Programacion orientada a objetos

Resumen sobre Java

Ultima modificación:3 de octubre de 2005

Bueno, esto es un resumen mínimo sobre lo indispensable para programar en Java.

Cabe aclarar que las recomendaciones de este documento son para “zafar” del parcial, y no son la mejor manera de realizar programas eficientes, y/o bien realizados.

Al principio del código, incluir:

```
import java.util.*;
```

Un poco de teoría:

Podemos realizar asociaciones de la siguiente manera (hay otras maneras también):

- Composición } “es parte de”
- Agregación } “es parte de”
- Herencia } especialización “es un”

La clase tiene interfaz privada e interfaz publica ; lo normal es poner todos los atributos como privados o protegidos, y los métodos de acceso a la clase como públicos.

En cuanto al polimorfismo, cabe destacar que la sobrecarga de métodos no es polimorfismo.

Algunos tipos primitivos de Java

- `int`
- `long`
- `char`
- `float`
- `double`
- `boolean`

Ejemplo de como definir una clase:

```
public class NombreClase
{
	private int atributo1; // un atributo o variable de instancia
	public NombreClase()
	{
		// este es el constructor de la clase
		atributo1 = 0; // ejemplo de como iniciarlo, todos los atributos deberían iniciarse aquí
	}
	// ejemplo de como sobrecargar el constructor
	public NombreClase(int p)
	{
		atributo1 = p;
	}
}
```

Definir en este orden

- atributos
- constructor
- métodos

Vectores: métodos mas comunes:

```
add(objeto);
remove(objeto);
size();
elementAt(indice 0..size-1);
```

Nota: siempre recordar hacer `atributoVector = new Vector();` en el constructor, de lo contrario no anda.

Nota: siempre hacer el **cast** al obtener un objeto desde un vector.

Ejemplo: `Clase1 obj = (Clase1)vector.elementAt(i);`

Strings: métodos mas comunes:

```
equals(otraString);
equalsIgnoreCase(otraString);
trim();
length();
replace( oldChar,  newChar);
substring(int beginIndex);
toLowerCase();
toUpperCase();
```

Herencia

Ejemplo:

```
public class Empleado
{
	// blah blah aca
	public Empleado() { // mas blah aca }
}

public class Docente extends Empleado
{
	public Docente()
	{
		super Empleado(); // llamamos al constructor de la clase padre
		// aca inicializo lo que sea de la clase Docente
	}
}
```

Nota: para poder usar los métodos get y set sin castear, hay que definir los atributos de la superclase como **protected.**

La manera correcta, hablando en términos de OO, es hacer el Método de la clase hija, y usar **super**.

Ejemplo:

```
public class Docente extends Empleado
{
	public string getNombre()
	{
		return super getNombre(); // llama a getNombre de Empleado.
	}
}
```

Polimorfismo

A groso modo, defino el Método en la superclase, pero solo lo implemento en las clases hijas. Ejemplo:

```
public abstract class empleado
{
      public abstract float calcularSueldo(); // NO lo implemente
}
// en la clase hija, se implementa
public class Docente extends Empleado
{
      public float calcularSueldo();
      {
            return valorHora * horas + bonificacion; // aca lo implemente
      }
}
```

Nota: una clase definida como **final** no permite que se le haga herencia.