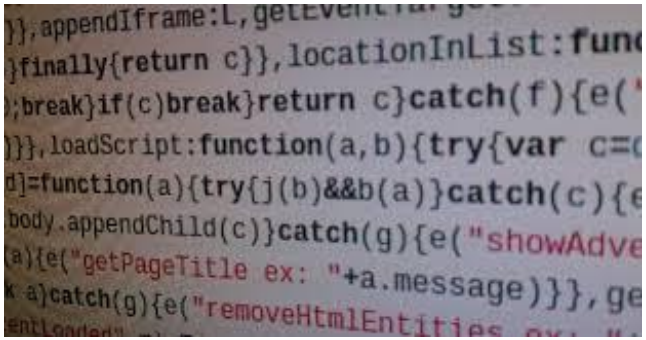


UT 9

INTRODUCCIÓN P.O.O. DESARROLLO DE CLASES

IES JUAN DE LA CIERVA
DPTO. INFORMÁTICA

CONTENIDOS DE LA PRESENTACIÓN

Objetivos

Esta unidad introduce al alumno hacia los conceptos generales de clase, la estructura y miembros de la clase. Los modificadores de acceso.

Contenidos

PModificadores de Acceso.
Encapsulación y visibilidad.
Paquetes y su uso.
Static
Recursividad
Argumentos de longitud variable
Argumetos del método main()

Modificador: static

Modificadores

Los atributos y métodos también pueden tener otros modificadores:

- **static**, todos los objetos de la clase lo comparten y pueden ser modificados por todos los objetos de una misma clase. Solo se crea una vez. Pertenece a la clase y no a la instancia de la clase.
- **final**, su valor no puede ser modificado..
- **transient**, el atributo no es serializable. Permite almacenar objetos para utilizarlos posteriormente para mandarlos por la red,….
- **volatile**, que no realice ciertas optimizaciones al compilar.

Modificador static

Si a un miembro de una clase le anteponemos la palabra **static**, lo convertimos en miembro de clase independiente de los objetos de la misma.

Cuando se instancian objetos de la clase no se hacen copias de las variables static, son iguales para todos los objetos.

Se pueden declarar como static:

- métodos
- variables
- Constantes

Atributos Estáticos

<modoAcceso><modificadorstatic>tipoAtributo nombreAtributo

- Están asociados a la clase, no a una instancia de ella.
- Cuando se instancian objetos de la clase no se hacen copias de las variables static, son iguales para todos los objetos.
- Son llamados atributos estáticos
- Están compartidos por todas las instancias de la clase
- Pueden ser manipulados por cualquier instancia, pero sus cambios afectan a todas las demás.
- También pueden ser manipulados sin crear instancias de clase

Ejemplo : Bicicleta

Métodos Estáticos

[modo acceso][modificador] <tipo de dato de retorno> nombre (lista de parametros){…..}

- Similar a los atributos static
- Se pueden invocar sin crear una instancia de esa clase
  - Clase.metodoEstatico();
- Dentro de un método estático solo se pueden usar los miembros de la clase que sean estáticos.
- Hay clases con métodos auxiliares como por ejemplo la clase Array, cuyos métodos son static. No se crea para ser instanciada.
- Los métodos **static** no tienen referencia this.

CONSTANTES

- Se suelen definir como estáticas.
- No se puede modificar su valor (error)
- Nomenclatura: en mayúsculas, separando palabras con guiones bajos.

CONSTANTES

- **Constantes:** los atributos compartidos por todas las instancias de la clase deben inicializarse una sola vez.
- Se definen con:

  static final tipo identificador = valor.

  static final double PI = 3.14159265;

- Si a la variable le anteponemos la palabra **final**, es una constante de la instancia, que no puede cambiar de valor.

- Se realiza en la declaración o en el constructor.

Atributos y métodos static

Ejemplo: Bicicleta

Atributos y métodos static

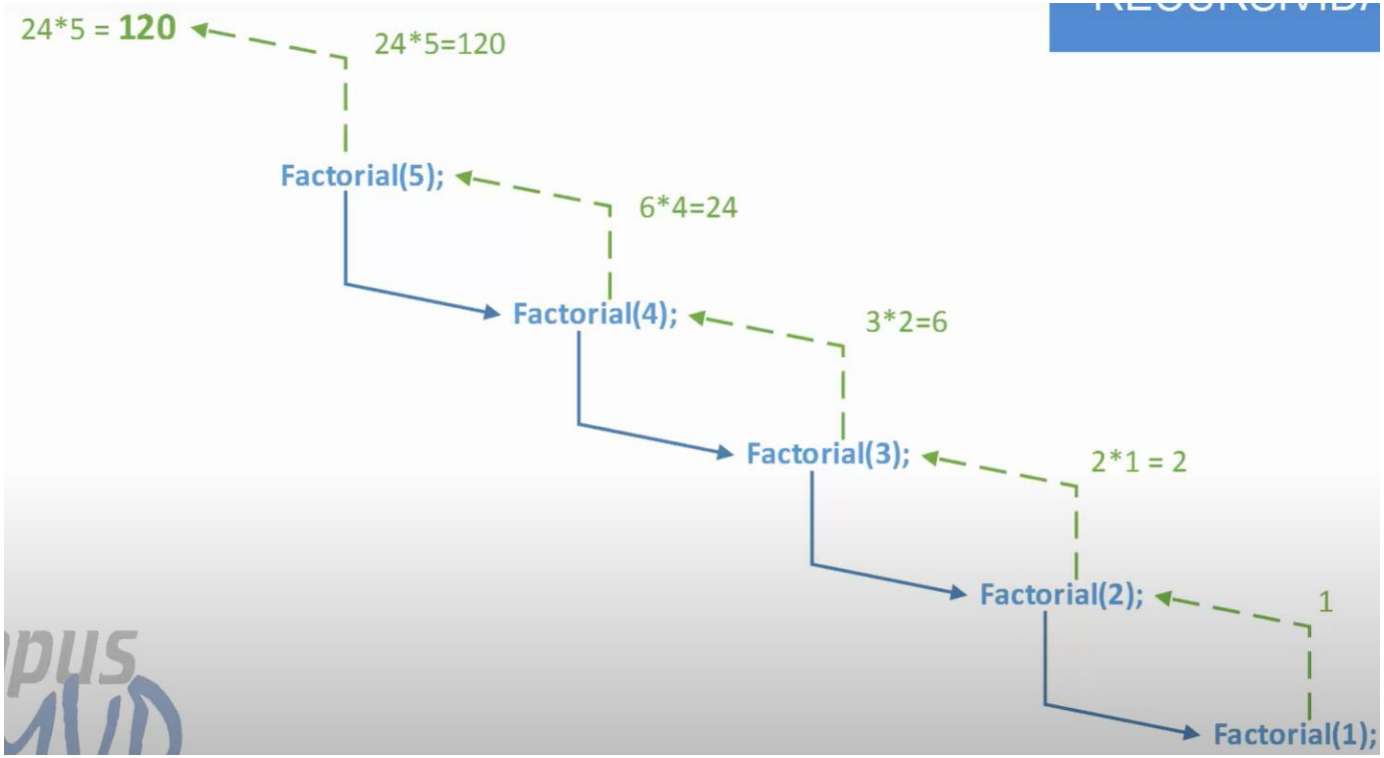
```
public class Bicicleta {
    //Atributos
    private int diametroRueda;
    private int numMarchas;
    private int id;

    //variable estática
    private static int numeroDeBicicletas=0;

    public static int getNumeroDeBicicletas() {
        return numeroDeBicicletas;
    }

    //Construtores
    public Bicicleta() {
        super();
        id=++numeroDeBicicletas;
    }

    public Bicicleta(int diametroRueda, int numMarchas) {
        super();
        this.diametroRueda = diametroRueda;
        this.numMarchas = numMarchas;
        this.id =++numeroDeBicicletas;
    }
```

Atributos y métodos static

```
public class ElementosEstaticos {
    public static void main(String[] args) {
        Bicicleta bici1= new Bicicleta(26,27);
        Bicicleta bici2= new Bicicleta(16,24);

        System.out.println(bici1);
        System.out.println(bici2);

        System.out.println(Bicicleta.getNumeroDeBicicletas());
        System.out.println(bici1.getNumeroDeBicicletas());


    }
```

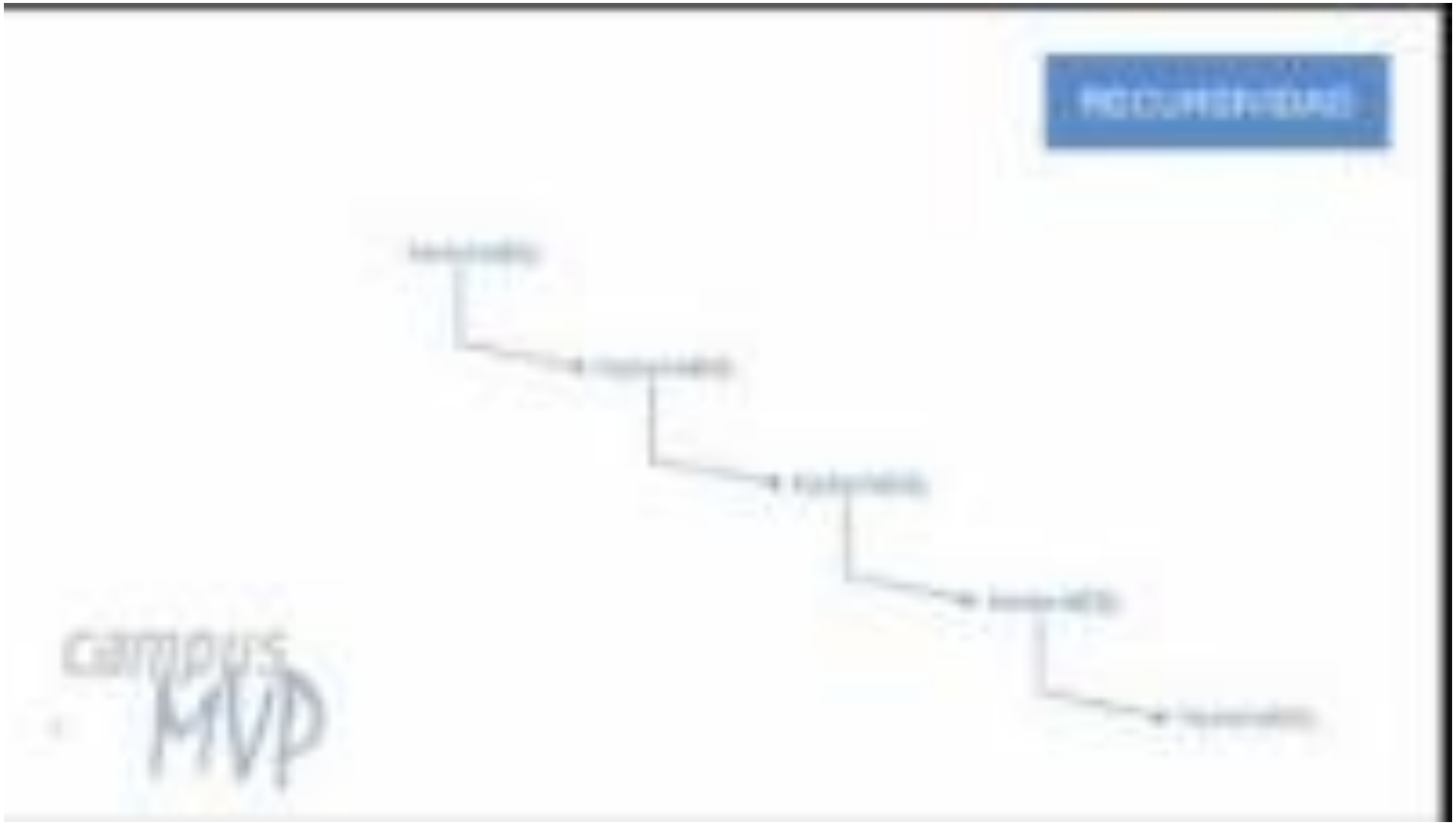
Clases: Recursividad

Clases: Recursividad



https://youtu.be/rCXmRKXOJOU

Clases: Recursividad



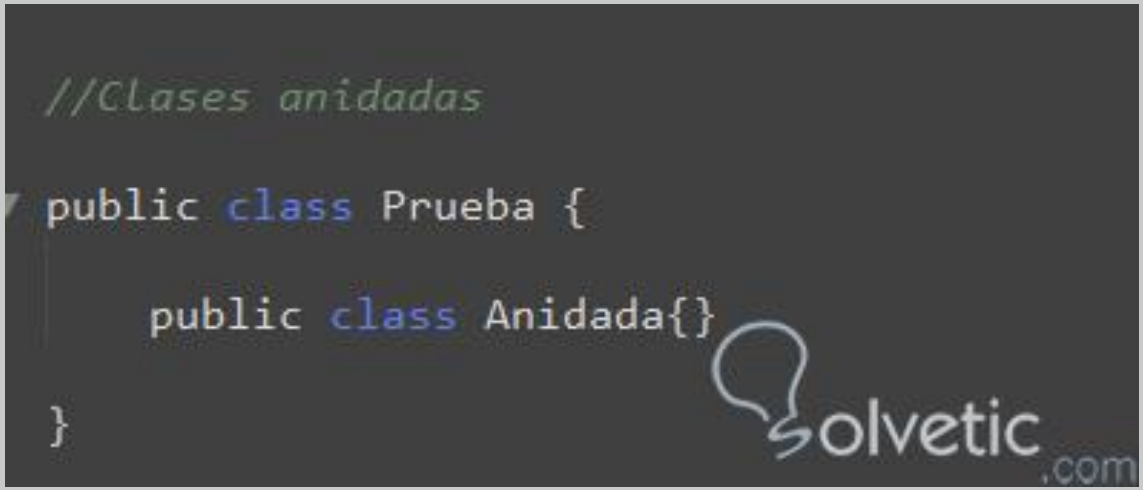
Clases: Recursividad

En Java un método puede invocarse a sí mismo.

A este proceso se le llama recursividad o recursión, y al método que lo realiza se le denomina recursivo.

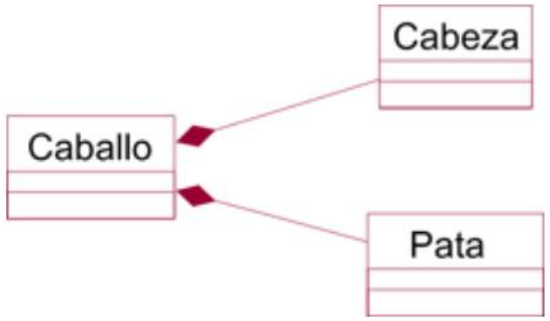
Ejemplo

```
class Factorial {
        int fact(int n) {
            if (n==0) return 1;
            return n*fact(n-1);
         }
 }
```

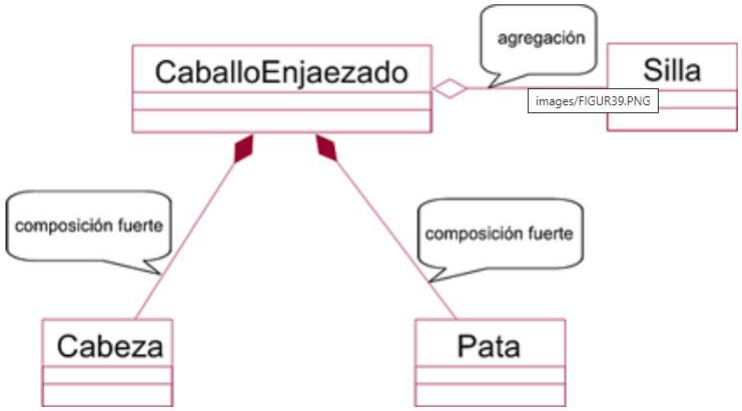


Clases: Clases anidadas

Composición Débil - Fuerte



Composición Fuerte



Composición Débil - Agregación

Clases anidadas - Composición

En Java se pueden definir clases anidadas, es decir **una clase que se define dentro de otra.**

A la clase anidada debe ser estática y es un miembro de la clase contenedora.

La clase anidada tiene acceso a todas las variables y métodos de la clase externa, pero no al revés.

En ocasiones se usan clases internas para **ofrecer una serie de servicios que sólo necesitan usarse en su clase contenedora** e **incrementar el nivel de encapsulación** u ocultamiento.

Clases anidadas

```
public class Composicion {
    // Programa Java para demostrar el acceso a una clase anidada estática

    //Declaración de variables
    static int externo_x=10;
    int externo_y=20;
    private static int externo_privado=30;
    //System.out.println("variable de la clase anidada: "+internaX);

    //Declaracion Clase anidada
    public static class ClaseAnidada{
        int interna;

        public ClaseAnidada() {}

        void mostrar(){
            System.out.println("externo_x: "+externo_x);
            System.out.println("externo_privado: "+externo_privado);
        }
    }
}
```

Clases anidadas

```
public class ClaseAndidadaStaticDemo {
    public static void main(String[] args) {
        //Accediendo a la clase anidada estática
        Composicion.ClaseAnidada objetoAnidado = new Composicion.ClaseAnidada();
        objetoAnidado.mostrar();
    }

}
```

Argumentos de longitud variable

Argumentos de longitud variable

Un método que acepta un número variable de argumentos se denomina método varargs.

La lista de parámetros de un método varargs no es fija, sino de longitud variable.

Se especifica:

static void vaTest(**tipo_arg … nombre**)

que crea un vector nombre con elementos de tipo tipo_arg

(*)Puedo mezclar argumentos normales y de longitud variable. Este irá siempre al final.

- **Ejemplos** : **ArgsLongVar y ArgsLongVar1**

Ejemplo I Uso Argumentos de longitud variable

```
// sencillo ejemplo de método con argumentos de longitud variable
import java.io.*;
import java.util.*;

class ArgsLongVar {
    static void vaTest(int ... v){
        System.out.println("Numero de argumentos: "+v.length);
        System.out.println("Contenido: ");
        for (int i=0; i<v.length;i++)
            System.out.println(" arg "+i+": "+v[i]);
    }

    public static void main(String args[]){

        //Diferentes invocaciones de vaTest
        vaTest(10);        // 1 argumento
        vaTest(1, 2, 3);   // 3 argumentos
        vaTest();          //sin argumentos
    }
}
```

Ejemplo II Uso Argumentos de longitud variable

```
// sencillo ejemplo de método con argumentos de longitud variable
import java.io.*;
import java.util.*;

class ArgsLongVar1 {
    static void vaTest(String...s){
        System.out.println("Numero de argumentos: "+s.length);
        System.out.println("Contenido: ");
        for (int i=0; i<s.length;i++)
            System.out.println(" arg "+i+": "+s[i]);
    }

    public static void main(String args[]){

        //Diferentes invocaciones de vaTest
        vaTest("hola");        // 1 argumento
        vaTest("a", "b", "c");  // 3 argumentos
        vaTest();              //sin argumentos

    }
}
```

Argumentos del método main()

public static void main(String[] args)

- Al argumento podemos darle el nombre que queramos.

- Un programa java se puede ejecutar desde la línea de comandos del sistema operativo con la orden:

  C:\> java nombrePrograma

- Pero además, mediante esta orden, podemos enviar valores al programa.

  Ejemplo: si tenemos un programa que se llama *ordenar* que ordena 5 números enteros, pero en lugar de leerlos por teclado los números se le pasan al programa como argumentos, este programa lo ejecutaríamos así:

  - **C:\> java ordenar 4 6 3 7 1**

Argumentos del método main()

```
public class Saludo {

    public static void main(String[] args) {
        if (args.length > 1) { //si hay más de 1 parámetro
            for(String arg:args) {
                System.out.println("Hola "+arg);
            }

        } else if (args.length == 0) { //si no hay parámetros
            System.out.println("Falta el nombre de la persona");
        } else {
            System.out.println("Buenos Días " + args[0]);
        }

    }

}
```