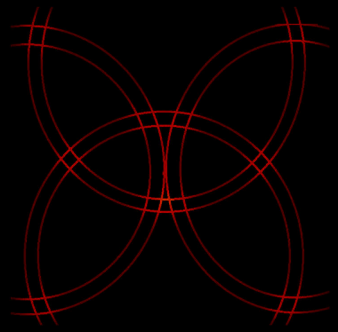
Laboratorio de Investigaciones de Artefactos del Principio y Fin.

L.I.A.P.F.



Divergencia del CAOS

Lenguajes de programación

Filosofía de Java

1. Debería usar la metodología de la programación orientada a objetos.

2. Debería de permitir la ejecución de un mismo programa en múltiples sistemas
.
3. Deberá de incluir por defecto, trabajo en red.

4. Debería de diseñarse para ejecutar código de manera remota y segura.

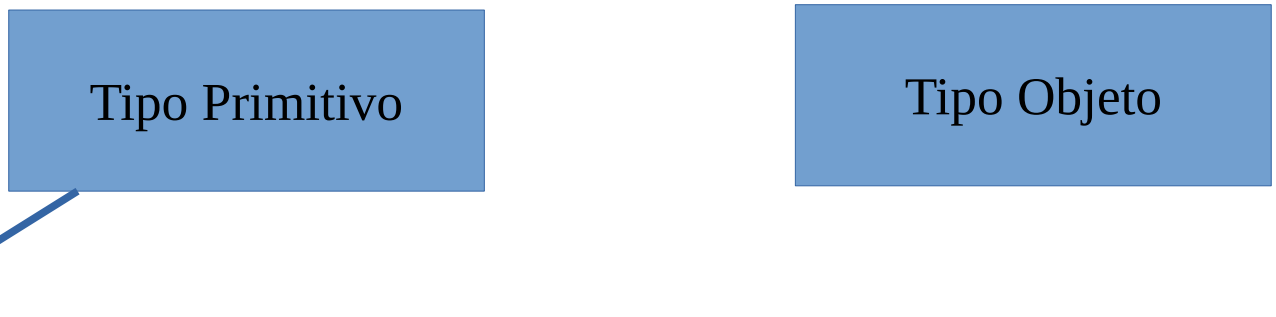
5. Debería ser fácil de usar y tomar lo mejor de otros lenguajes orientados a objetos.

Hola Mundo

```
public class HolaMundo {

    public static void main(String[] args) {
            System.out.println(“Hola Mundo :D”);
    }
}
```

Tipos de Datos

Definición: Un espacio de memoria al que le asignamos un contenido, puede ser un valor numérico, de tipo carácter o cadena de carácter.



Tipos Enteros:

byte: Rango de -128 a 127, ocupa 1 byte

Short: Rango de -32,768, ocupa 2 bytes

Int: Rango de -2,147,483,648 to -2,147,483,647, ocupa 4 bytes

Long: Rango de -9,223,372,036,854,775,808 to -9,223,372,036,854,775,807, ocupa 8 bytes

Datos tipos flotante:

float: Rango de 1.40129846432481707e-45 to 3.40282346638528860e+38, 4 bytes

Ejemplo: float variable = 1.258F; //siempre se pone la F en las variables float

double: Rando de 4.94065645841246544e-324d to 1.79769313486231570e+308d, 8 bytes

Ejemplo: double variable = 1.58; //NO es necesario poner la F

Datos tipos texto:

Char: Rango Unicode, 2 bytes

Ejemplo: char carácter = 'a';

Datos tipos logicos:

boolean : Rango true o false, 1 bit;

Ejemplo: boolean variable = true;

Normas de Java

.Java sigue la siguiente convención para nombrar variables

-Es sensible al uso de las mayúsculas y minúsculas

int a - No es igual a - int A

-Debe de comenzar con una letra, se permite usar $ y "_"

int a = 0; int $b = 0; //valido

-Las letras posteriores pueden ser letras, números, $ y "_"

int n$_123 = 4; //valido

-Por convención se debe de usar la técnica del "camello" (Buena practica)

int numeroAzar = 5656; //Buena practica

-También por convención las constantes se escriben con mayúsculas y contienen "_" (Buena practica)

final char HOLA = 'c'; final int VALOR_MAXIMO = 10;

CAST

En la programación hay situaciones en las que se necesita cambiar el tipo de dato.

Un Cast es una operación en Java que :

-Da como resultado una variable con un tipo de dato diferente al inicial

-Puede usarse con datos primitivos, instancias de una clase y objetos

Ejemplo : double d = 10;

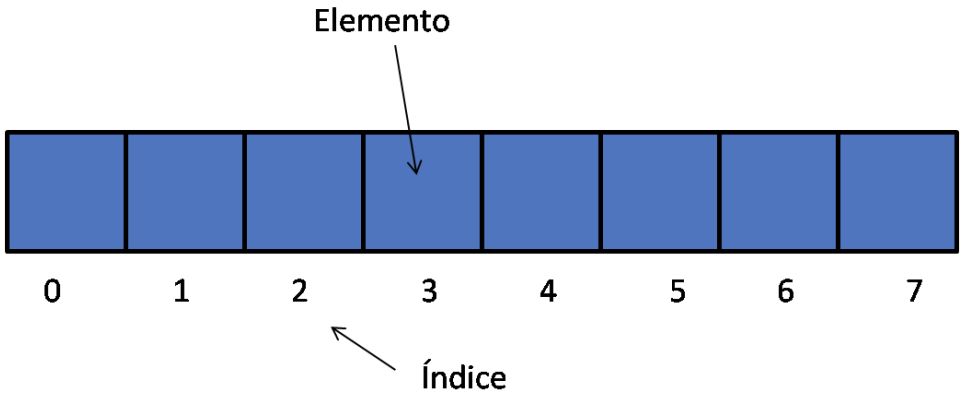
Int i = (int) d; //Cambiar el dato double a entero

-Se puede realizar el cast a todos los tipos de datos primitivos, excepto boolean.

-A menudo, el tipo de cast se realiza en situaciones donde el tipo de resultado es más grande que su tipo original.

-Por lo tanto, a menudo se puede usar un byte o char como un int, un int como un long, float o double.

Arrays



-Los arreglos se pueden definir como objetos en los que podemos guardar más de una variable.

Ejemplo:

int conjuntoEnteros = **new int**[8];

.La estructura de declaración de un arreglo es el siguiente:

tipo_dato[ ] nombre_variable;

tipo_dato nombre_variable[ ];

.Para signar a un arreglo su tamaño o capacidad, se hace de la siguiente manera:

arreglo = new tipo_dato[capacidad];

Arrays [Asignar valores]

Una vez se tiene declarado el arreglo, y al mismo se le ha asignado un tamaño o capacidad, podemos acceder a los datos dentro del mismo y asignarles valores.

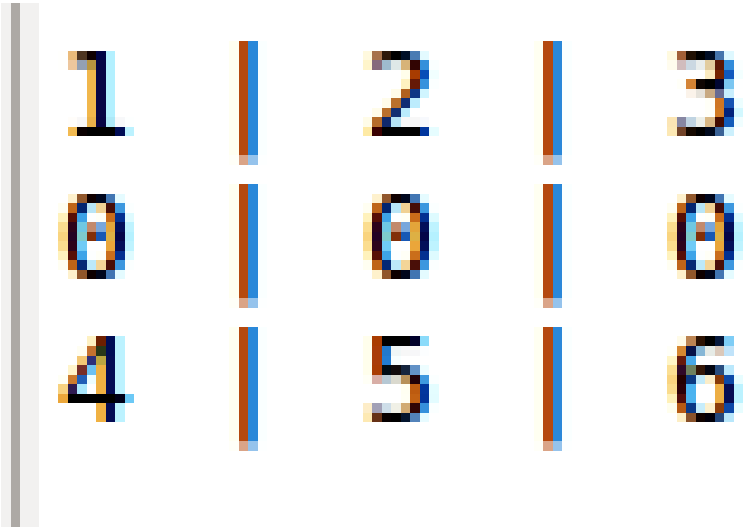
arreglo[indice] = valor;

```
int[] arreglo = new int[3];

arreglo[0] = 1;
arreglo[1] = 2;
arreglo[2] = 3;

System.out.println("Arreglo:  |" + arreglo[0] + "|" + arreglo[1] + "|" + arreglo[2] + "|");
```

```
Arreglo:  |1|2|3|
```

Arrays 2D (de dos dimensiones)

```
int[][] conjuntoEnteros2D = new int[3][3];
conjuntoEnteros2D[0][0] = 1;
conjuntoEnteros2D[0][1] = 2;
conjuntoEnteros2D[0][2] = 3;
conjuntoEnteros2D[2][0] = 4;
conjuntoEnteros2D[2][1] = 5;
conjuntoEnteros2D[2][2] = 6;
```




```
System.out.println(conjuntoEnteros2D[0][0] + " | " + conjuntoEnteros2D[0][1] + " | " +
conjuntoEnteros2D[0][2]);
System.out.println(conjuntoEnteros2D[1][0] + " | " + conjuntoEnteros2D[1][1] + " | " +
conjuntoEnteros2D[1][2]);
System.out.println(conjuntoEnteros2D[2][0] + " | " + conjuntoEnteros2D[2][1] + " | " +
conjuntoEnteros2D[2][2]);
```

Operadores

Una vez que el código fuente de Java tiene variables, las podemos usar para crear y formar expresiones que regresen valores

Operadores Aritméticos: Son los símbolos que se usan para realizar aritmética básica en el lenguaje de programación Java (Suma, resta, multiplicación, etc).

Concatenación de cadenas

-El perador + puede usarse para agregar o concatenar cadenas.

-Unión de dos elementos.

```
System.out.println("El numero es:   " + numero );
```

Operadores de Asignación: + += -= /= %=

X += 2; Equivalente X = X + 2;

Operadores de incremento y decremento:

. Incremento: Se usan para agregar un 1 al valor de la expresión (**++**)

i++ **es igual a** i = i + 1 **y** ++i **es igual a** i + 1 = i

.Decremento: Se usan para substraer un 1 del valor de la expresión (**--**)

i-- **es igual a** i = i – 1 **y** ++i **es igual a** i + 1 = i

Equidad y operaciones relacionales

. Todas las expresiones creadas con equidad y operadores relacionales regresaran un valor booleano, dependiendo si la comparación se realiza o no

. Hace uso de dos operadores, uno en cada lado del operador

. Los operadores de equidad se describen a continuación:

Operadores relacionales		**Operadores lógicos**	
igualdad	==	**and**	& (ampersand)
desigualdad	~=	**or**	| (pipe)
menor	<	**not**	~ (tilde)
mayor	>	**xor**	o exclusivo
menor o igual	<=	**any**	True si **algún** elemento del vector es true
mayor o igual	>=	**all**	True si **todos** los elementos del vector son true

Control de flujo

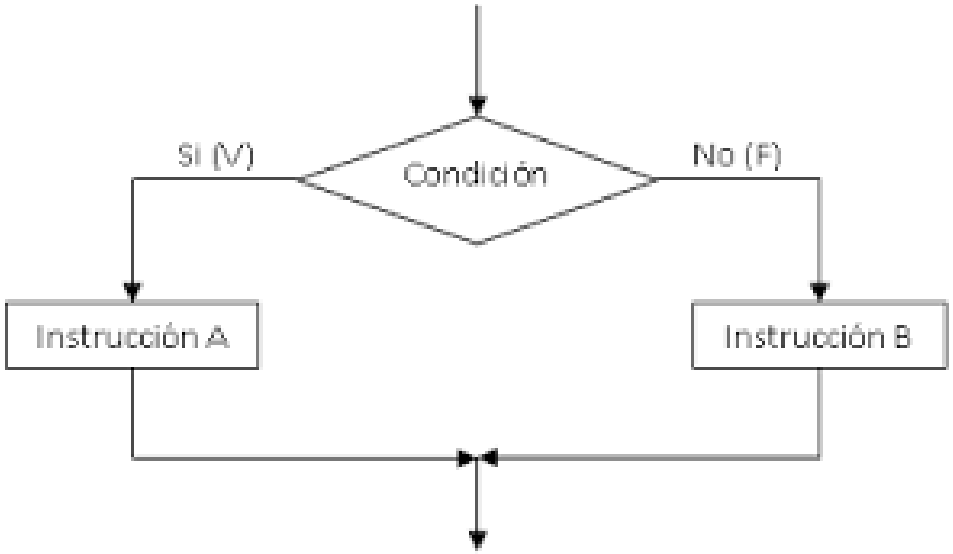
Las sentencias de código en Java son ejecutadas secuencial mente desde arriba hasta abajo en el orden que van apareciendo.
Sin embargo podemos controlar el control de flujo usando sentencias condicionales, ciclos, etc.

If / Else

. Una condición es una expresión booleana

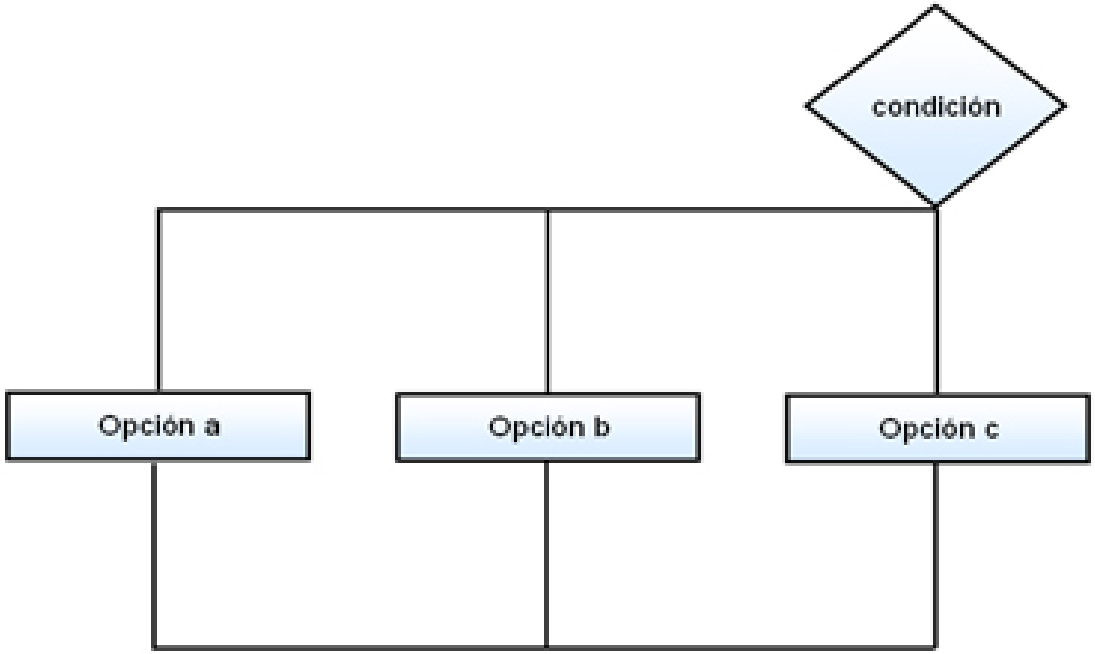
. La sentencia se ejecuta solamente si la expresión booleana es verdadera

```
if (Condición)  {

        instrucciones
}  else  {

        instrucciones
}
```



Switch

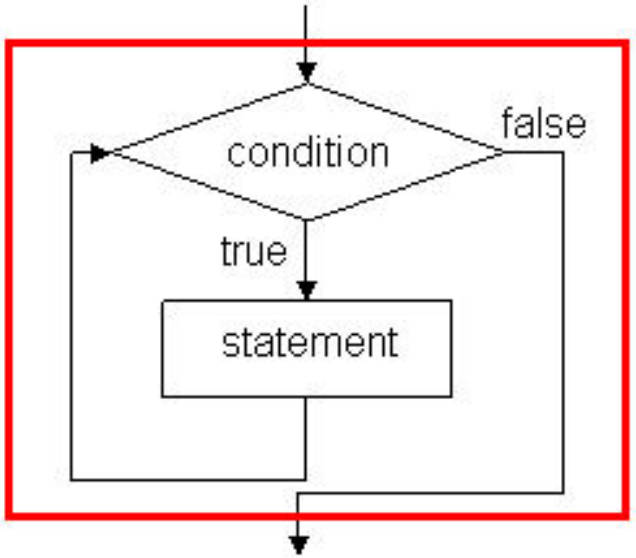
. A diferencia de la sentencia if / else, la sentencia switch puede tener un número de posibles rutas de ejecución.



Ciclo While

. Se ejecuta continuamente un bloque de código mientras una condición particular, sea cierta. Su sintaxis se puede expresar cómo:

```
while  (condición)  {
        //Instrucciones
}
```



Ciclo For

. La sentencia proporciona una forma compacta para iterar sobre un rango de valores.

```
for (inicializa; fin-condición; incremento) {
        //Instrucciones
}
```

```
int[] conjuntoEnteros = new int[3]; //arreglos

for(int i = 0; i < conjuntoEnteros.length; i++) //ciclo for
{
    conjuntoEnteros[i] = 1 + i;
    System.out.println("indice [ " + i + " ]: " + conjuntoEnteros[i]);
}
```

```
indice [ 0 ]: 1
indice [ 1 ]: 2
indice [ 2 ]: 3
```

Ciclo For extendido (foreach)

. Es más fácil para recorrer colecciones de datos sin necesidad de conocer o definir el número de elementos a recorrer.

```
For  ( Tipo-Elemento : Colección)  {
        //Instrucciones
}
```

Programación orientada a objetos (POO)

. Se trata de descomponer un problema en subproblemas y más subproblemas

.Definir un Dominio del Problema (PROBLEM DOMAIN)

.Recopilación de requisitos del cliente y tener por escrito un enlace

.Fijarnos en el escenario del problema y tratar de simularlo con objetos

Identificar objetos

. Pueden ser físicos o conceptuales

.Los objetos tienen atributos (características)
.tamaño
.nombre
.forma
.representación del objeto

. Los objetos tienen operaciones (las cosas que puede hacer el objeto)

. Los nombres de los objetos por lo general son Sustantivos:
Cuenta, cliente

.Los atrivutos de los objetos también

. Las operaciones suelen ser Verbos o sustantivo y verbo
Mostrar, Enviar pedido

.Vehiculo

Atributos:
- . matricula
- . marca
- . modelo
- . año

Comportamiento:
- . arrancar
- . frenar
- . reversa

Objeto y Clase

. Diseñando un modelo de Clase

.Una Clase es la forma en como defines un objeto

.Las Clases son descriptivas - plantillas

. Clase vehiculo

Atributos:
. matricula . marca . modelo .año
comportamiento:
. arrancar . frenar . reversa

Tipo de datos Objeto: Byte Short Integer Long Float Double Characer Boolean String

Variables ≠ Objetos

. Variables son entidades elementales (muy sencillas)

- Un número

-Un carácter

-Un valor verdadero/falso

. Objetos son entidades complejas que pueden estar formadas por la agrupación de muchas variables y métodos

Declaración de métodos

. Una declaración de un método es un elemento de código en Java que:

-Consiste en cuatro partes: tipo de datos de regreso, nombre, argumentos y cuerpo entre llaves.

Mod. acceso	Valor regreso	nombre	argumentos
public	int	suma	(int a int b)

-Tienen un valor de regreso explícitamente invocado en su cuerpo usando la palabra reservada return.

- No regresa ningún valor si es declarado void.

- No puede declararse de otro método.

```
public int suma(int a  int b) {
        return a+b;
 }
```

Constructor

. Un constructor es un conjunto de sentencias que:

- Crea nuevas instancias de una clase.
- Tiene el mismo nombre que la clase inicializa.
-Usa la palabra reservada new para invocarlo.
- Usa cero o más argumentos contenidos dentro los paréntesis que siguen al nombre
- No regresa valor

. La sintaxis para llamarlo es:

TipoClase variable = new TipoClase (Argumentos);

Getters y Setters

. Un conjunto de métodos se crean por lo general en una clase para leer/escribir específicamente los valores de las variables miembro.

. Estos se llaman getters – se utilizan para obtener valores.

. Y setters – se utilizan para cambiar los valores de las variables miembros

Control de acceso

Alcance	Visibilidad
public	Accesible en cualquier lugar que una clase es accesible
protected	Accesible en subclases, en clases que residen en el mismo paquete y en la clase en donde esta definido.
private	Accesible solo en la clase en donde esta definido
default	Accesible por clases que residen en el mismo paquete y en la clase que en donde esta definido

Access Levels

Modifier	Class	Package	Subclass	All Other
public	Y	Y	Y	Y
protected	Y	Y	Y	N
Default	Y	Y	N	N
private	Y	N	N	N

List

.Una list es una interfaz del framework de collections de Java que:

- Contiene datos en un orden definido y puede tener elementos aplicados.

-Tiene implementaciones generales de ArrayList y Vector

-Requiere del paquete java.util

-Almacena elementos por medio de un índice entero, iniciando con cero.

-Extiende a la interfaz Collection.

Clases que implementa la interfaz List

. La clase ArrayList almacena un arreglo que puede cambiar de tamaño, tiene una capacidad especifica que crece

```
//Declaramos un ArrayList
ArrayList a = new ArrayList();
// Añadir elementos
a.add("Lenguaje"),
a.add("a");
a.add(23.5);
```

. Un vector es similar a un array, la diferencia estriba en que un vector crece automáticamente cuando alcanza la dimensión inicial máxima.

. Además, proporciona métodos iniciales para añadir, eliminar elementos, e insertar elemento entre dos ya existentes.

Herencia

. En algunas circunstancias, es necesario utilizar el estado y comportamiento de una clase en conjunto con otras clases

. La jerarquía de herencia en java permite tener esta opción en nuestro código

. Se establece una relación padre-hijo entre dos objetos diferentes

. La idea de la herencia es permitir crear nuevas clases basadas en las ya existentes

.La herencia sirve para crear objetos que incorporen propiedades y métodos de otros objetos

.Así evitamos reescribir utilizando el código ya existentes

Subclase y Súper Clase

. Una subclase es una clase que:

- Heredada de una súper clase.

-Es reservada con la palabra reservada extends.

-Puede acceder miembros de la súper clase con el mismo nombre con la palabra super y notación punto.

-Se debe tener una implementación de los métodos de la superclase y constructores en su cuero.

La declaración de una subclase y súper clase usa la siguiente sintaxis:

```
Class SuperClass {…}
Class SubClass extends SuperClass {...}
```

Miembros heredados en la subclase

. Una subclase hereda todos los miembros de su súper clase que están declarados como public o protected.

. Los miembros heredados pueden tener múltiples formas dentro de la subclase y superclase, esto se conoce como poliformismo.

Sobre escritura de constructores

. La sobre escritura de constructores es la definición de un constructor en una subclase usando los miembros heredados de la superclase, con argumentos diferentes.

.Tú sobrescribes constructores simplemente cambiando cualquier argento que sea necesario

. El constructor de la subclase invoca al constructor de la superclase.

.Para ello se incluye, obligatoriamente, la palabra clave **super** como primera línea del constructor de la subclase.

. La palabra super irá seguida de paréntesis dentro de los cuales pondremos los parámetros que requería el constructor de la superclase al que queramos invocar.

```
/* constructor  de una subclase */
public Subclase (parámetros…) {
    //invocar al constructor de la superclase
   super(parámtros para la superclase);
    // inicializa sus atributos
      ……
 }
```

Polimorfismo

.Posibilidad de construir varios métodos con el mismo nombre, pero con relación a la clase a la que pertenece cada uno, con comportamientos diferentes.

Implementación de Interfaces

. En Java una interfaz es una clase abstracta pura, es decir una clase donde todos los métodos son abstractos (no se implementan ninguno)

. Permite al diseñador de clases establecer la forma de una clase (nombres de métodos, listas de argumentos y tipos de retorno, pero no bloques de código).