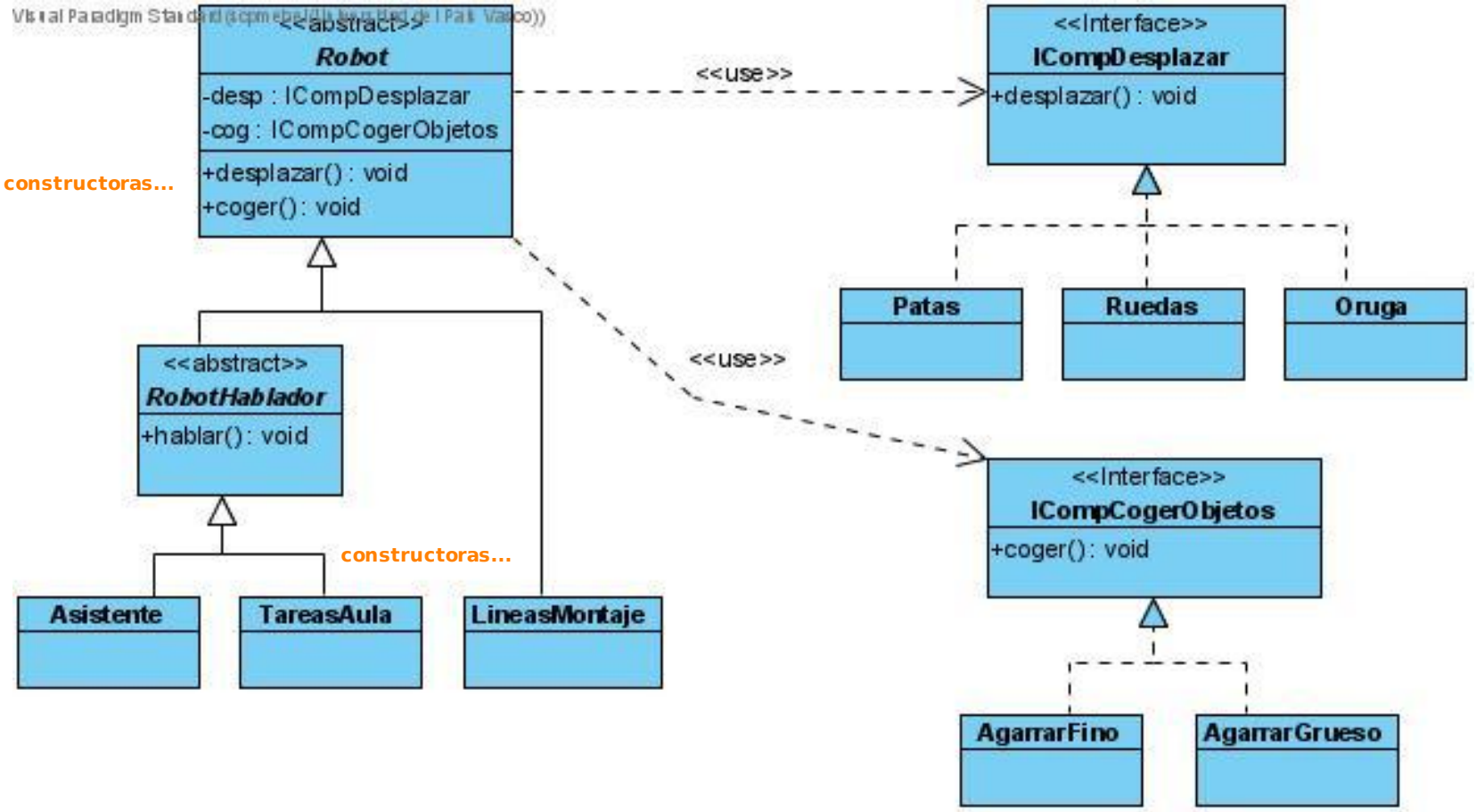
sin ChabgeStrategy

DIAGRAMA DE CLASES (Robots)



CÓDIGO (Robots)

```
public abstract class Robot {
	private ICompDesplazar desp;
	private ICompCogerObjetos cog;
	public Robot(ICompDesplazar pDesp, ICompCogerObjetos pCog)
	{
		this.desp = pDesp;
		this.cog = pCog;
	}
	public void desplazar() {
		desp.desplazar();
	}
	public void coger() {
		cog.coger();
	}
}

public class LineasMontaje extends Robot {
	public LineasMontaje(ICompDesplazar pDesp, ICompCogerObjetos
pCog)
	{
		super(pDesp, pCog);
	}
}

public abstract class RobotHablador extends Robot {
	public RobotHablador(ICompDesplazar pDesp, ICompCogerObjetos
pCog) {
		super(pDesp, pCog);
	}
	public abstract void hablar();
}
```

protected

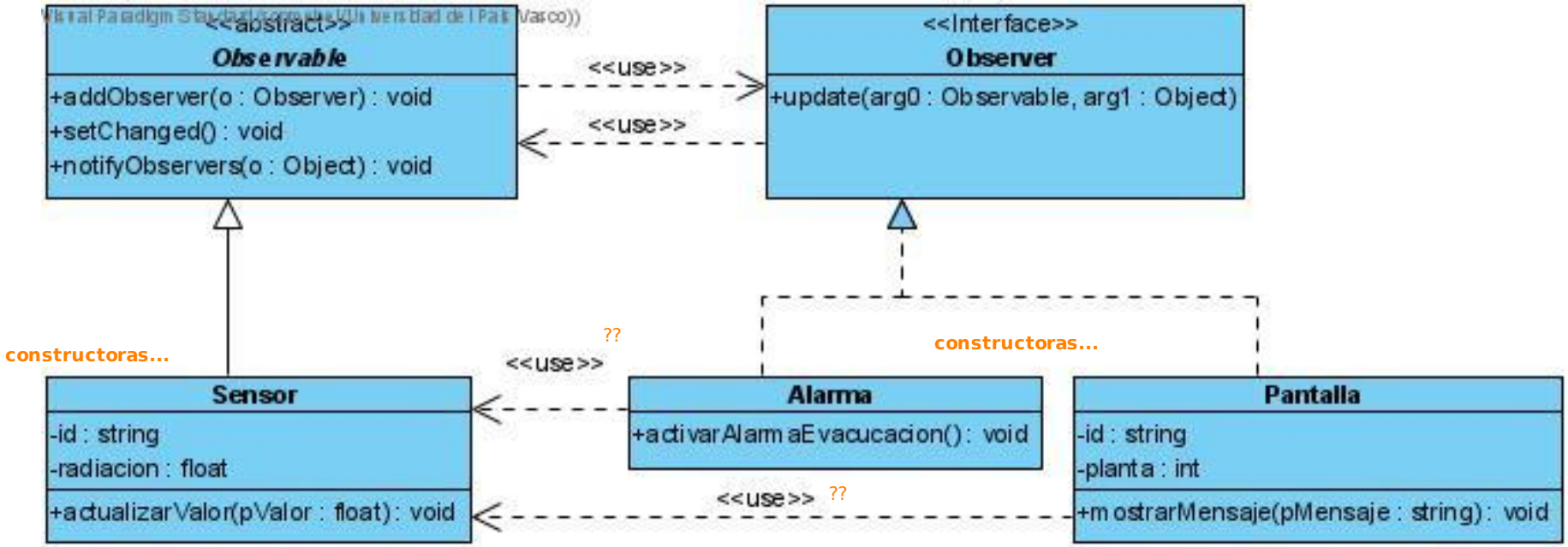
```
public class TareasAula extends RobotHablador {
	public TareasAula(ICompDesplazar pDesp, ICompCogerObjetos
pCog) {
		super(pDesp, pCog);
	}
	public void hablar() {
		System.out.println("habla robot tareas");

	}
}
public class Asistente extends RobotHablador {...}


public interface ICompCogerObjetos {
	void coger();
}
public class AgarrarFino implements ICompCogerObjetos {
	public void coger() {
		System.out.println("coger modo fino");
	}
}
public class AgarrarGrueso implements ICompCogerObjetos {...}


public interface ICompDesplazar {
	void desplazar();
}
public class Patas implements ICompDesplazar {
	public void desplazar() {
		System.out.println("desplazar con patas");
	}
}
public class Ruedas implements ICompDesplazar {...}
public class Oruga implements ICompDesplazar {...}
```

DIAGRAMA DE CLASES (Central Nucelar)



CÓDIGO (Central Nucelar)

```
public class Sensor extends Observable {
	private float radiacion;
	private String id;
	public Sensor(String i)
	{
		id = i;
		radiacion = 0;
	}
	public void actualizarValor(float val)
	{
		if(radiacion!=val){
			radiacion = val;
			setChanged();
			notifyObservers(val);
		}
	}
}
```

```
public class Pantalla implements Observer {
	private String id;
	private int planta;
	public Pantalla(String i, Observable o)
	{
		id=i;
		o.addObserver(this);
	}
	public void update(Observable arg0, Object arg1) {
		if(arg1 instanceof Float)
		{
			float valor = (float) arg1;
			mostrarMensaje("la radiacion es " + valor);
		}
	}
	public void mostrarMensaje(String pMensaje)
	{
		System.out.println(pMensaje);
	}
}
```

```
public class Alarma implements Observer{
	public Alarma( Observable o)
	{
		o.addObserver(this);
	}
	public void update(Observable arg0, Object arg1) {
		if(arg1 instanceof Float)
		{
			float valor = (float) arg1;
			if(valor > 3)
			{
				activarAlarmaEvacuacion();
			}
		}
	}
	public void activarAlarmaEvacuacion()
	{
		System.out.println("alarma de evacuacion");
	}
}
```