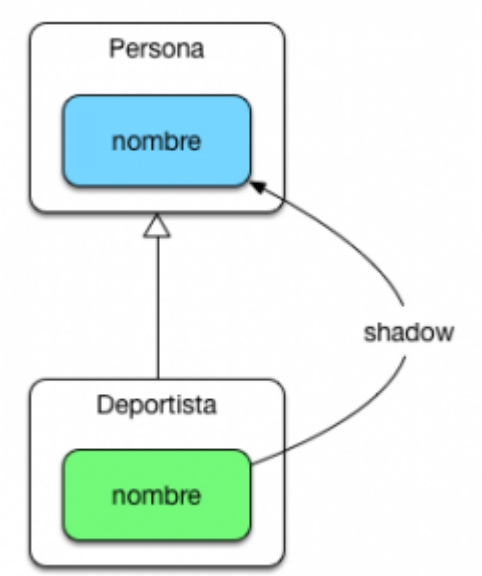
El concepto de Java Shadow Variables a veces resulta difícil de entender y es una de las preguntas típicas de los examenes de certificación. ¿ Qué son exactamente las shadow variables o variables de sombra? . Vamos a explicarlo utilizando un diagrama de clases de Herencia.

CURSO SPRING BOOT
GRATIS
APUNTATE!!



Las Java Shadow variables son variables que tienen el mismo nombre pero diferente ámbito. Por ejemplo en el diagrama tenemos dos variables nombre que se encuentran en distinto nivel de la jerarquía. Una pertenece a la clase Persona y la otra a la clase Deportista.

```
package com.arquitecturajava;
```

```
public class Persona {

private String nombre="pepe";

public String getNombre() {
return nombre;
}

public void setNombre(String nombre) {
this.nombre = nombre;
}

public void imprimirNombre() {

System.out.println(nombre);
}
}
```

```
package com.arquitecturajava;

public class Deportista extends Persona {

private String nombre="juan";

public String getNombre() {
return nombre;
}
```

```
public void setNombre(String nombre) {
this.nombre = nombre;
}

}
```

En la clase Persona el nombre se llama “pepe” y en la clase Deportista se llama “juan” . Vamos a construir un programa que instancie un objeto de la clase Deportista.

```
package com.arquitecturajava;

public class Principal {

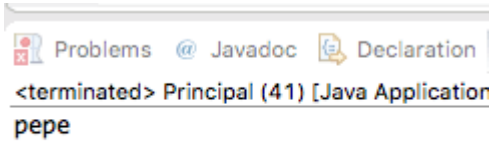
public static void main(String[] args) {
Deportista d= new Deportista();
d.imprimirNombre();

}

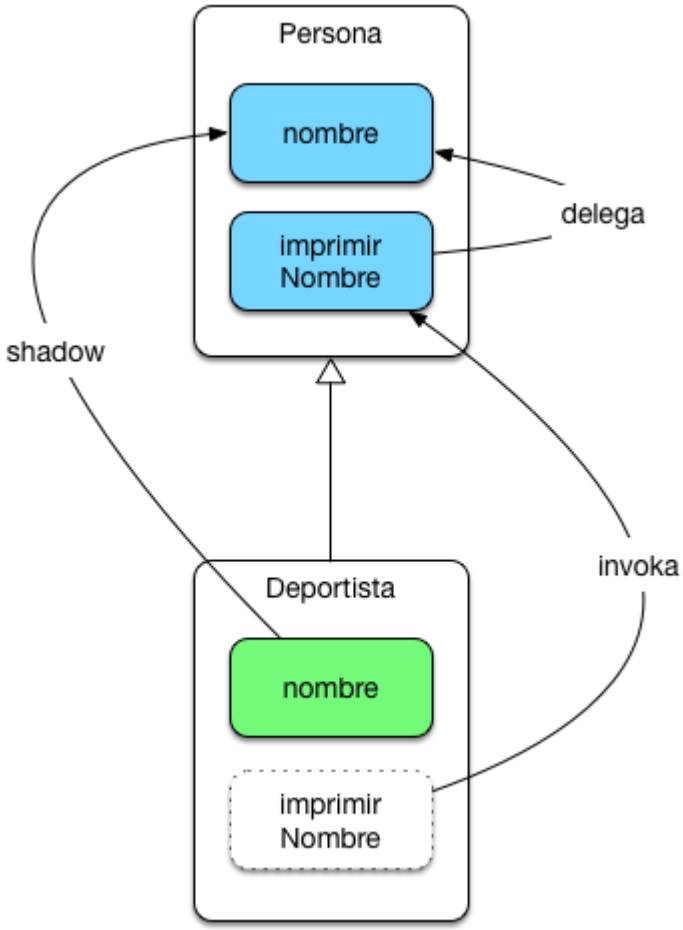
}
```

Java Shadow Variables

Muchas personas piensan que cuando ejecutamos el método imprimirNombre , el resultado será “juan” ya que la variable nombre hace referencia al Deportista y hace shadow sobre la variable nombre de la clase Persona. Sin embargo el resultado es “pepe”.



¿Qué es lo que ha sucedido exactamente? . Recordemos que la clase Persona es la que implementa el método imprimirNombre() por lo tanto cuando ejecutemos en la clase Deportista este método realmente estará delegando en la clase Persona.



Por eso se imprime el nombre de “pepe” y no de juan ya que el método se invoca sobre la clase Persona. En cambio si nosotros simplemente usamos un método getNombre de Deportista:

```
package com.arquitecturajava;

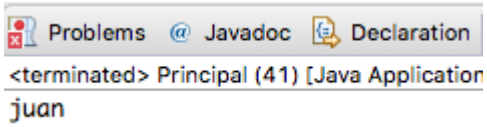
public class Principal {

public static void main(String[] args) {
 Deportista d= new Deportista();
 System.out.println(d.getNombre());

}

}
```

El resultado será “juan”.

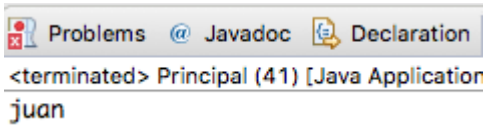


Por último podemos sobreescribir el método imprimirNombre en la clase Deportista.

```
 @Override
 public void imprimirNombre() {

 System.out.println(nombre);
 }
```

En este caso cuando ejecutemos el resultado será:



Ahora si que se produce un efecto de shadow sobre la variable nombre ya que el método accede a la variable que tiene la propia clase. Estas y otras preguntas similares son clásicas del examen de Java Programmer.

CURSO SPRING BOOT
GRATIS
APUNTATE!!

Otros artículos relacionados:

Java this vs super()

Java equals vs ==