormente fue reorientado hacia Internet, y rebautizado con el nombre de Java

Java: origen

- En un intento de resolver simultáneamente los problemas ocasionados por la diversidad y crecimiento de arquitecturas incompatibles, tanto entre máquinas diferentes como entre los diversos sistemas operativos y sistemas de ventanas que funcionaban sobre una misma máquina, añadiendo la dificultad de crear aplicaciones distribuidas en una red como Internet.

Java: características de diseño.

- Es un lenguaje de programación de alto nivel, de propósito general, y cuyas características son:
  - Simple y familiar.
  - Orientado a objetos.
  - Independiente de la plataforma
  - Portable
  - Robusto.
  - Seguro.
  - Multihilos.

Java: simple y familiar.

- Es simple:
  - Tanto la estructura léxica como sintáctica del lenguaje es muy sencilla.
  - Elimina las características complejas e innecesarias de sus predecesores.
- Es familiar:
  - Java incorpora las mejores características de lenguajes tales como:
    - C/C++, Modula, Beta, CLOS, Dylan, Mesa, Lisp, Smalltalk, Objective-C, y Modula 3.

Java: Orientado a Objetos.

- Todo en Java son objetos.
  - No es posible que existan funciones que no pertenezcan a una clase.
  - La excepción son los tipos da datos primitivos, como números, caracteres y booleanos.
- Cumple con los 4 requerimientos de Wegner:

OO = abstracción + clasificación + polimorfismo + herencia

Java: independiente de la plataforma.

- Java resuelve el problema de la distribución binaria mediante un formato de código binario (*bytecode*) que es independiente del hardware y del S.O., gracias a su máquina virtual.
- Si el sistema de *runtime* esta disponible para una plataforma específica, entonces una aplicación puede ejecutarse sin necesidad de un trabajo de programación adicional.

Java: portable.

- Independiente de la plataforma.
- La especificación de sus tipos de datos primitivos y sus tamaños, así como el comportamiento de los operadores aritméticos, son estándares en todas las implementaciones de Java.

Java: Robusto

- Robusto=confiable, gracias a:
  - Validación de tipos.
  - Control de acceso a las variables y métodos.
  - Validación de apuntador NULL.
  - Validación de los límites de un arreglo.
  - No tiene aritmética de apuntadores.
  - Manejo de memoria.

Robusto - Validación de tipos.

- Los objetos de tipos compatibles pueden ser asignados a otros objetos sin necesidad de modificar sus tipos.
- Objetos de tipos potencialmente incompatibles requieren un modificador de tipo (*cast*)
  - Si la modificación de tipo es claramente imposible, el compilador lo rechaza y reporta un erro en tiempo de compilación.
  - Si la modificación resulta legal, el compilador lo permite, pero inserta una validación en tiempo de ejecución.
  - Cuando el programa se ejecuta se realiza la validación cada vez que se ejecuta una asignación potencialmente invalida.

Robusto - control de acceso a variables y métodos.

- Los miembros de una clase pueden ser privados, públicos o protegidos.
- En java una variable privada, es realmente privada.
- Tanto el compilador como la máquina virtual de Java, controlan el acceso a los miembros de una clase, garantizando así su privacidad.

Robusto - Validación del apuntador Null

- Todos los programas en Java usan apuntadores para referenciar a un objeto.
- Una validación del apuntador a Null ocurra cada vez que un apuntador deja de referencia a un objeto.

Robusto: Limites de un arreglo.

- Java verifica en tiempo de ejecución que un programa no usa arreglos para tratar de acceder a áreas de memoria que no le pertenecen.

Robusto: aritmética de apuntadores.

- Aunque todos los objetos de manejan con apuntadores, Java elimina la mayor parte de los errores de manejo de apuntadores porque no soporta la aritmética de apuntadores:
  - No soporta acceso directo a los apuntadores
  - No permite operaciones sobre apuntadores.

Robusto: Manejo de memoria.

- Muchos de los errores de la programación se deben a que el programa no libera la memoria que debería liberar, o libera la misma memoria más de una vez.
- Java, hace recolección automática de basura, liberando la memoria y evitando la necesidad de que el programador se preocupe por liberar memoria que no utilice.

Java y C++: diferencias.

- No tiene aritmética de apuntadores.
- No permite funciones con ámbito global.
- No contiene estructuras y uniones (*struct* y *union*)
- No contiene tipos de datos sin signo.
- Elimina la instrucción *goto.*
- Las cadenas no terminan con ‘\0’

Java y C++: diferencias.

- No permite alias (*typedef*).
- No maneja macros (*#define*)
- No tiene conversión automática de tipos compatibles.
- No soporta un preprocesador.
- El tipo char contiene 16 bits para soportar UNICODE.

Java y C++: diferencias.

- Java soporta múltiples hilos de ejecución.
- Todas las condiciones en Java deben tener como resultado un tipo booleano.
- Java no soporta el operador *sizeof*.
- No tiene herencia múltiple.
- No tiene liberación de memoria explícita (*delete* y *free*()).

Archivos .java y .class

- El código fuente se almacena en archivos con extensión .java
- El *bytecode* o código compilado se almacena en archivos .class
- El compilador de Java crea un archivo .class por cada declaración de clase que encuentra en el archivo .java
- Los archivos .class tienen el nombre de la clase, no el nombre de los archivos .java

Java: Programas.

- Existen dos tipos de programas en Java:
  - Aplicaciones
    - Son programas *standalone*, escritos en Java y ejecutados por un intérprete del código de bytes desde la línea de comandos.
  - Applets
    - Son pequeñas aplicaciones escritas en Javam que siguen un conjunto de convenciones que le permiten ejecutarse dentro de un navegador.
    - Un applet siempre está incrustado en una página html.

Java: applet y aplicación.

- En términos del código fuente las diferencia entre un applet y una aplicación son:
  - Una aplicación debe definir una clase que contenga el método main(), que controla su ejecución. Un applet no usa el método main(); su ejecución es controlado por varios métodos definidos en la clase aplet.
  - Un applet, debe definir una clase derivada de la clase Applet.

Java: jdk

- Los principales programas del **Java™ Development Kit**:
  - javac. Es el compilador en línea del JDK.
  - java. Es la máquina virtual para aplicaciones de Java.
  - Appletviewer. Visor de applets de java.

Java: Archivo HTML

- Un applet debe ser invocado desde una página de html. Sólo es necesario agregar la etiqueta <APPPLET> para indicar el archivo de *bytecodes* del applet

<APPLET CODE="MiClase.class" WIDTH=250 HEIGHT=300></APPLET>

Java: Compilación

Utilizando el JDK, los programas se compilan desde el símbolo del sistema con el compilador *javac*.

Ejemplo:

C:\MisProgramas> javac MiClase.java

Java: Opciones del compilador

Sintaxis: javac [opciones] <archivo1.java>

Opciones:

- -classpath <ruta> Indica donde buscar los archivos de clasede Java
- -d <directorio> Indica el directorio destino para los archivos .class
- -g Habilita la generación de tablas de depuración.
- -nowarn Deshabilita los mensajes del compilador.
- -O Optimiza el código, generando en línea los métodos estáticos, finales y privados.
- -verbose Indica cuál archivo fuente se esta compilando.

Java: Aplicación Hola Mundo

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

Java: Applet Hola Mundo

```
import java.applet.*;
import java.awt.Graphics;
import java.awt.Color;
public class Hola extends Applet  {
    public void paint(Graphics g) {  // Java llama a
    paint automáticamente
       g.setColor(Color.red);
        g.drawString("¡Hola, Mundo!", 0, 50);
      }
}
```

Java: Ejecución

De un applet desde el appletviewer:

C:\> appletviewer hola.html

De una aplicación:

C:\> java HolaMundo

Java: Comentarios

- Java maneja tres tipos de comentarios:
  - /* comentario */
    - El compilador ignora cualquier cosa entre /* y */
  - // comentario
    - El compilador ignora cualquier cosa entre // y el fin de línea.
  - /** documentación*/
    - Indica un comentario de documentación. El compilador también ignora este tipo de comentario. La herramienta *javadoc* usa este tipo de comentarios para generar documentación de manera automática.

Java: método main()

- En una aplicación de Java debe existir un método de nombre main. este tiene un solo argumento (String args[ ]), a través del cual recibe información de los *argumentos de la línea de comandos*. ejemplo:

```
public class EscribeParam  {
    public static void main(String args[ ])       {
        for (int j= 0; j < args.length; j++)
            System.out.println("Parametro #" + j + "-> " +
    args[j]);
       }
}
```

Java: import

- Las clases en Java están organizadas en paquetes, y son similares a las librerías de c++ para agrupar funciones. Puede utilizarse opcionalmente la instrucción *import*, o hacer referencia a toda toda la ruta completa cada vez que se usa una clase:
  - java.util.Hashtable miTabla = new java.util.Hastable();
  - import java.util.Hashtable; //importar esta clase
  - Hashtable miTable = new Hashtable();

Java: import

- Existen algunas clases que no necesitan ser importadas, como la clase *System*, estas son las que se encuentran dentro del paquete *java.lang*, pues son importadas automáticamente para todo programa de Java.
- Es posible importar todas las clases de un paquete mediante el uso del *. Ejemplo:
  - import java.awt.*;

Java: tipos de datos primitivos.

Tipo	Tamaño	Rango
long	8 bytes	-9,223,372,036,854,775,808L a 9,223,372,036,854,775,807L
int	4 bytes	-2.147,483,648 a 2,147,483,647
short	2 bytes	-32768 a 32,767
byte	1 byte	-128 a 127
double	8 bytes	+- 1.79769313486231570E+308
float	4 bytes	+- 3.40282347E+38
char	2 bytes	65,536 caracteres posibles
boolean	1 bit	*true* o *false*

Java: Conversiones de tipos.

- Existe una conversión implícita cuando se trata de un tipo de dato más pequeño hacia uno más grande. La conversión implícita soportada sería:

double <- float <- long <- int <- short <- byte

- De otra forma debe usarse una conversión explícita mediante conversión o enmascaramiento (cast):

(tipo) valor_a convertir

Java: Definir una clase

- Debe especificarse la palabra reservada *class*, seguida del nombre de la clase:

```
class MiNuevaClase{
    //instrucciones
}
```

Pueden definirse múltiples clases en un archivo .java, pero solo puede haber una clase pública por archivo.

Java: Definir una clase

- Una clase pública es una clase que puede utilizarse en otros programas. el nombre de la clase pública debe coincidir con el nombre del archivo .java
- Existen dos formas de crear un objeto de una clase:

```
MiNuevaClase obj1;
obj1 = new MiNuevaClase();
```

ó

```
MiNuevaClase obj1 = new MiNuevaClase();
```

Java: atributos

- La declaración de atributos es similar a la forma en C++; es decir, se especifica el tipo de dato o clase a la que pertenece y el nombre del atributo.
- Java a diferencia de C++, asigna valores predeterminados a los atributos.

```
class Fruta {
    float precio;
    String Variedad;
    ...
}
```

Java: métodos

- De la misma forma, la declaración de métodos es parecida a la de C++.

```
public class Fruta {
   float precio;
   String variedad;
    Fruta (float p, String v) {
      precio = p;      variedad = v;
   }
   void desplegar() {
      System.out.println ("Variedad: "+ variedad +          ",
    Precio: " + precio);
   }
}
```

Java: sobrecarga de métodos

- La sobrecarga de métodos consiste en declarar dos o más métodos en una clase con el mismo nombre, pero que se diferencien en el número o en el tipo de los parámetros que reciben.

```
public class Fruta {
   float precio;
   String variedad;
   Fruta (float p, String v) {
     precio = p;      variedad = v;
   }
   Fruta (String v) {
     precio = 1.00F;
     variedad = v;
   }
   void desplegar() {
     System.out.println ("Variedad: "+ variedad +         ", Precio: " + precio);
   }
}
```

Java: Paso de parámetros

- Se puede pasar como parámetro cualquier tipo de dato válido: tipos de datos primitivos y tipos de datos referenciados (objetos y arreglos).
- Los argumentos son pasados por valor.
  - Si el parámetro es de un tipo de datos primitivo, el paso por valor significa que el método no puede cambiar su valor.
  - Cuando el argumento es un tipo de dato referenciado, el paso por valor significa que elmétodo no puede cambiar la referencia al objeto.

Java: Modificadores de clase.

- **Abstract**. Declara a la clase como abstracta y evita que un programa pueda crear una instancia de esa clase.

  public **abstract** class ClaseAbstracta {...

- **Public**. Este modificador vuelve accesible a la clase en cualquier lugar. Si no se usa *public*, la clase es visible únicamente en el paquete al que pertenece. Solo se permite una clase *public* por archivo .java.

  public class EstoyEnTodosLados {...

Java: Modificadores de clases.

- **Final.** Incluir la palabra reservada *final* en la declaración de la clase evita que se creen subclases de ella. Esto evita que programadores extiendan una clase.

```
public final class AquiTerminatodo {...
```

Java: Modificadores de miembros

- **Public.** Un miembro público es visible desde cualquier lugar del programa donde la clase sea visible.

  public int todosMeVen;

- **Private.** Un miembro privado es visible sólo dentro de su clase.

  private int InvisibleAfuera

- **Protected.** Un miembro protegido sólo es visible dentro de su clase, subclases o en clases que pertenecen al mismo paquete.

  protected int atributoProtegido;

Java: Modificadores de miembros

- **Static**. Los miembros estáticos son accesibles sin necesidad de instanciar un objeto de una clase. el valor del atributo es compartido por todas las instancias de la clase. Los métodos estáticos no pueden acceder a miembros no estáticos de la clase.

  public **static** int EsteValorSeComparte;

  public **static** int llamameCuandoQuieras() { ... }

Java: Modificadores de miembros

- **Atributo Final**. Declarar a un atributo como *final*, indica que este atributo será constante y que su valor no puede ser modificado.

```
protected static final int MAX_LLAVES = 256;
```

- **Método Final**. Si se declara a un método como *final*, se indica que este método no podrá ser redefinido en las clases derivadas.

```
public final int esteMetodoNoSeReemplaza(){ ...
   }
```

Java: Modificadores de miembros

- **Abstract.** Un método abstracto no tiene implementación, y no se permite crear una instancia de la clase que contiene al método. Una subclase debe redefinir el método incluyendo la implementación.

```
public abstract void instrumentameDespues( int
   x);


//Definición inválida de un método abstracto
final abstract void metodoIlegal();
```

Java: Síntesis de acceso

Nivel de acceso	clase	subclase	paquete	Mundo
private	X			
Protected	X	X	X	
Public	X	X	X	X
package	X		X	

Java: Arreglos

- En Java los arreglos son objetos que son creados dinámicamente.
- La declaración de un atributo de tipo arreglo no lo crea, ni reserva memoria.

```
int [] a= new int[10];
float objeto[]= new float[23];
int [] factorial = { 1, 1, 2, 6, 24, 120, 720, 5040 };
byte arr[][]= new byte [2][3];
```

Java: Arreglos

```
public class referenciaArreglog {
  static public void main(String args){
       int primero[] = { 1, 2, 3, 4 };
      int segundo[] = { 5, 6, 7, 8, 9, 10 };       int arreglo[];
      arreglo = primero;
      System.out.println("Primer Arreglo: ");
      for (int indice=0; indice < arreglo.length; indice++){
            System.out.println(arreglo[indice]);
     }
      arreglo = segundo;
      System.out.println("Segundo Arreglo:");
      for (int indice=0; indice < arreglo.length; indice++){
          System.out.println(arreglo[indice]);
        }
    }
}
```

Java: Arreglos

```
public class cicloArreglo {
   static public void main(String args) {
      Integer arreglo[][] = new Integer [50][5];
     for (int i=0; i < arreglo.length; i++) {
        for (int k=0; k < arreglo[i].length; k++) {
           arreglo[i][k] = new Integer(i*10 + k);
           System.out.println(arreglo[i][k]);
         }
      }
   }
}
```

Java:Métodos del Applet

public void init()	Es invocada una sola vez, cuando se carga el applet. Sirve para inicializar el applet.
public void start()	Se invoca después de que se ejecuta *init* y cada vez que el usuario regresa a la página HTML en la que el applet reside.
public void paint (Graphics g)	Se invoca para dibujar en el applet después que el *init* termina de ejecutarse y se ha comenzado a ejecutar el método *start*. Se invoca automáticamente cada vez que el applet necesita redibujarse.
public void stop()	Se invoca cuando el appelt debe terminar su ejecución; por ejemplo, que el usuario abandone la página HTML
public void destroy()	Se invoca cuando el applet se va a eliminar de la memoria; por ejemplo, cuando el usuario sale del navegador.

Java: Applet

```
import java.awt.Graphics;
import java.applet.Applet;
public class InitArray extends Applet {
   int n[];
   public void init(){
      n = new int[ 10 ];    }
   public void paint( Graphics g ){
      int yPosition = 25;
      g.drawString( "Element", 25, yPosition );
      g.drawString( "Value", 100, yPosition );
      for ( int i = 0; i < n.length; i++ ) {
         yPosition += 15;
         g.drawString( String.valueOf( i ), 25, yPosition );
         g.drawString( String.valueOf( n[ i ] ), 100, yPosition );
      }
   }
}
```

Java: Applet

```
import java.awt.Graphics;
import java.applet.Applet;
public class SumArray extends Applet {
   int a[] = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };
   int total;
   public void init(){
      total = 0;
      for ( int i = 0; i < a.length; i++ )
         total += a[ i ];
   }
   public void paint( Graphics g ) {
      g.drawString( "Total : " + total,25, 25 );
   }
}
```

Java: Seguridad en applets

Acción	Appletviewer	Netscape (archivo)	Netscape (URL)	Applet de Java
Conexión a otra computadora	Sí	No	No	Sí
Conexión a la computadora local	Sí	Sí	No	Sí
Leer un archivo local.	Sí	No	No	Sí
Escribir en un archivo local	Sí	No	No	Sí
Borrar un archivo desde Java	No	No	No	Sí
Ejecutar programas externos	Sí	No	No	Sí
Cargar biblioteca	Sí	Sí	No	Sí

Java:Paquetes de la API

API: Applications Programming Interface

java.applet	Contiene a la clase applet y varias interfaces que permiten la creación de applets, interacción de las applets con el navegador, etc.
java.awt	Paquete de herramientas para trabajar con ventanas. Clases e interfaces pata crear y manipular interfaces gráficas con el usuario(GUI)
java.awt.image	Contiene clases e interfaces que permiten almacenar y manipular imágenes en un programa.
java.io	Entrada/salida de Java.
java.lang	Importado automáticamente en los programas.
java.net	Clases que permiten a los programas comunicarse a través de la Intranet o Internet.
java.util	Diversas clases como manipulaciones de fecha, aleatorios, manipulaciones de cadena, etc.