



Centro de Investigación en Computación del IPN

Java - curso básico

Dra. Erandi Castillo Montiel

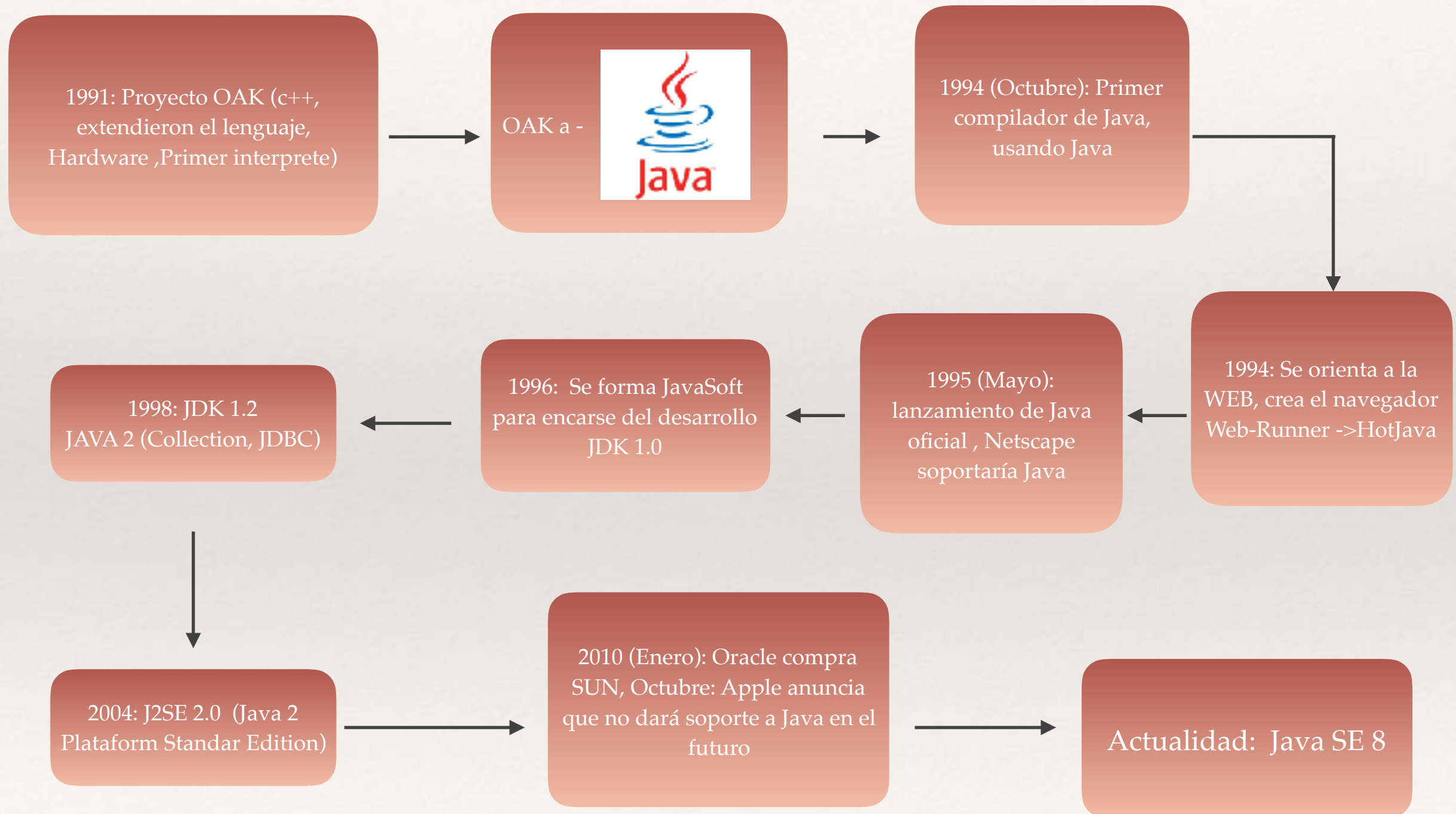
Introducción

- ❖ Historia del lenguaje
- ❖ Panorama actual
- ❖ Tipos de programas
- ❖ Obtención en instalación
- ❖ Elementos del JDK
- ❖ Variables de ambiente
- ❖ Uso general de IDE

Java

- ❖ Lenguaje de programación de propósito general, concurrente y orientado a objetos.
- ❖ Permite WORA, “write one, run anywhere”.
- ❖ Popular para las aplicaciones cliente-servidor.
- ❖ Desarrollado por James Gosling de Sun Microsystems (1995).
- ❖ Sintaxis derivada de C y C ++.

Historia del Lenguaje (I)



Características

- ❖ Orientado a objetos
- ❖ Amplio conjunto de bibliotecas
- ❖ Multiplataforma (JVM)
- ❖ Seguridad en la ejecución
- ❖ Recolector de basura

Ediciones Java

JSE

-Desarrollo de aplicaciones de gama media y estaciones de trabajo
JDK SE, JRE

JEE

-Desarrollo de aplicaciones empresariales



JME

-Desarrollo de aplicaciones para dispositivos
móviles, PDA, Microcontroladores

Instalación de JDK

www.oracle.com/technetwork/java/javase/downloads/index.html

Está en ¿Quieres traducirla?


ORACLE  Menu   Sign In  Count

Oracle Technology Network / Java / Java SE / Downloads

[Overview](#) [Downloads](#) [Documentation](#) [Community](#) [Technologies](#) [Training](#)

Java SE Downloads



DOWNLOAD 

Java Platform (JDK) 10



DOWNLOAD 

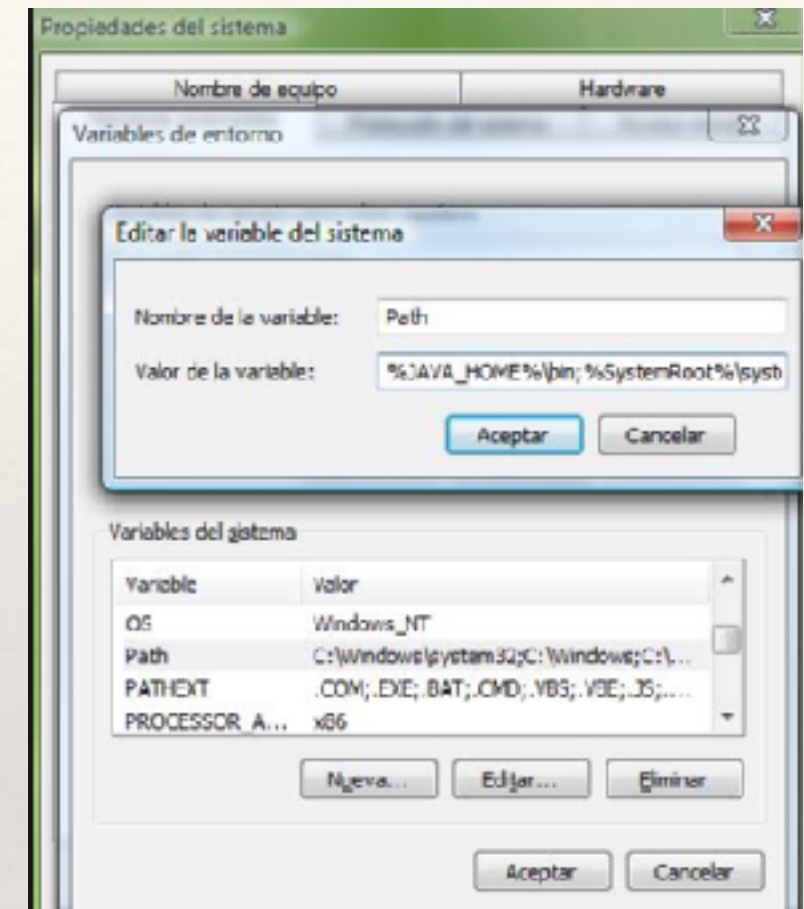
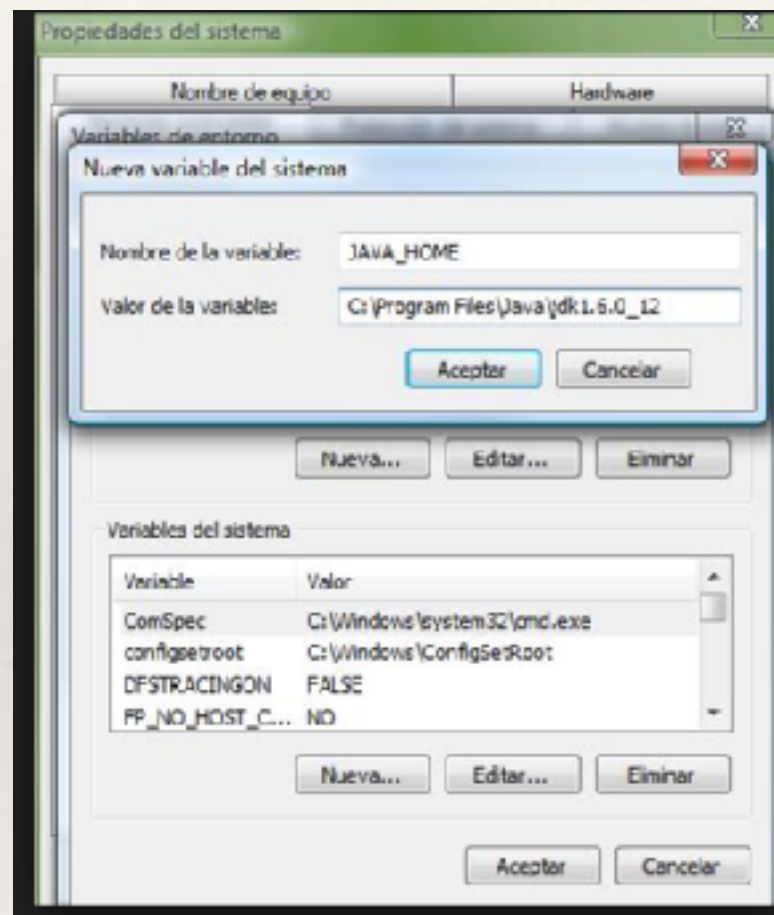
NetBeans with JDK 8

Java Platform, Standard Edition

Java SE 10
Java SE 10 is the latest feature release for the Java SE Platform
[Learn more](#) 

Variables de entorno

- ❖ JAVA_HOME = C:\Program Files\Java\jdk1.7.0_79
- ❖ PATH=%JAVA_HOME%\bin
- ❖ Test: Java-version

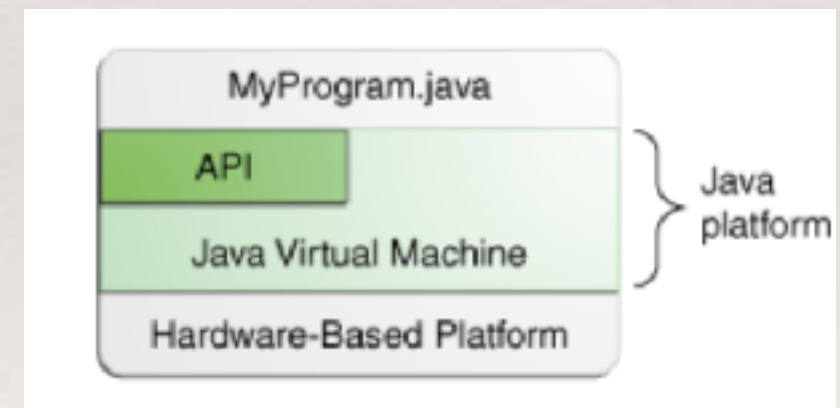
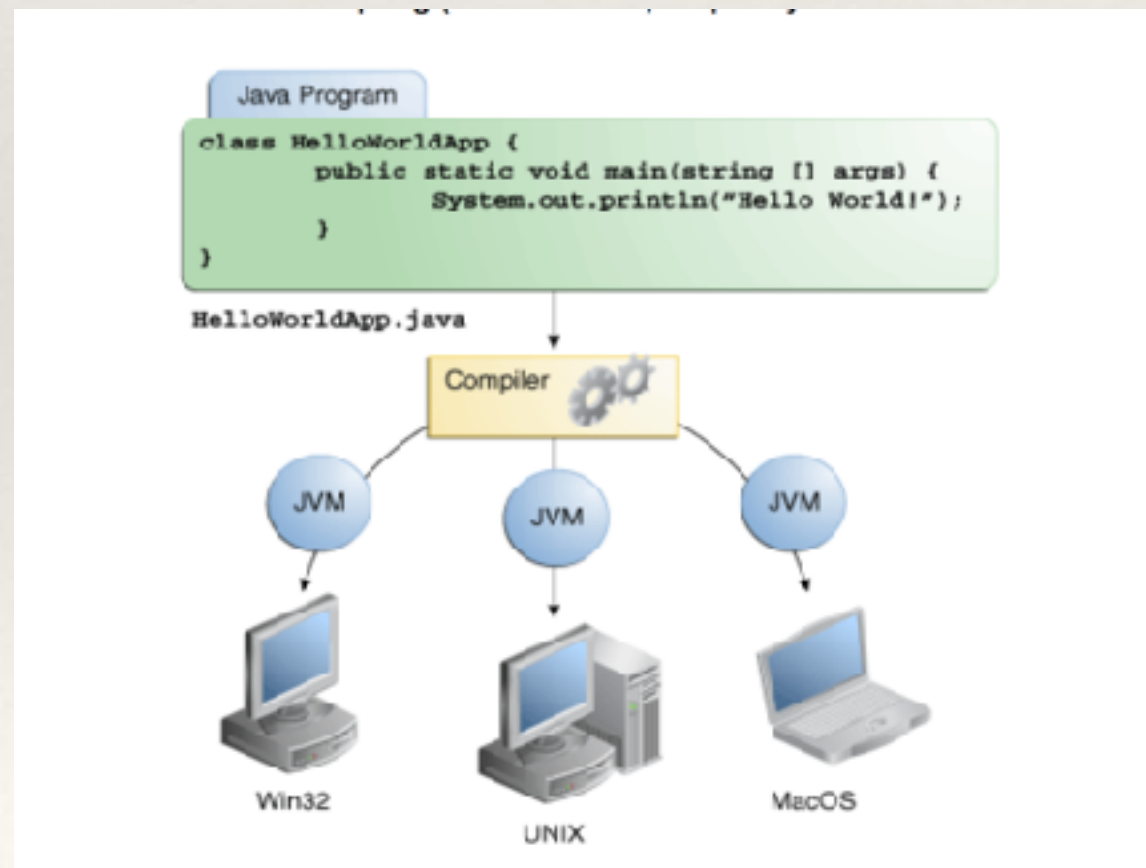
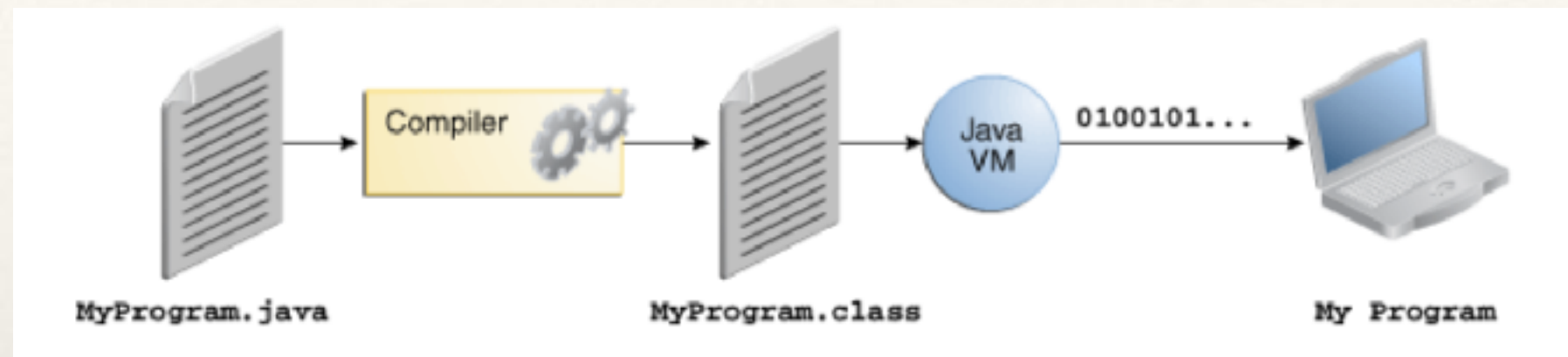


```
MacBook-Pro-de-Erandi:~ erandi$ java -version
java version "1.7.0_80"
Java(TM) SE Runtime Environment (build 1.7.0_80-b15)
Java HotSpot(TM) 64-Bit Server VM (build 24.80-b11, mixed mode)
MacBook-Pro-de-Erandi:~ erandi$
```


Hola mundo!

```
public class HolaMundo {  
    public static void main(){  
        System.out.println("Hola Mundo desde Java");  
    }  
}
```

```
[MacBook-Pro-de-Erandi:holamundo erandi$ javac HolaMundo.java  
[MacBook-Pro-de-Erandi:holamundo erandi$ java HolaMundo  
Hola Mundo desde Java  
MacBook-Pro-de-Erandi:holamundo erandi$ ]
```



Download Netbeans

https://netbeans.org/downloads/

inglés ¿Quieres traducirla? No Traducir Nunca traducir inglés

Choose page language ▶

NetBeans NetBeans IDE NetBeans Platform Plugins Docs & Support Community Partners Search

HOME / Download

NetBeans IDE 8.2 Download

8.1 | 8.2 | Development | Archive

Email address (optional):

Subscribe to newsletters: ☒ Monthly ☐ Weekly

☒ NetBeans can contact me at this address

IDE Language: English Platform: Mac OS X

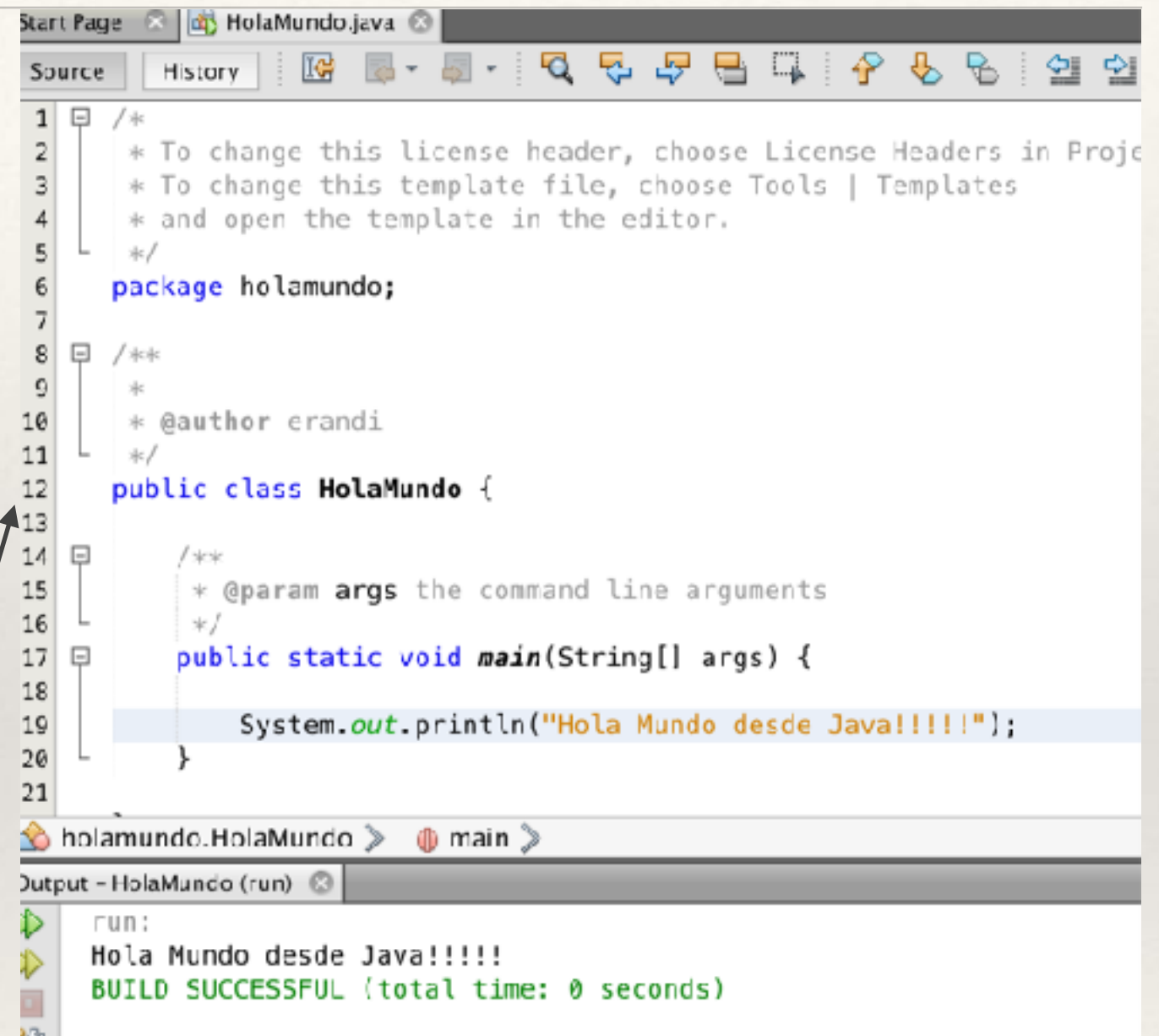
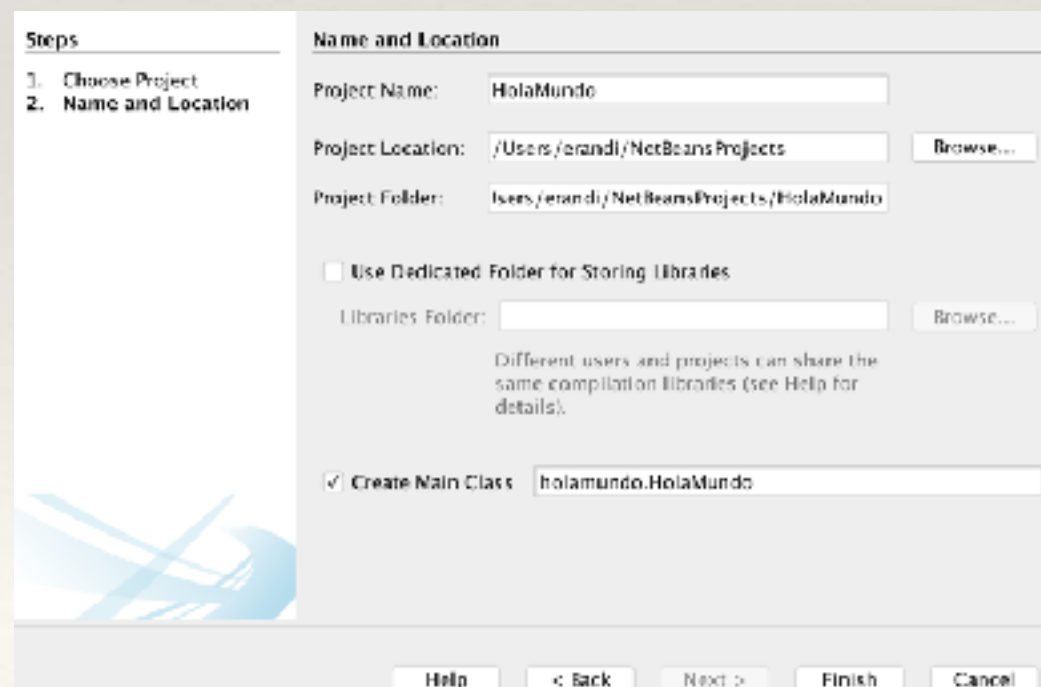
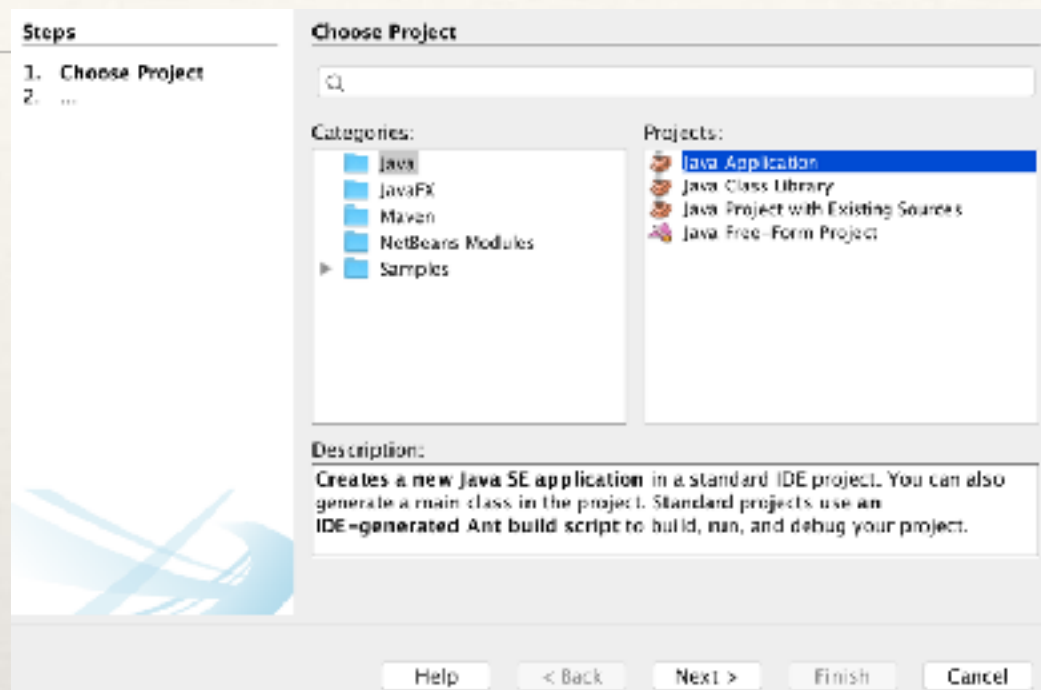
Note: Greyed out technologies are not supported for this platform.

NetBeans IDE Download Bundles

Supported technologies *	Java SE	Java EE	HTML5/JavaScript	PHP	C/C++	All
NetBeans Platform SDK	•	•				•
Java SE	•	•				•
Java FX	•	•				•
Java EE		•				•
Java ME						—
HTML5/JavaScript		•	•	•		•
PHP			•	•		•
C/C++					•	•
Groovy						•
Java Card™ 3 Connected						—
Bundled servers						
GlassFish Server Open Source Edition 4.1.1		•				•
Apache Tomcat 8.0.27		•				•

Download Download Download Download Download Download

Hola Mundo desde NetBeans



Java-Conceptos básico de POO

Un perro

- Características:
 - Color, nombre, raza, edad.
- Comportamiento:
 - Ladra, va al baño, mueve la cola.
- Un objeto de software
 - Almacena estados en campos (variables)
 - Expone un comportamiento por medio de métodos

Clases (I)

- Contiene la definición de los objetos
 - Atributos
 - Métodos que va a exponer

Sintaxis:

```
[public] class NombreClase{  
    // declaración de atributos  
    // declaración de métodos  
}
```

Clases (II)

- Atributos

- Variables que va a almacenar las características de los objetos

Sintaxis:

[private] tipo nombre del atributo;

- Métodos

- Implementados mediante funciones, definen el comportamiento de los objetos de una clase

[public] tipoRetorno nombreMetodo
(argumento, ..){

return valor;

}

Clase (III): MiClase.java

```
public class MiClase{  
    private tipo atributo1;  
    private tipo atributo2;  
    public tipo miMetodo1(tipo arg1, tipo arg2, ..){  
        return valor;  
    }  
    public tipo miMetodo2(tipo arg1, tipo arg2, ..){  
        return valor2;  
    }  
}
```

Comentarios

// Comentarios de una línea

/* Comentarios de

varias líneas*/

/**

* comentarios para documentación

*

*/

Los comentarios ayudan al desarrollador a entender el comportamiento u objetivos de una clase.

Revisar JavaCode Conventions

Sentencias de escape

Secuencia	Significado
\b	Retroceso
\n	Salto de línea
\t	Tabulación Horizontal
\\	Barra \
\'	Comilla simple
\"	Comilla doble

Tipos primitivos

Tipo básico	Tamaño en memoria
byte	8 bits
short	16 bits
int	32 bits
long	64 bits
char	16 bits
float	32 bits
double	64 bits
boolean	-

Declaración de variables

- Deben iniciar con un carácter alfabético.
- No puede contener espacios, signos de puntuación o secuencias de escape.
- No pueden usarse palabras reservadas como nombres de variables.
- El nombre debe ser relacionado con el contenido de la variable.
- Preferible usar una definición por variable.

Sintaxis:

tipo nombreVariable;

int k,p;

long v1;

char car_2;

NO VALIDAS:

int num uno;

log class;

boolean 7q;

Palabras reservadas

abstract	default	if	private	this
boolean	do	implements	protected	throw
break	double	import	public	throws
byte	else	instanceof	return	transient
case	extends	int	short	try
catch	final	interface	static	void
char	finally	long	strictfp	volatile
class	float	native	super	while
const	for	new	switch	assert
continue	goto	package	synchronized	enum

Ejercicio: Clase televisión

- Crear una clase que defina una televisión
 - Definir sus atributos
 - Definir sus clases

```
public class Television{
    private boolean encendido;
    private int canal;
    private int volumen;
    private String marca;

    public setMarca (String nuevaMarca){
        marca = nuevaMarca;
    }

    public void cambiarCanal (int vol){
        volumen = volumen+vol;
        System.out.println("volumen : " +
            volumen);
    }

}
```

```
public void cambiarCanal (int nuevoCanal){
    canal = nuevoCanal;
    System.out.println("canal : " + canal);
}

public void imprimeMarca (String marca){
    System.out.println("marca de TV: " + marca);
}

public void encender(boolean enc){
    encendido=enc;
    System.out.println("El televisor esta : "+
        enc);
}
```

Creación de objetos

❖ Sintaxis:

```
TipoObjeto nombreObjeto = new TipoObjeto();
```

Use la clase televisión e implemente un objeto de tipo Televisión y opere el objeto.


```

package ejercicio2;

/**
 *
 * @author erandi
 */
public class Ejercicio2 {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        Television sony = new Television();
        sony.estado();
        sony.encender(true);
        sony.estado();
        sony.encender(false);
        sony.estado();
    }

}

```

```

public class Television {

    private boolean encendido;
    private int canal;
    private int volumen;
    private String marca;

    public void estado(){
        System.out.println("El estado de la TV es : " + encendido);
    }

    public void encender(boolean enc) {
        encendido = enc;
        System.out.println("El televisor esta: " + enc);
    }

}

```

Utilizar los nuevos métodos

```
public class Ejercicio2 {  
  
    public static void main(String[] args) {  
        Television sony = new Television();  
        sony.estado();  
        sony.encender(true);  
        sony.estado();  
        sony.encender(false);  
        sony.estado();  
        sony.cambiarCanal(3);  
        sony.cambiarVolumen(100);  
        sony.setMarca("sony");  
        sony.imprimeMarca();  
    }  
}
```

Implementar los otros métodos de la clase Televisión

```
public void setMarca(String nuevaMarca) {  
    marca = nuevaMarca;  
}  
  
public void cambiarVolumen(int vol) {  
    volumen = volumen + vol;  
    System.out.println("volumen: " + volumen);  
}  
  
public void cambiarCanal(int nuevoCanal) {  
    canal = nuevoCanal;  
    System.out.println("canal: " + canal);  
}  
  
public void imprimeMarca() {  
    System.out.println("marca de TV: " + marca);  
}
```

Asignación de literales

❖ Sintaxis

`variable = expresión;`

Donde la “expresión” se refiere a cualquier expresión Java que regrese un valor compatible con el tipo de dato de la variable.

`int p, q, r;`

`p=10;`

`q=p+5;`

`r=q*8;`

Ámbitos de variables

- Atributos: Están presentes en la vida del objeto, pueden ser utilizadas sin haber sido inicializadas de forma explícita.
- Variable local: Variables declaradas dentro de los métodos de una clase, sólo están presentes en el periodo de ejecución de la función donde se crearon.

Operadores aritméticos

Operador	Descripción	Ejemplo
+	Suma dos valores numéricos	<code>c=2+5;</code>
-	Resta	<code>c= 2-5;</code>
*	Multiplicación de dos números	<code>c= 2*5;</code>
/	Divide dos números:	
%	Módulo: calcula el resto de la división	<code>c=3%2; // resultado 1</code>
++	Incrementa una variable en 1 y asigna el resultado a la misma variable	<code>c=3;</code> <code>c++; // resultado 4</code>
--	Decrementa la variable en 1 unidad	<code>c--; //resultado 3</code>
+	Se usa para concatenar 2 cadenas.	

Operadores de asignación

Operador	Descripción	Ejemplo
=	Asigna la expresión de la derecha al operando de la izquierda	
+=	Suma la expresión de la derecha y la asigna a la variable de la izquierda	<code>c=3;</code> <code>c+=5; //resultado 8</code>
-=	Resta	<code>c=2;</code> <code>c-=3; // resultado -1</code>
=	Multiplica	<code>c=5;</code> <code>c=2; // equivale c=c*2;</code>
/=	Divide	<code>c=6;</code> <code>c/=3; // equivale c=c/3;</code>
%=	Módulo	<code>c%=2; //equivale c=c%2;</code>

Ejercicio

- Crear una clase llamada Operadores
 - Implementar operaciones para dos números enteros
 - Suma
 - Resta
 - Multiplicación
 - División

Métodos estáticos

- No están asociados a objetos, por lo que no es necesario crear un objeto para invocarlos.
- También llamados Métodos de la clase.
- No pueden hacer referencia a atributos no estáticos dentro de la clase.
- No pueden hacer uso de *super* y *this*

Sintaxis:

```
public static tipoRetorno nombreMetodo (tipo arg1, ...){  
    return variable;  
}
```

Métodos estáticos

- Métodos que pueden ser invocados sin necesidad de crear un objeto particular de la clase.

Sintaxis:

```
static tipoRetorno nombreMetodo( parametros){  
  
}
```

Ejercicio: crear una clase que implemente la suma, resta, multiplicación y división de tipo entero con funciones estáticas.

```
public class Operaciones {  
    public static void main(String[] args) {  
        float a = 0.5f;  
        float b = 0.4f;  
        float suma;  
        float resta;  
        float division;  
        float multiplicacion;  
        suma = a + b;  
        System.out.println("suma " + suma);  
  
        resta = a - b;  
        System.out.println("resta " + resta);  
        division = a / b;  
        System.out.println("Division " + division);  
        multiplicacion = a * b;  
        System.out.println("Multiplicacion " + multiplicacion);  
    }  
}
```

Asignación y referencia de valores

```
int a = 8;
```

```
int b = a;
```

```
String r = "Hola";
```

```
String s=r;
```

Operadores condicionales

Operador	Descripción
==	Compara dos valores, true: si son iguales, false: en caso contrario
<	Si la expresión de la izquierda es menor que la de la derecha. true.
>	Si la expresión de la izquierda es mayor que la derecha, true.
<=	Si la exp. izquierda es menor o igual a la derecha, true.
>=	Si la exp. izquierda es mayor o igual a la derecha, true.
!=	Si las expresiones son diferentes el resultado es true.

Sentencias de control

Sentencia *if*: permite comprobar una condición en la secuencia de un método

```
if (condicion){  
  
    //sentencias  
  
} else{  
  
    //sentencias  
  
}
```

```
if (a>b){  
    System.out.println( "el mayor es a" );  
}  
else{  
    System.out.println( "el mayor es b" );  
}
```

Entrada desde el teclado

```
package cjava;

import java.util.Scanner;

/**
 *
 * @author erandi
 */
public class Cjava {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int i = sc.nextInt();
        System.out.println("El número es : " + i);
    }
}
```

Ejercicio

Desarrollar un método estático que tome como argumento un entero y lo imprima en pantalla si es un número par o impar


```
public class Cjava {  
  
    public static void main(String[] args) {  
        Scanner sc = new Scanner(System.in);  
        int i = sc.nextInt();  
        System.out.println("El número es : " + i);  
        paroirpar(i);  
    }  
  
    public static void paroirpar(int numero) {  
        if (numero % 2 == 0) {  
            System.out.println("el numero es par");  
        } else {  
            System.out.println("el numero es impar");  
        }  
    }  
}
```

Ejercicio

Desarrollar un método que reciba como entrada tres enteros y devuelva el mayor de los 3.

- Los enteros deben ser capturados desde pantalla

Igualdad de objetos

- Los operadores “==” y “!=” se utilizan para comparar variables de un tipo de dato compatible.
- Al usarse en variables de tipo OBJETO, lo que compara son referencias NO EL OBJETO EN SI.

Operadores lógicos

Operador	Descripción
&&	Operador AND, true, si ambos lados del operador son true.
	Operador OR, true, si al menos uno de los operandos es true
!	Operador NOT, da como resultado el valor contrario de la expresión booleana al que se aplica.

Ejercicio

- ❖ Genere un programa en el cual, se capture un número desde el teclado entre 1 y 10.
- ❖ Se compare contra un número aleatorio.
- ❖ De ser iguales, imprima en pantalla un mensaje “ganador”.
- ❖ De ser diferentes, imprima “SUERTE PARA LA PROXIMA”.

```
package aleatorio;

import java.util.Random;
import java.util.Scanner;

public class Aleatorio {

    public static void main(String[] args) {
        Scanner keyboard = new Scanner(System.in);
        System.out.println("De un número entre 0 y 10: ");
        int inputNumber = keyboard.nextInt();
        Random grand = new Random();
        int randomNumber= grand.nextInt(10)+1;

        if(inputNumber == randomNumber){
            System.out.println("*****");
            System.out.println("****Ganador****");
            System.out.println("*****");
        }else{
            System.out.println("Suerte para la próxima");
            System.out.println("El número fue : "+randomNumber+ " .");
        }
        System.out.println("Gracias por jugar!!!");
    }
}
```

Operador *instanceOf*

Sintaxis:

objeto *instanceOf* Clase

Ejemplo:

```
String p = new String("Algo inteligente");  
if(p instanceOf String){  
    System.out.println("Es una cadena");  
}
```

Operador condicional ?

Sintaxis:

```
variable = (condición)? valor_si_true: valor_si_false
```

Ejemplo:

```
int 4;
```

```
String s=(k%2==0)?"par":"impar";
```

```
System.out.println (s);
```

Instrucción *switch*

Usada en caso de tener múltiples alternativas

Ejecuta diferentes bloques de instrucciones

Sintaxis:

```
switch (expresión){  
    case valo1:  
        sentencias;  
        break;  
    case valo2:  
        sentencias;  
        break;  
}
```

Sólo se pueden evaluar valores primitivos enteros.

No puede haber dos case con el mismo valor.

Instrucción *for*

- ❖ Permite ejecutar un conjunto de sentencias, un determinado número de veces:

Sintaxis:

```
for(inicialización, condición_paro; incremento){  
    sentencias;  
}
```

Ejercicio

Use su clase de Operaciones: genere una función que reciba como argumento un entero y devuelva el valor factorial de dicho número.

Use la operación creada para imprimir el factorial de un número en pantalla.

Instrucción *while*

- Ejecuta un bloque de sentencias, mientras se cumpla una determinada condición.
- Al igual que el *for*, se utilizan llaves para delimitar el conjunto de sentencias.

Sintaxis:

```
while (condición){  
sentencias;  
}
```

```
do{  
sentencias;  
}
```

```
int a =10;
```

```
while (a>0){  
    System.out.println("El número es :"+ a);  
    a- -;  
}
```

Ejercicio

- ❖ Implemente una función que reciba como parámetro un entero, después imprima la tabla de multiplicar del argumento.
- ❖ Implemente con `while`.
- ❖ Implemente con `do..while`.

Rompiendo el bucle

break;

Su uso puede extenderse para provocar la salida forzada de un ciclo o bucle.

```
public static void aleatorio2() {  
    Random rand = new Random();  
    int numero = 6;  
    boolean acierto = false;  
    for (int i = 0; i < 100; i++) {  
        int bandera = rand.nextInt(10);  
        if (bandera == numero) {  
            acierto = true;  
            System.out.println("Acerto " +  
                break;  
        }  
        System.out.println("No acerto");  
    }  
}
```

Rompiendo el bucle (II)

`continue;`

Provoca que el ciclo detenga la iteración actual, si se usa en un `for`, realiza el incremento y si se usa el `while`, probara la condición de entrada.

Ejercicio

- Desarrollar un programa que, dados dos números cualquiera, muestre en pantalla la suma de todos los números pares comprendidos entre ellos.
- Deberá implementarse una función que maneje la operación, tomando como argumentos los dos enteros y devolviendo el valor de la suma de los números pares, en la definición de la clase que incluye el main.
- La clase que implemente la función main, deberá involucrar la clase creada e imprimir el resultado.

```
public static void main(String[] args) {  
    Scanner numero1 = new Scanner(System.in);  
    Scanner numero2 = new Scanner(System.in);  
    System.out.println("Dame el primer numero : ");  
  
    int a = numero1.nextInt();  
    System.out.println("Dame el segundo numero : ");  
    int b = numero2.nextInt();  
    int total=sumaPares(a,b);  
    System.out.println("total suma: "+ total);  
}
```

```
public static int sumaPares(int a , int b){  
    int suma=0;  
    for(int i=a;i<=b;i++){  
        if(i%2==0){  
            suma=suma+i;  
            System.out.println("suma "+ suma);  
        }  
    }  
    return suma;  
}
```

Arreglos

Un arreglo o Array, es un objeto en el que se pueden agrupar un número de objetos o datos primitivos , del mismo tipo.

Cada uno de los objetos dentro el array, tiene un identificador numérico único asignado dependiendo de sus posición en el arreglo.

Sintaxis:

```
tipo [] variable_arreglo;  
tipo variable_arreglo[];
```

Ejemplo:

```
int [] arrI;  
String [] arrS;  
float arrf[];
```

Dimensionando un arreglo

Consiste en crear un arreglo con el tamaño que tendrá

`variable_array= new tipo[tamaño];`

*Los índices de un arreglo se encuentran entre 0 y tamaño-1

`arrI.length;`

`arrI = new int [10];`

`arrS = new String[5];`

`arrf= new float[10];`

```
int numeros [] = new int [10];  
  
for (int i =0; i<numeros.length; i++){  
    numeros[i]=i*2;  
    System.out.println(numeros[i]);  
}
```

Array como argumento

Sintaxis:

Como argumento:

```
tipo_retorno nombreMetodo (tipo []arreglo){  
    // sentencias  
}
```

Retorno de un arreglo:

```
tipo [ ] nombreMetodo (){  
    return arreglo;  
}
```

```
public static void main(String[] args) {  
    int[] arreglo = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};  
    imprimeNumeros(arreglo);  
    cambiaNumeros(arreglo);  
    imprimeNumeros(arreglo);  
  
}  
  
public static void cambiaNumeros(int[] array) {  
    for (int i = 0; i < array.length; i++) {  
        array[i] = array[i] * 2;  
    }  
}  
  
public static void imprimeNumeros(int[] array) {  
    for (int i = 0; i < array.length; i++) {  
        System.out.print("valor: " + array[i] + " ");  
    }  
    System.out.println("");  
}
```

Ejercicio

Cree un programa que genere un arreglo de 10 elementos.

Inicialice los elementos con un número aleatorio.

Sume el contenido del arreglo.

Imprima el mayor y el menor de todos los números dentro del arreglo.

Sentencia *for each*

Sintaxis:

```
for (tipo variable: array){  
    // sentencias  
}
```

Ejemplo

```
for (int n: arrayI){  
    System.out.println(n);  
}
```

Ordenamiento de burbuja

metodoBurbuja ($a_0, a_1, a_2, \dots, a_{n-1}$)

para $i=0$ hasta n hacer

para $j=0$ hasta $n-1$ hacer

si $a(j) > a(j+1)$ entonces

cambiarlos de lugar

fin si

fin for

fin for

fin método