



UNIVERSIDAD
Popular del cesar

Ingeniería de Sistemas

Programación de Computadores II



UNIDAD 1:

INTRODUCCION A LA PROGRAMACION ORIENTADA A OBJETOS

1.1. Paradigmas de la programación

1.2. Características de la Programación Orientada a Objetos

1.3. Conceptos de Programación Orientada a Objetos

1.3.1. Abstracción

1.3.2. Encapsulamiento

1.3.3. Polimorfismo

1.3.4. Herencia

1.3.5. Modularidad

1.4. Clases y Objetos

1.5. Mensajes y Métodos

1.6 Nociones de lenguaje de programación orientado a objetos

1.6.1 Generalidades

1.6.2 Sintaxis básica y estructuras de control

1.6.3 Vectores y matrices



UNIDAD 1:

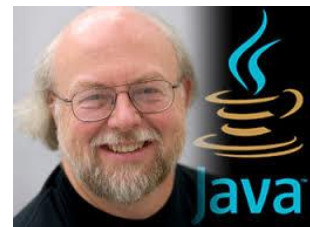
OBJETIVOS DE APRENDIZAJE

- Presentar ante los estudiantes el Lenguaje de Programación Java, como herramienta base para la enseñanza de a asignatura.
- Presentar ante los estudiantes los componentes básicos requeridos para la escritura y ejecución de aplicativos escritos en Java, su descarga e instalación.
- Explicar a los estudiantes las etapas de ejecución de aplicativos escritos en Java
- Presentar ante los estudiantes las reglas básicas de sintaxis para la escritura de programas en Java.
- Presentar ante los estudiantes las instrucciones de control de flujos de programas en Java (Condicionales y Ciclos)



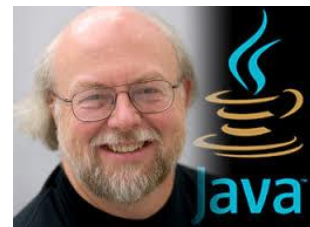
Generalidades: Un recorrido rápido por Java - Historia

- Desarrollado por **Sun Microsystems** a principio de los años 90's.
- Diseñado por **James Gosling** como lenguaje de programación para electrodomésticos. Con el objetivo de “escribir una vez, ejecutar en cualquier parte”.
- En el mes de Mayo de 1995 se hace su lanzamiento oficial.
- Adquiere Popularidad por aumentar el potencial de la WWW, para agregar contenido dinámico a las paginas web (interactividad y animaciones)
- Toma mucha de sus sintaxis de **C y C++**, a diferencia este tiene un modelo de objetos más simple
- Adquirido por la compañía **Oracle** en 2009.
- Utilizado en la actualidad para:
 - Desarrollo de aplicaciones empresariales a gran escala
 - Mejorar funcionalidad de servidores web
 - Aplicaciones para dispositivos de uso domestico
 - Otros.



Generalidades: Un recorrido rápido por Java - Características

- **Lenguaje Simple:** Basado en C y C++
- **Orientado a Objeto:** Toda la programación está orientada a objeto.
- **Distribuido:** Facilita la creación de aplicaciones distribuidas
- **Robusto:** Realiza verificaciones de problemas en tiempo de compilación y ejecución.
- **Seguro:** Implementa barreras de seguridad
- **Portable e Indiferente a la arquitectura :** El código compilado Java puede ejecutarse sobre cualquier máquina.
- **Interpretado y compilado :** EL código fuente (.java) se compila para obtener códigos de bytes (.class) que es interpretado por la JVM.
- **Multihilos:** Varios procesos en la misma maquina simultáneamente
- **Alto rendimiento:** es veloz en el momento de correr los programas y por ahorrase muchas líneas de código.



Generalidades: Un recorrido rápido por Java – Tipos de aplicaciones

Aplicaciones Standalone (escritorio): Se ejecutan como programas Independientes, no necesitan un browser para su ejecución, requieren la ayuda de la máquina virtual de Java.

☐ **Programa de consola:** programas que funcionan en modo texto

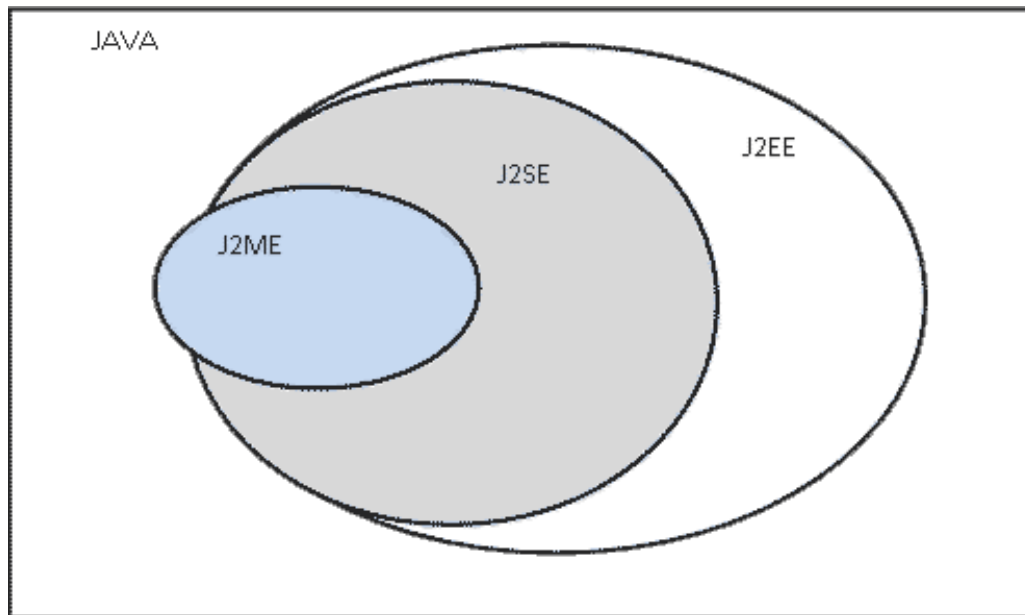
Programa Grafico: Interacción mediante componentes gráficos.

Java Applets: Pequeñas aplicaciones Java que se integran en una pagina web y requieren de un browser para su ejecución, en lugar de ejecutarse como aplicación independiente.

Servlets: Programas que están pensados para trabajar en el lado del servidor y desarrollar aplicaciones Web que interactúen con los clientes.



Generalidades: Un recorrido rápido por Java – Distribuciones



J2SE: Java 2 Standard Edition o Java Standard Edition. Se utiliza principalmente para crear aplicaciones para entornos de escritorio. No incluye soporte a tecnologías para internet.

J2EE: Java 2 Enterprise Edition o Java Enterprise Edition. Orientada al desarrollo de servicios web, y otras aplicaciones basadas en web.

J2ME: Java 2 Micro Edition o Java Micro Edition. Para aplicaciones que se ejecutan en sistemas integrados, móviles y dispositivos pequeños.



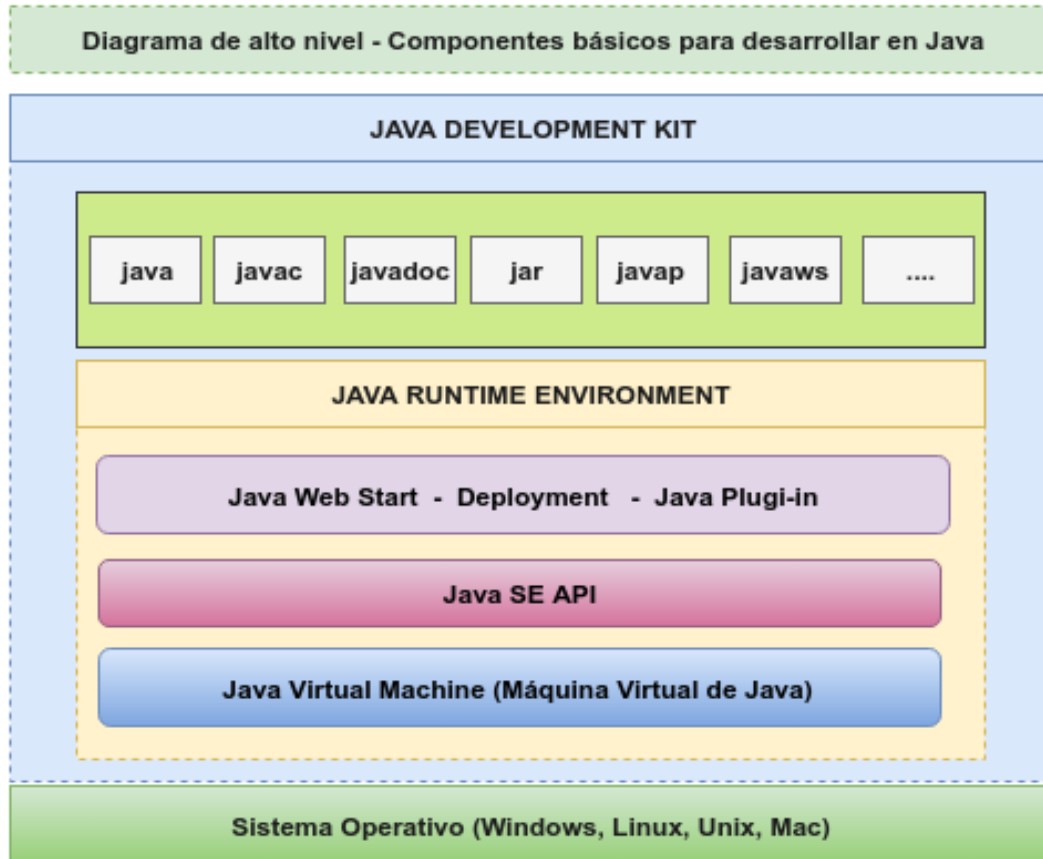
Generalidades: Un recorrido rápido por Java – Versiones

Oracle Java SE Support Roadmap [†]				
Release	GA Date	Premier Support Until	Extended Support Until	Sustaining Support
7 (LTS)	July 2011	July 2019	July 2022*****	Indefinite
8 (LTS)**	March 2014	March 2022	December 2030*****	Indefinite
9 (non-LTS)	September 2017	March 2018	Not Available	Indefinite
10 (non-LTS)	March 2018	September 2018	Not Available	Indefinite
11 (LTS)	September 2018	September 2023	September 2026	Indefinite
12 (non-LTS)	March 2019	September 2019	Not Available	Indefinite
13 (non-LTS)	September 2019	March 2020	Not Available	Indefinite
14 (non-LTS)	March 2020	September 2020	Not Available	Indefinite
15 (non-LTS)	September 2020	March 2021	Not Available	Indefinite
16 (non-LTS)	March 2021	September 2021	Not Available	Indefinite
17 (LTS)	September 2021	September 2026****	September 2029****	Indefinite
18 (non-LTS)	March 2022	September 2022	Not Available	Indefinite
19 (non-LTS)***	September 2022	March 2023	Not Available	Indefinite
20 (non-LTS)***	March 2023	September 2023	Not Available	Indefinite
21 (LTS)***	September 2023	September 2028	September 2031	Indefinite



Para lanzamientos de productos posteriores a Java SE 8, Oracle designará solo ciertos lanzamientos como **lanzamientos de soporte a largo plazo (LTS)**. **Java SE 7, 8, 11 y 17 son versiones LTS**

Generalidades: Un recorrido rápido por Java – JAVA SE



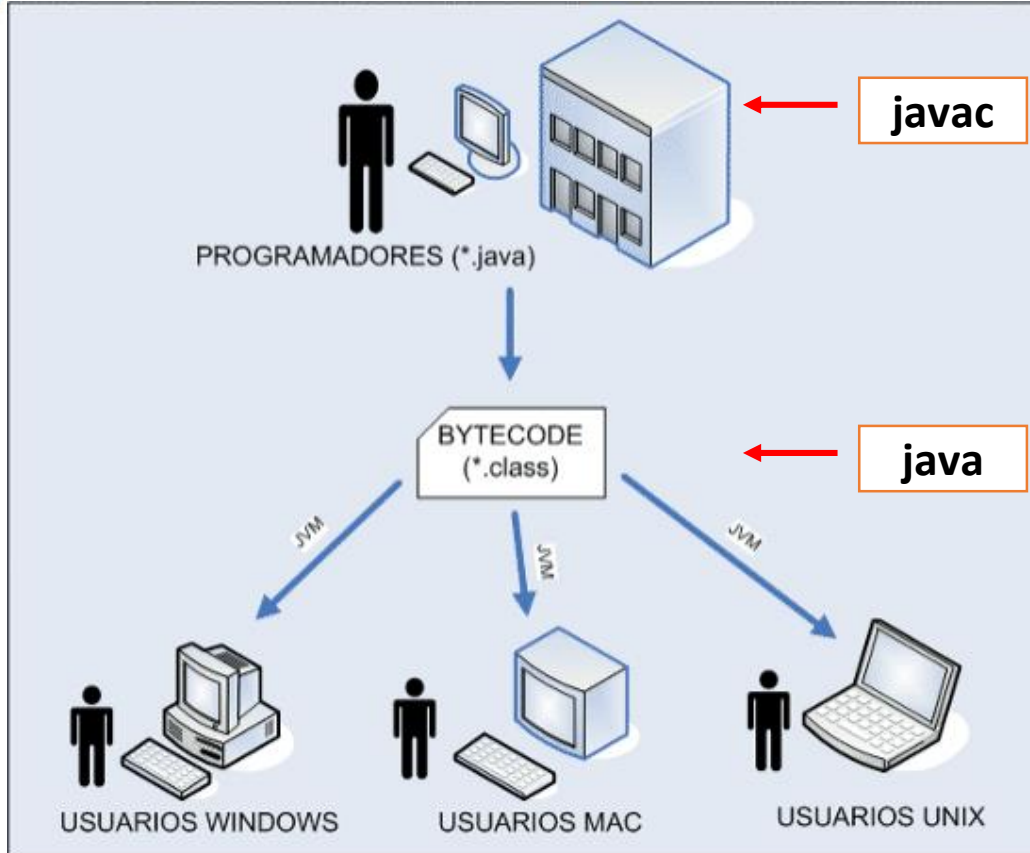
JDK: Java Development Kit. Proporciona entre otras herramientas, un compilador, un interpretador, un mecanismo de compresión y un **entorno de ejecución** de proyectos. Contiene además APIs de desarrollos y otros mecanismos para el desarrollo de software.

JRE: Java Runtime Environment, consiste en el conjunto de bibliotecas y herramientas necesarias para ejecutar y administrar aplicaciones Java, incluye la JVM.

JVM: Java Virtual Machine. Es la que hace la correspondencia entre código de Java y Bytecodes. Ésta en tiempo real genera un binario que puede ejecutarse en la plataforma donde esté instalado.



Generalidades: Un recorrido rápido por Java – JVM



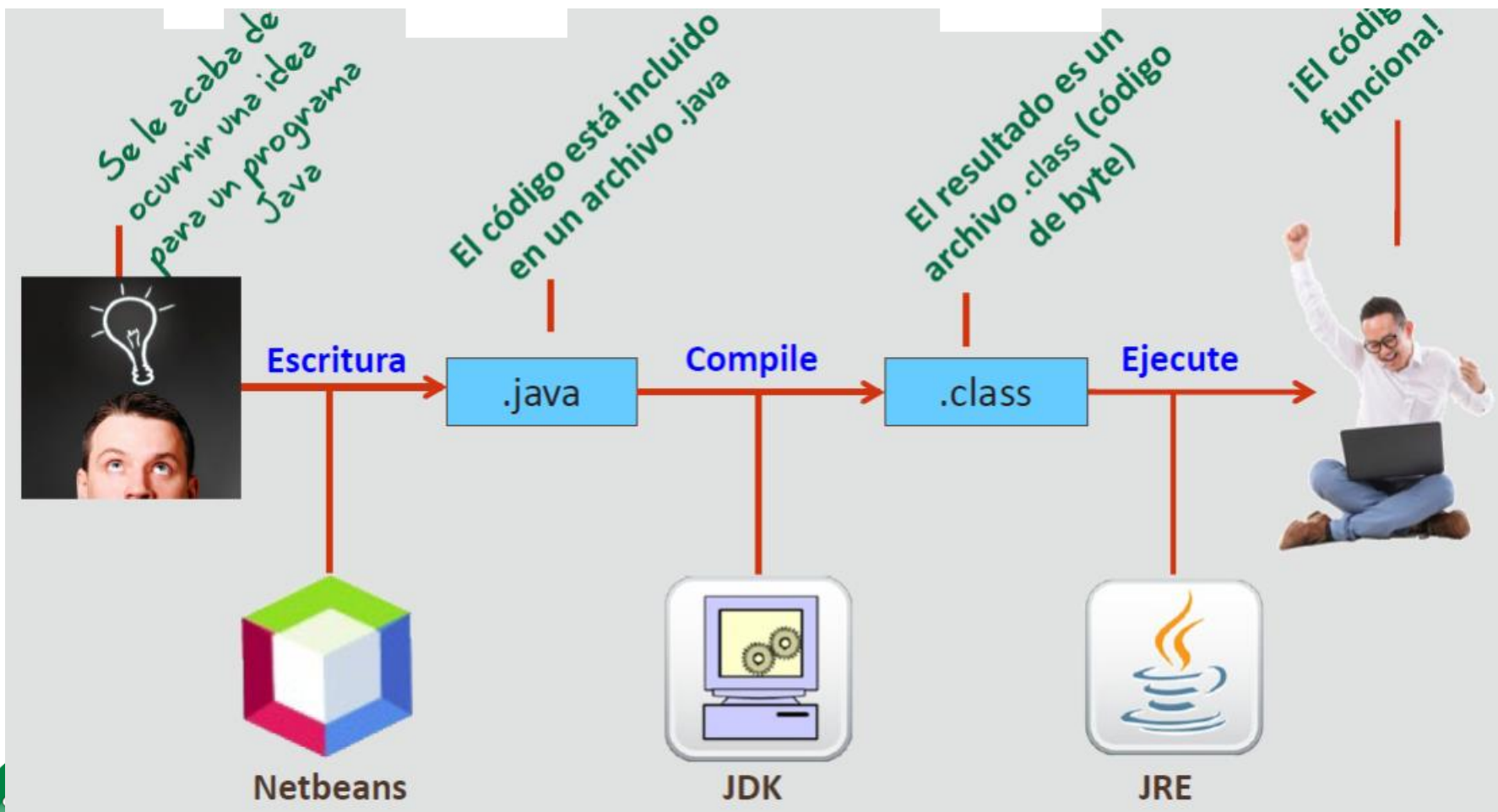
javac: Dada una clase **.java** inicia la compilación del mismo a ByteCode. Este comando no está disponible en la JRE

java: Inicia el entorno de ejecución, para que funcione se le pasa como parámetro un fichero binario ejecutable de tipo ByteCodes (**.class**).

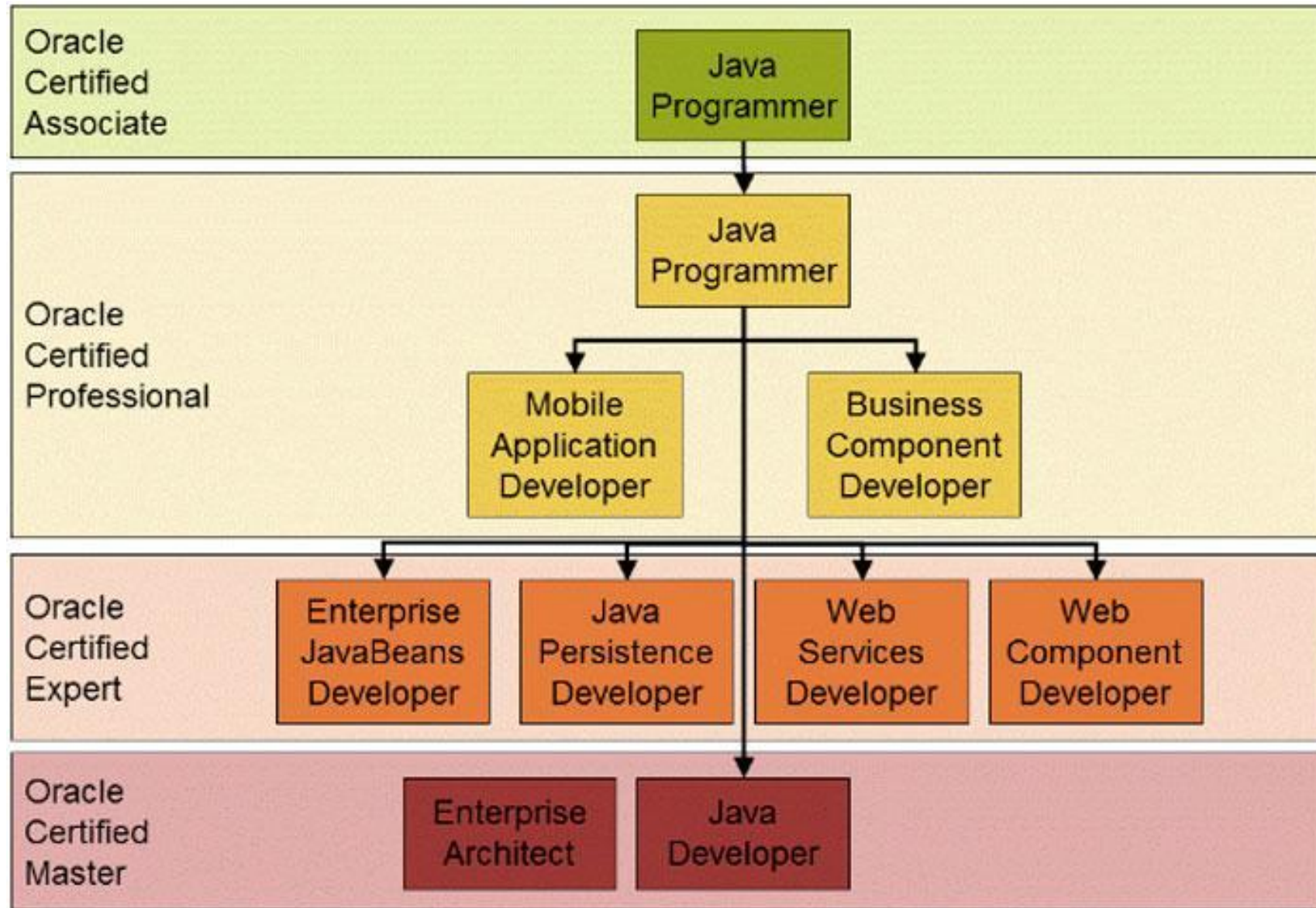
jar: Inicia el empaquetador de clases de Java en un fichero **.jar**. Solo disponible en el JDK.



Generalidades: Un recorrido rápido por Java – Fases de ejecución de un programa



Generalidades: Certificaciones Java



Generalidades: Entorno de desarrollo



Generalidades: Ejecución programa en java

1. Descarga el código Java del siguiente repositorio:
<https://github.com/UpcProgramingII/2023-II.git>
2. Copia los archivos descargados en una carpeta en C
3. Abre el Símbolo del sistema
4. Direcciona a la carpeta creada
5. Ejecuta comando javac
6. Ejecuta comando java

```
Bienvenido.java
1  /* -----
2      Autor: Jairo francisco seoanes leon
3      Fecha: 29-08-2013
4      Descripcion: Mi primer programa
5  */
6  //Declaracion de la clase Bienvenido
7  public class Bienvenido{
8      // declaracion del metodo main
9      public static void main(String[] arg){
10
11          // Mostrar mensaje en pantalla
12          System.out.println("Bienvenido a Java");
13
14      } // fin del metodo main
15  }//fin de la clase Bienvenido
```

```
C:\> javac Bienvenido.java
```

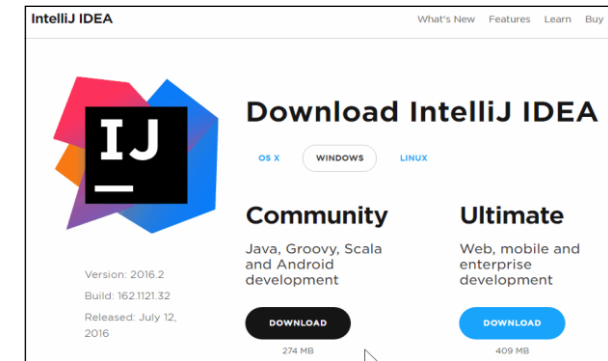
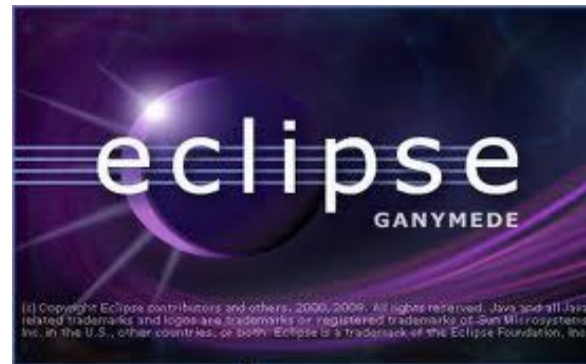
```
C:\> java Bienvenido
```



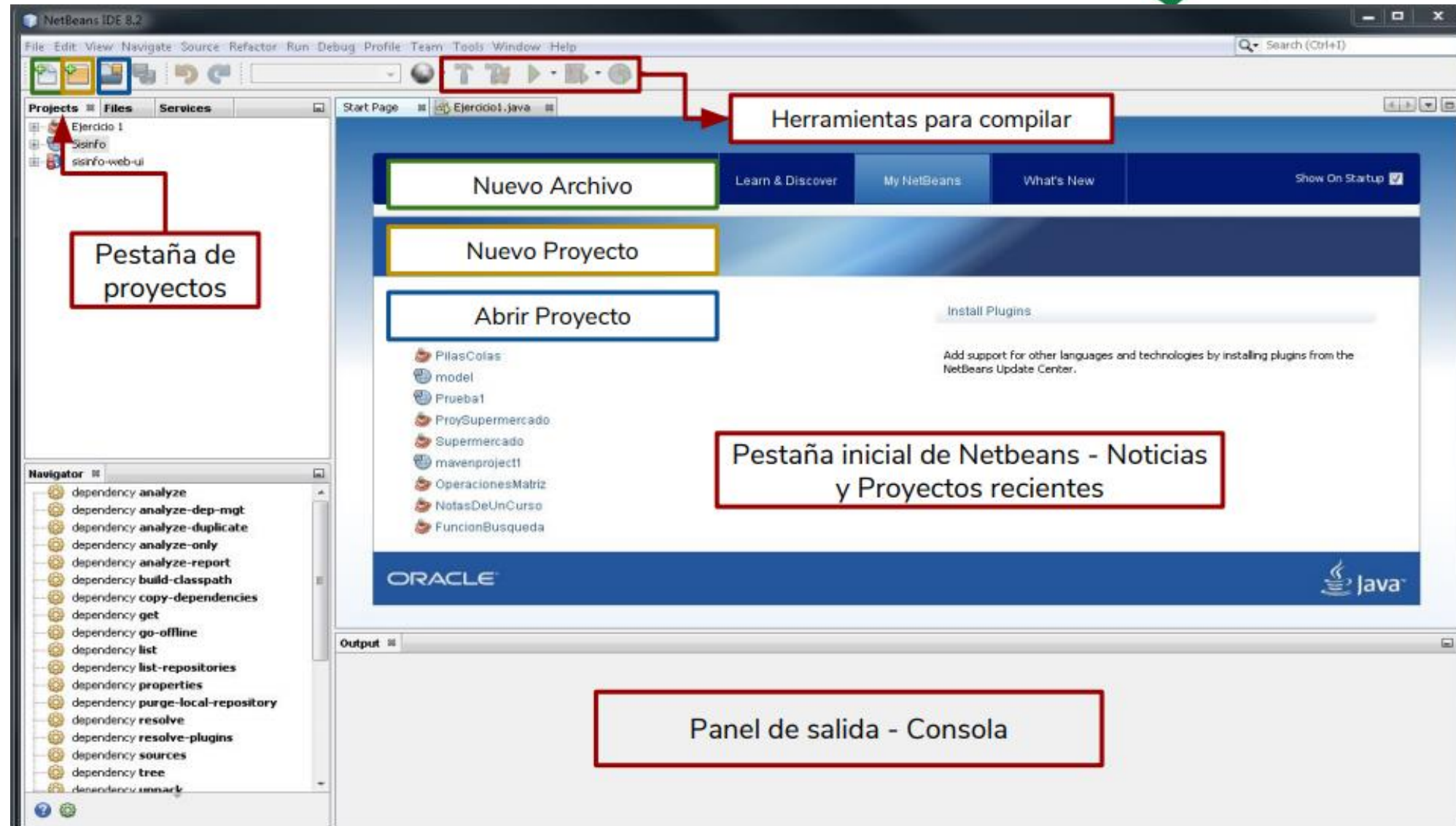
Generalidades: Entornos de desarrollo integrado IDE

Para las organizaciones que desarrollan sistemas de información extensos, hay **entornos de desarrollo integrados (IDEs) disponibles**. Los IDEs proporcionan herramientas que dan soporte al proceso de desarrollo del software, incluyendo editores para escribir y editar programas, y depuradores para localizar errores lógicos.

Los IDEs populares son: Eclipse (www.eclipse.org), NetBeans(www.netbeans.org – www.oracle.com), JCreator (www.jcreator.com)



Generalidades: El entorno de desarrollo - IDE



NetBeans IDE
www.netbeans.org



Generalidades: Sintaxis básica – Salida por pantalla

El objeto ***System.out*** dispone de métodos para escribir con formato en el flujo de salida, usualmente el monitor.

Los más interesantes son:

print, método sobrecargado que puede recoger cualquier tipo de datos básico, cadena u objeto;

println, idéntico a *print* salvo en que imprime un salto de línea final;

Format o printf, que permite escribir los datos ajustándose a un determinado formato

<https://github.com/jfseoanesl/Unicesar-Programacion-II-Unidad-1.-Java-Basic-Examples.git>

Caracteres de conversion

%d : Numerico entero [byte, short, int, long]

%f : Numerico punto flotante [float, double]

%c : Caracter

%s: Cadena de caracteres

%n: Salto de linea

/n: Salto de linea



Generalidades: Sintaxis básica – Salida por pantalla

MiPrimerPrograma - NetBeans IDE 7.3.1

File Edit Format Preview View Navigate Source Refactor Run Debug Profile Team Tools Window Help

<default config> cnxsrpaBD

Files Servi... Favi... Start Page x MiPrimerPrograma.java x

Source History

```

7 // nombre de la clase principal
8 public class MiPrimerPrograma {
9     // metodo principal - metodo Main
10    public static void main(String[] args) {
11        System.out.println("1. Mi primer programa");
12        System.out.print("2. Mi primer programa-");
13        String msg="3. Mi primer programa";
14        System.out.printf("%s\n", msg);
15        msg="4. Mi primer programa";
16        System.out.printf("%s", msg);
17        System.out.format("%s",msg);
18    }
19 }

```

main - Navigator x

Members <empty>

MiPrimerPrograma

main(String[] args)

Output - MiPrimerPrograma (run) x

run:

```

1. Mi primer programa
2. Mi primer programa-3. Mi primer programa
4. Mi primer programa4. Mi primer programaBUILD SUCCESSFUL (total time: 1 second)

```



Generalidades: Sintaxis básica - Comentarios

Los comentarios se utilizan para explicar el código fuente. Un comentario es un texto que se escribe dentro de un programa con el fin de facilitar la comprensión del mismo.

En Java se pueden utilizar tres tipos de comentarios:

La primera, **para comentarios de una sola línea.**

// comentario de una línea.

```
6 //Declaracion de la clase Bienvenid
7 public class Bienvenido{
8     // declaracion del metodo main
```

La siguiente para **comentarios de más de una línea.**

/* Esto sería un comentario de dos líneas*/

```
1 /* -----
2     Autor: Jairo francisco seoane
3     Fecha: 29-08-2013
4     Descripcin: Mi primer progra
5 */
```

Y finalmente, en tercer lugar unos **comentarios especiales** que los usaremos para elaborar documentación con la herramienta de documentación de Java, JavaDoc.

/ Este será un cometario de la herramienta JavaDoc*/**



Generalidades: Sintaxis básica - Comentarios

TAG	DESCRIPCIÓN	COMPRENDE
<code>@author</code>	Nombre del desarrollador.	Nombre autor o autores
<code>@deprecated</code>	Indica que el método o clase es obsoleto (propio de versiones anteriores) y que no se recomienda su uso.	Descripción
<code>@param</code>	Definición de un parámetro de un método, es requerido para todos los parámetros del método.	Nombre de parámetro y descripción
<code>@return</code>	Informa de lo que devuelve el método, no se aplica en constructores o métodos "void".	Descripción del valor de retorno
<code>@see</code>	Asocia con otro método o clase.	Referencia cruzada referencia (#método()); clase#método(); paquete.clase; paquete.clase#método()).
<code>@version</code>	Versión del método o clase.	Versión

javadoc utiliza ciertas etiquetas (tags) precedidas por el carácter "@", dentro de los símbolos de comentario. Si no existe al menos una línea que comience con @ no se reconocerá el comentario para la documentación de la clase.



Generalidades: Sintaxis básica – Variables

Las **Variables** permiten acceder a la memoria para almacenar y recuperar los datos con los que nuestros programas van a trabajar.

Java es un lenguaje tipado y nos obliga a declarar nuestras variables antes de poder hacer uso de ellas.

valor.entero (en 32 bits) que vale 123456789

0xfde23a50	
0xfde23a51	
0xfde23a52	
0xfde23a53	
0xfde23a54	
0xfde23a55	
0xfde23a56	
0xfde23a57	123456789
0xfde23a58	
0xfde23a59	

```
int var = 123456789;
```

```
var = var + 1 ;
```

```
long total=999999999;
long ss_num = 999_99_9999;
double d1 = 123.4;
double d2 = 1.234e2;
```

El ámbito de las variables

```
public void someMethod()
{
    if(gameOver && score>highScore)
    {
        String name;                //Point A
        System.out.println("Please enter your name:");
        name=reader.next();
    }//end if                        //Point B

    System.out.println("Thank you " + name + ", ");
    System.out.println("your high score has been saved.");
} //end method someMethod
```



Generalidades: Sintaxis básica – Tipos primitivos

Tipo de dato	Representación	Tamaño (Bytes)	Rango de Valores	Valor por defecto	Clase Asociada
byte	Numérico Entero con signo	1	-128 a 127	0	Byte
short	Numérico Entero con signo	2	-32.768 a 32.767	0	Short
int	Numérico Entero con signo	4	-2147483648 a 2147483647	0	Integer
long	Numérico Entero con signo	8	-9223372036854775808 a 9223372036854775807	0	Long
float	Numérico en Coma flotante de precisión simple Norma IEEE 754	4	$\pm 3.4 \times 10^{-38}$ a $\pm 3.4 \times 10^{38}$	0.0	Float
double	Numérico en Coma flotante de precisión doble Norma IEEE 754	8	$\pm 1.8 \times 10^{-308}$ a $\pm 1.8 \times 10^{308}$	0.0	Double
char	Carácter Unicode	2	\u0000 a \uFFFF	\u0000	Character
boolean	Dato lógico	-	true ó false	false	Boolean
void	-	-	-	-	Void

En muchos casos se requiere la conversión entre tipos de datos.

```
Integer i = Integer.valueOf("12345");
int j = Integer.parseInt("123456");
int k = (int) 2.5;
```

Promociones y conversiones entre tipos de datos



Generalidades: Sintaxis básica – Constantes

Cuando se declara una variable de tipo **final**, se ha de inicializar y cualquier intento de modificarla en el curso de la ejecución del programa da lugar a un error en tiempo de compilación.

Normalmente, las constantes de un programa se suelen poner en letras mayúsculas, para distinguirlas de las que no son constantes. He aquí ejemplos de declaración de constantes.

```
final double PI=3.141592653589793;
```

```
final int MAX_DATOS=150;
```

```
char curso;  
char docente;  
int numEstudiantes;
```

```
final int salonDeClase = 302;
```



Generalidades: Sintaxis básica – Identificadores

Reglas:

Son los nombres que podemos ponerles a nuestros/as variables, métodos, clases, interfaces y objetos.

Todas las variables y constantes deben ser definidas antes de ser usadas.

- Todos los identificadores han de comenzar con una letra, el carácter subrayado (_) o el carácter dollar (\$).
- Puede incluir, pero no comenzar por un número
- No puede incluir el carácter espacio en blanco
- Distingue entre letras mayúsculas y minúsculas
- No se pueden utilizar palabras reservadas como identificadores



Generalidades: Sintaxis básica – Palabras reservadas

abstract
assert
boolean
break
byte
case
catch
char
class
const

continue
default
do
double
else
enum
extends
final
finally
float

for
goto
if
implements
import
instanceof
int
interface
long
native

new
package
private
protected
public
return
short
static
strictfp
super

switch
synchronized
this
throw
throws
transient
try
void
volatile
while

Las palabras reservadas son identificadores predefinidos que tienen un significado para el compilador y por tanto no pueden usarse como identificadores creados por el usuario en los programas.



Generalidades: Sintaxis básica – Identificadores

Tener en cuenta:

- Para declarar **Clases**, el primer carácter va en Mayúscula y el nombre en singular
(Camisa, Factura, Persona y Cliente.)
- Los **Atributos, Objetos y Métodos** inician en minúscula para el caso de nombres compuesto, el siguiente inicia con mayúscula.
(nombre, cedula, cantidad, edad y antigüedad edadEstudiante, salarioBasico, totalNeto)
(calcularNeto(), \$informeResultados(), calcularEdad() y calcularArea())
- Los identificadores asociados a **Constantes** se escriben sus caracteres en mayúscula.
(PI, MAX_ANCHO)
- La clase la cual declara el método **main()** deberá ser de dominio público, se recomienda la declaración de una clase la cual trabaje el método main().

Lectura recomendada:

Convenciones de Código para el lenguaje de programación JAVA ([link](#))



Generalidades: Sintaxis básica – Identificadores

poder densidad m1234\$
ampNuevo 1234 abcd total
tangente val Abs computado
b34a 34ab voltios\$ a2B3
while _val Carro seno
\$seno
coseno velocidad distMax suma
return pila Balon

Detectar cuales de los siguientes son identificadores validos y cuales no



Generalidades: Sintaxis básica – Operadores

Aritméticos

Operación en Java	Operador aritmético	Expresión algebraica	Expresión en Java
Suma	+	$f+7$	<code>f + 7</code>
Resta	-	$p - c$	<code>p - c</code>
Multiplicación	*	bm	<code>b * m</code>
División	/	x/y o $\frac{x}{y}$ o $x \div y$	<code>x / y</code>
Residuo	%	$r \bmod s$	<code>r % s</code>

Operador(es)	Operación(es)	Orden de evaluación (precedencia)
* / %	Multiplicación División Residuo	Se evalúan primero. Si hay varios operadores de este tipo, se evalúan de izquierda a derecha.
+ -	Suma Resta	Se evalúan después. Si hay varios operadores de este tipo, se evalúan de izquierda a derecha.



Generalidades: Sintaxis básica – Operadores

Relacionales

Operador estándar algebraico de igualdad o relacional	Operador de igualdad o relacional de Java	Ejemplo de condición en Java	Significado de la condición en Java
<i>Operadores de igualdad</i>			
=	==	x == y	x es igual a y
≠	!=	x != y	x no es igual a y
<i>Operadores relacionales</i>			
>	>	x > y	x es mayor que y
<	<	x < y	x es menor que y
≥	>=	x >= y	x es mayor o igual que y
≤	<=	x <= y	x es menor o igual que y



Generalidades: Sintaxis básica – Operadores

Relacionales

Operador estándar algebraico de igualdad o relacional	Operador de igualdad o relacional de Java	Ejemplo de condición en Java	Significado de la condición en Java
<i>Operadores de igualdad</i>			
=	==	x == y	x es igual a y
≠	!=	x != y	x no es igual a y
<i>Operadores relacionales</i>			
>	>	x > y	x es mayor que y
<	<	x < y	x es menor que y
≥	>=	x >= y	x es mayor o igual que y
≤	<=	x <= y	x es menor o igual que y

Lógicos

OPERADOR	NOMBRE	EJEMPLO
&&	y	(7 > 2) && (2 < 4)
	o	(7 > 2) (2 < 4)
!	no	!(7 > 2)



Generalidades: Sintaxis básica – Operadores

```
int x = 4;  
int y = 5;  
int z = 10;  
int total = 12;
```

```
int x = 4, y = 5, z = 10;  
int total = x + y * z;
```

```
int x = 4, y = 5, z = 10;  
int total = z / (x * y);
```

```
System.out.println("The total is " + total + ".");
```

```
double volume = 3.14 * radius * radius * height * 1 / 3;
```



Generalidades: Sintaxis básica – Condicionales

Simples

```
1  
2  if ( condicion )  
3  {  
4  
5  }  
6  else  
7  {  
8  
9  }
```

Anidados

```
1  
2  if ( condicion 1 )  
3  {  
4      if ( condicion 2 )  
5      {  
6          if ( condicio 3 )  
7          {  
8              }  
9          }  
10     else  
11     {  
12     }  
13 }  
14
```

Switch

```
21 switch( Valor ){  
22     case Valor 1:  
23         (instrucciones);  
24     case Valor 2:  
25         (instrucciones);  
26     case Valor 3:  
27         (instrucciones);  
28     case Valor 4:  
29         (instrucciones);  
30  
31     default:  
32 }
```



Generalidades: Sintaxis básica – Condicionales

Implementación If

```
1 public class EjemploCondicional{
2     public static void main(String args[]){
3         int edadPedro=25;
4         int edadLuis=22;
5         System.out.println("BIENVENIDO AL PROGRAMA");
6
7         if (edadPedro>=18 && edadLuis>=18){
8             System.out.println("Pedro y Luis son mayores de edad");
9             System.out.println("Pedro y Luis pueden votar");
10        }
11        else if(edadPedro>=18 && edadLuis<18){
12            System.out.println("Pedro es mayor de edad,Luis no");
13            System.out.println("Pedro puede votar, Luis no");
14        }
15        else if(edadPedro<18 && edadLuis>=18){
16            System.out.println("Luis es mayor de edad, Pedro no");
17            System.out.println("Luis puede votar, Pedro no");
18        }
19        else{
20            System.out.println("Pedro y Luis no son mayores de edad");
21            System.out.println("Ni Pedro ni Luis pueden votar");
22        }
23        System.out.println("FIN DE PROGRAMA");
24    }
25 }
```

Implementación switch

```
public class EjemploSwitch{
    public static void main(String args[]){
        char operador='+';
        switch(operador){
            case '-': System.out.println("El operador es -"); break;
            case '+': System.out.println("El operador es +"); break;
            case '*': System.out.println("El operador es *"); break;
            case '/': System.out.println("El operador es /"); break;
            default: System.out.println("Operador desconocido"); break;
        }
        System.out.println("FIN DE PROGRAMA");
    }
}
```



Generalidades: Sintaxis básica – Ciclos

```
10  for(Condicion inicial; Condicion Final ; Incremento )
11  {
12      /*
13          Bloque de codigo
14      */
15  }
```

```
18  while( Condicion ){
19      //instrucciones;
20      //incremento
21  }
```

```
20  do {
21      // instrucciones
22      // incremento
23  } while( Condicion );
```

Estructuras utilizadas para ejecutar una o varias líneas de código un determinado número de veces.



Generalidades: Sintaxis básica – Ciclos

<pre> 1 /* ----- 2 Au 3 Fe 4 De 5 */ 6 //Decl 7 public 8 // 9 pu 10 11 12 13 14 15 }</pre>	<pre> 1 /* ----- 2 3 4 5 */ 6 //Decl 7 public 8 9 10 11 12 13 14 15 16 }</pre>	<pre> 1 /* ----- 2 Autor: Jairo francisco seoanes leon 3 Fecha: 29-08-2013 4 Descripcion: Ejemplo del funcionamiento del ciclo Do - while 5 */ 6 //Declaracion de la clase principal 7 public class EjemploDoWhile{ 8 public static void main(String args[]){ 9 int i=0; 10 do{ 11 System.out.println(i); 12 i++; 13 } 14 while(i<=5); 15 System.out.println("FIN DEL PROGRAMA"); 16 } 17 }</pre>
--	---	---





Generalidades: Lectura de datos

El uso de la clase **Scanner** es una de las mejores maneras de ingresar datos por teclado en Java.

- Scanner es una clase en el paquete **java.útil**
- Se utiliza para obtener la entrada de los tipos primitivos como **int**, **double** etc. y también **String**.
- Es la forma más fácil de leer datos en un programa Java

<https://docs.oracle.com/javase/8/docs/api/>



```
Nombre: Alex
Género: m
Edad: 23
Teléfono: 92578458
Promedio: 20.0
```

```
import java.util.Scanner;

public class ScannerDemo
{
    public static void main(String[] args)
    {
        // Declarar el objeto e inicializar con
        // el objeto de entrada estándar predefinido
        Scanner sc = new Scanner(System.in);
        // entrada de una cadena
        String name = sc.nextLine();
        // entrada de un carácter
        char gender = sc.next().charAt(0);
        // Entrada de datos numéricos
        // byte, short y float
        int age = sc.nextInt();
        long mobileNo = sc.nextLong();
        double average = sc.nextDouble();
        // Imprima los valores para verificar si la entrada
        // fue obtenida correctamente.
        System.out.println("Nombre: "+name);
        System.out.println("Género: "+gender);
        System.out.println("Edad: "+age);
        System.out.println("Teléfono: "+mobileNo);
        System.out.println("Promedio: "+average);
    }
}
```

Generalidades: Sintaxis básica – Ejercicios

Un programa que le permita a un trabajador saber cuál será su sueldo semanal, se sabe que si trabaja 40 horas o menos, se le pagará \$20 por hora, pero si trabaja más de 40 horas entonces las horas extras se le pagarán a \$25 por hora.

(Estructura selectiva)

Hacer un programa para una tienda de helado que da un descuento por compra a sus clientes con membresía dependiendo de su tipo, sólo existen tres tipos de membresía, tipo A, tipo B y tipo C.

Los descuentos son los siguientes: Tipo A 10% de descuento - Tipo B 15% de descuento - Tipo C 20% de descuento

El Programa debe imprimir el precio final de la compra, y el descuento realizado.

(Estructura selectiva)

Calcular la sumatoria de la serie **$1/1, 1/2, 1/3, \dots, 1/N$** , donde N es un numero definido por el usuario.

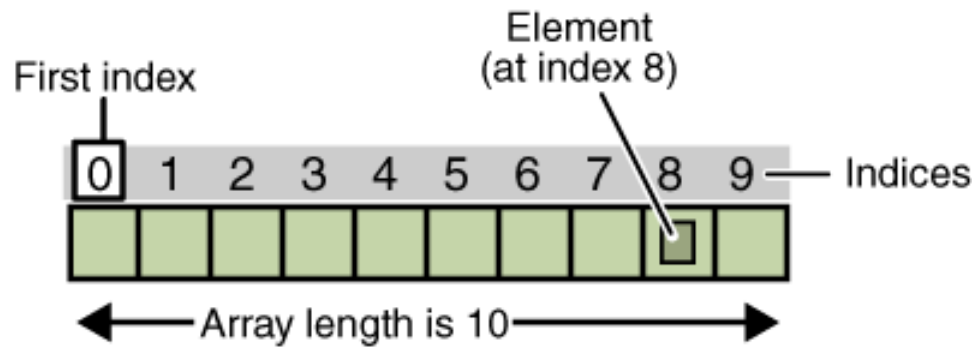
(Estructura repetitivas)

Desarrollar un programa que muestre la suma de los N primeros números pares e impares.

(Estructura repetitiva)

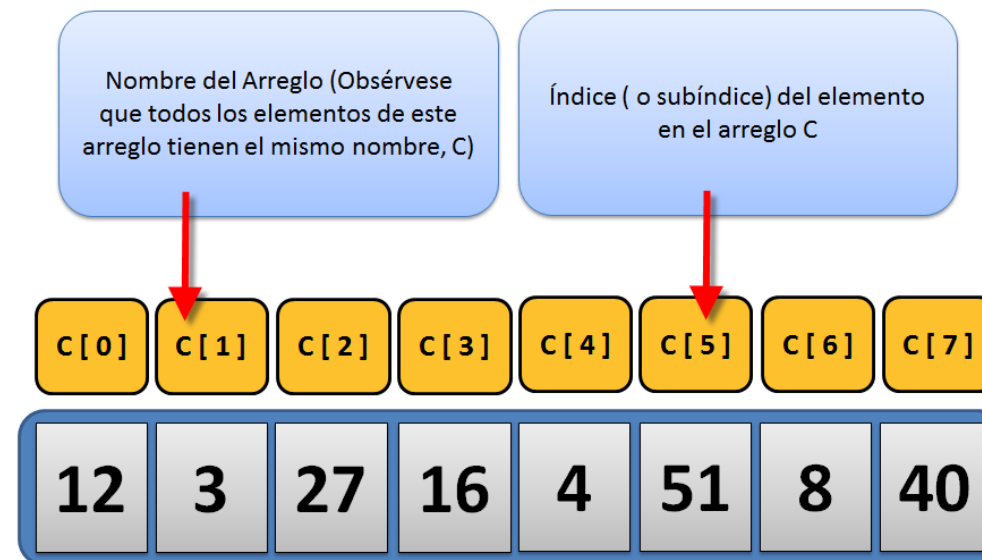


Generalidades: Sintaxis básica – Arreglos



Elemento 1,1	Elemento 1,n
Elemento 2,1	Elemento 2,n
Elemento 3,1	Elemento 3,n
.....
Elemento m,1	Elemento m,n

Un **arreglo** es un objeto contenedor que mantiene un número fijo de valores de un único tipo. La longitud de un arreglo se establece cuando el arreglo se crea.



Generalidades: Sintaxis básica – Arreglos

Declaración

```
int [] arreglo;  
arreglo = new int[10];  
  
int [] arreglo = new int[10];
```

Inicialización

```
for(int i=1; i<=arreglo.length;i++)  
{  
    arreglo[i]=i;  
}
```

```
int [] arreglo2 = {1, 2, 3, 4, 5, 6, 7, 8, 9, 0};
```



Generalidades: Sintaxis básica – Arreglos

Dado un arreglo de enteros de 10 elementos, calcular:

- a) la sumatoria de sus elementos
- b) el valor promedio de sus elementos

(Arreglos)

Dado un array o arreglo unidimensional de tamaño N , crear una programa que rellene el array o arreglo con los múltiplos de un numero M establecido por el usuario. Por ejemplo, si $N = 5$ y $M = 3$, el array resultado contendrá 3, 6, 9, 12, 15. Muéstralos por pantalla

(Arreglos)





UNIVERSIDAD
Popular del cesar