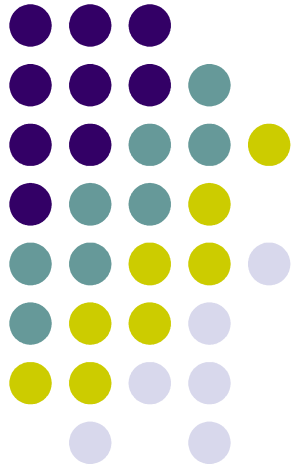
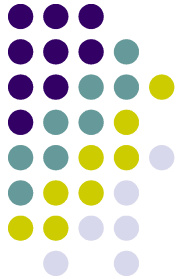
Detalles Generales sobre Java



Estructura



Los métodos en una clase tienen la siguiente estructura:

```
[modificadores] retorno nombre([tipo nombre]*) [throws Ex1, Ex2]
```

Los modificadores son opcionales.

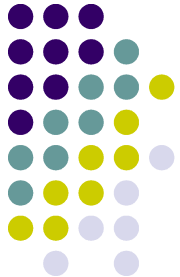
retorno indica el valor que el método devuelve. La palabra return regresa un valor del método. Si no regresa un valor se usa la palabra reservada void.

```
public int metodo() {
    return 7;
}
```

nombre indica el identificador del método.

Los métodos reciben cero o más argumentos.

Estructura



Los argumentos (o parámetros) deben tener nombres distintos.

Cuando un método recibe más de un argumento se utilizan comas para separarlos.

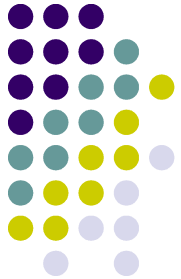
Un método puede lanzar cero o más excepciones.

Los tipos de excepciones lanzados por el metodo se separan por comas.

Ejemplo:

```
private static double suma(double a, double b)
public int divide(int a, int b) throws ArithmeticException
void calcula() throws ExceptionA, ExceptionB
```

Constructores



Los constructores son métodos especiales que son llamados cuando una clase se instancia para obtener un objeto.

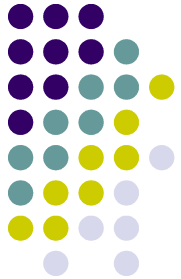
Llevan el mismo nombre que la clase que los define y tiene la siguiente estructura:

```
[modificador acceso] nombre([argumentos]*) [excepciones]*
```

Ejemplo:

```
public class Bicicleta
{
    public Bicicleta() throws Exception
    {
    }
}
```

Constructores



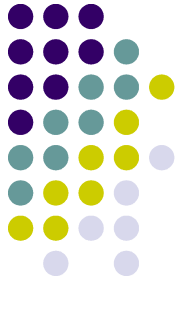
Para crear un objeto se utiliza la palabra reservada new seguida por el nombre de la clase.

```
Bicicleta silla = new Bicicleta();
```

Si no se define un constructor para una clase, el compilador Java crea un constructor vacío por defecto:

```
public NombreClase()
```

Firma



La firma de un método o constructor está compuesta por el nombre y los argumentos que recibe.

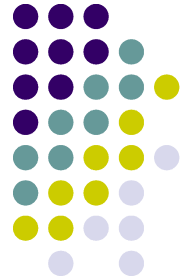
Si dos métodos o constructores tienen el mismo nombre y un número distinto de argumentos se les considera distintos.

```
public void nombreMetodo()
public void nombreMetodo(int a)
```

Si los tipos de los argumentos son distintos, los métodos son diferentes.

```
public int suma(int a, int b)
public float suma(float a, float b)
```

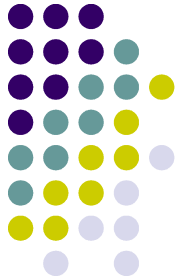
Sobrecargar (overloading)



Un método (o constructor) está sobrecargado cuando existe más de una definición del método con distintas firmas.

A pesar de tener el mismo nombre, Java los considera como métodos distintos porque tienen una firma distinta.

Sobreescribir (overwriting)



Cuando se presenta una relación de herencia el hijo puede redefinir el coportamiento de un método en particular.

La operación de redefinición de un método se llama sobreescribir.

Para sobreescribir un método se debe definir un método con la misma firma del método original en el padre.

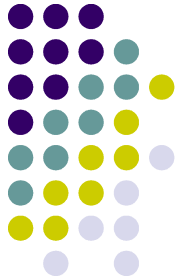
Ejemplo

```
public class Padre {
    public String getNombre() {
        return “Padre”;
    }
}


public class Hijo extends Padre {
    public String getNombre(String nombre) {
        return nombre;
    }
    public String getNombre() {
        return “hijo”;
    }

}
```

Interfaces



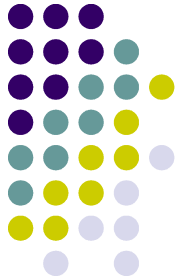
Las interfaces son archivos de Java que pueden definir cero o más métodos.

Las interfaces simplemente proveen las firmas de los métodos, no la implantaciones.

Una interfaz puede tener herencia de otra interfaz.

```
interface NombreInterface extends InterfacePadre
{
   public int metodo1();
   public boolean metodo2();
}
```

Interfaces



Las clases pueden implementar más de una interfaz utilizando la palabra reservada implements.

```
public class Hijo extends Padre implements Interfaz1, Interfaz2
```

Una clase que implementa una interfaz que defina una o más métodos debe proveer el código para cada método, de lo contrario debe ser declarada como abstracta.

Algunas interfaces sirven como marcadores de clases. Dichas interfaces carecen de definiciones de métodos.