<!--SOLID-->

SOLID - Principio de Sustitución de Liskov (LSP)

<Por="Raquel Martinez"/>

}

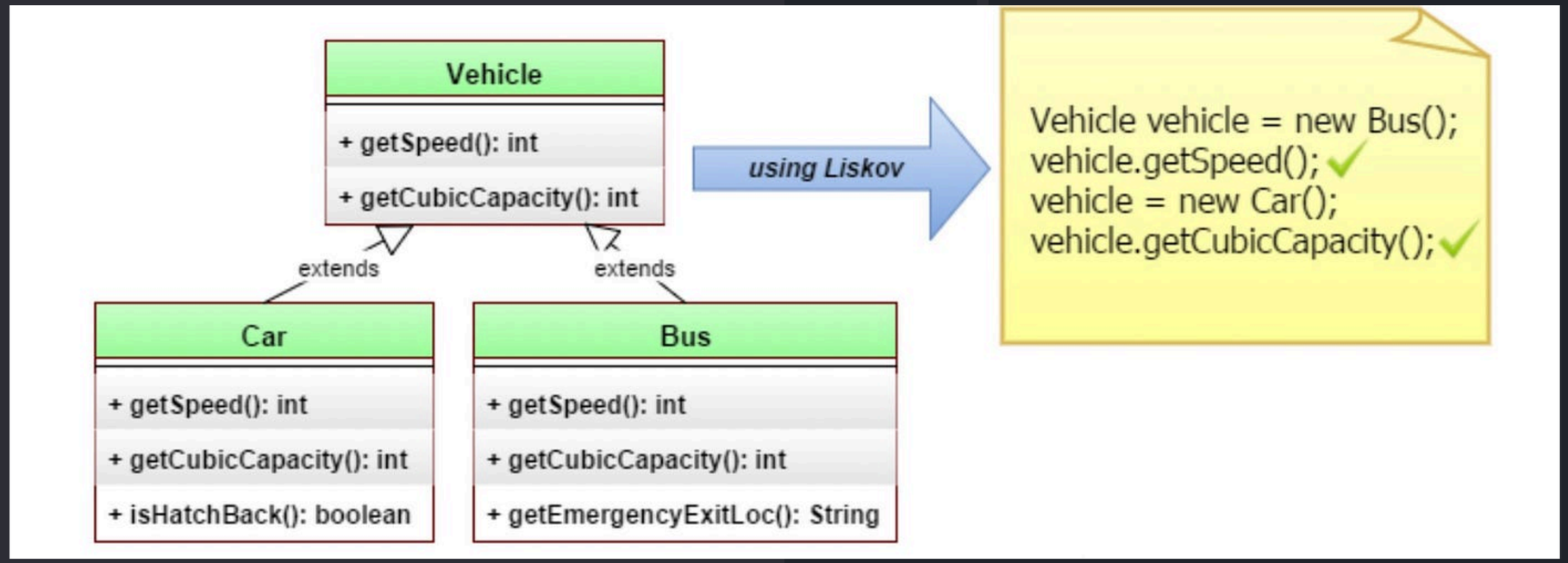
Definición{

Si S es un subtipo de T, entonces los objetos de tipo T en un programa pueden ser reemplazados por objetos de tipo S sin alterar ninguna de las propiedades deseables del programa (correctitud, tarea realizada, etc.).

}

Definición{

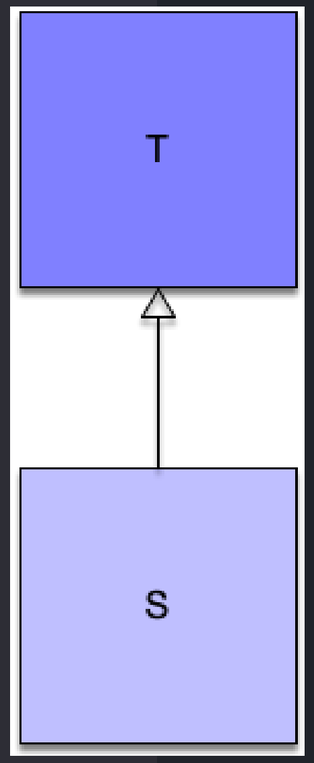
Si S es un subtipo de T, entonces los objetos de tipo T en un programa pueden ser reemplazados por objetos de tipo S sin alterar ninguna de las propiedades deseables del programa (correctitud, tarea realizada, etc.).



}

Beneficios {

- Reutilización del código
- Mantenimiento del código
- Polimorfismo
- Correctitud

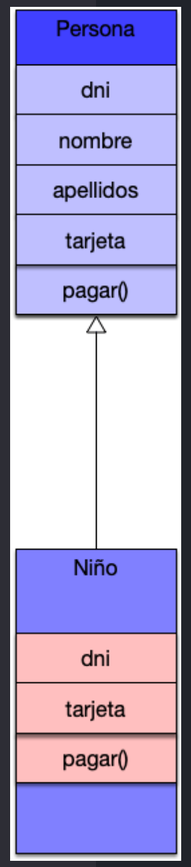


}

Herencia{

}

Herencia{

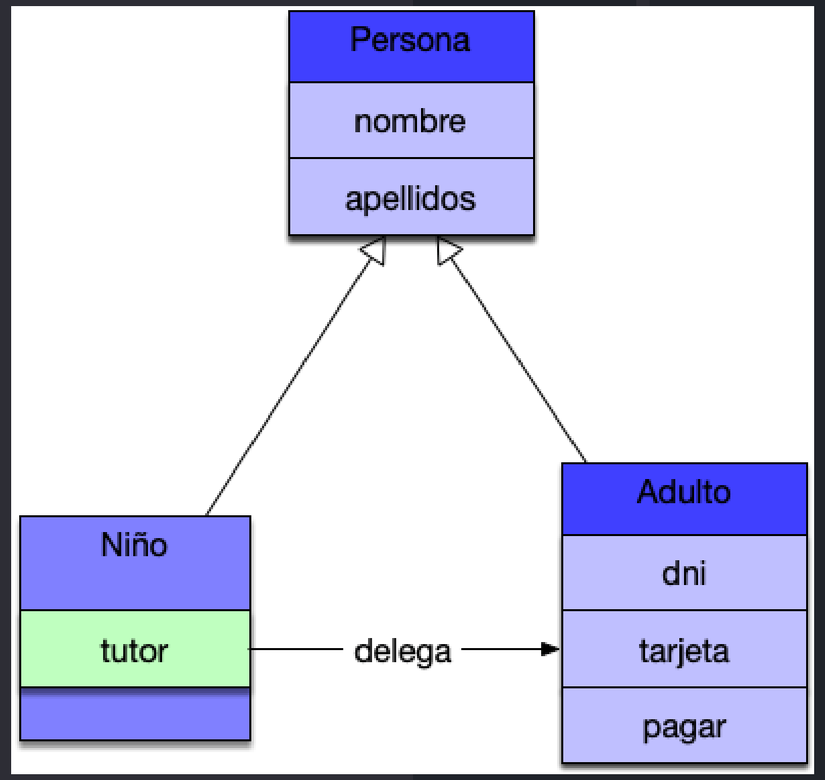

Persona
dni
nombre
apellidos
tarjeta
pagar()
Niño
dni
tarjeta
pagar()

}

```java
package com.arquitecturajava;

public class Niño  extends Persona{

    public Niño(String nombre, String apellidos) {
        super(null, nombre, apellidos, null);
        // TODO Auto-generated constructor stub
    }

    @Override
    public void pagar() {
        // TODO Auto-generated method stub
        throw new RuntimeException("un niño no puede pagar");
    }



}
```

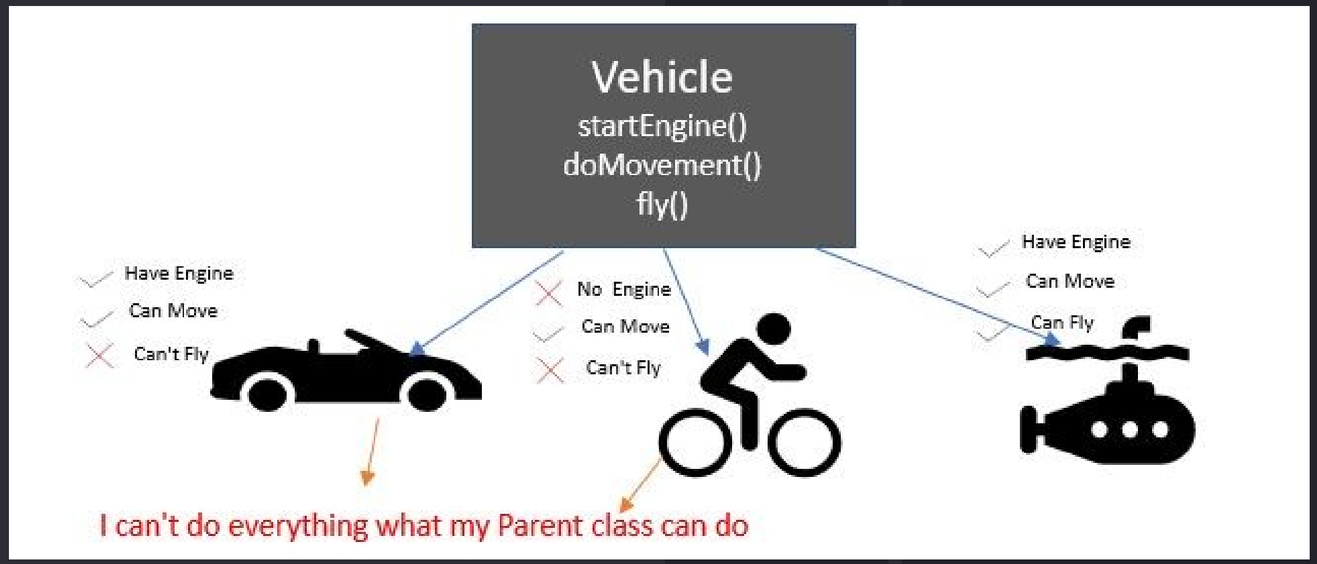
}

Herencia{


Persona
nombre
apellidos
Niño
tutor
delega
Adulto
dni
tarjeta
pagar

}

Herencia{


Vehicle
startEngine()
doMovement()
fly()
Have Engine
Can Move
Can't Fly
No Engine
Can Move
Can't Fly
Have Engine
Can Move
Can Fly
I can't do everything what my Parent class can do

}

Herencia{

```
public class DeviceWithoutEngine extends TransportationDevice {
    public void startMoving(){
        System.out.println("Basic functionality of moving for device
without engine");
    }
}

public class DeviceWithEngine {
    public void startEngine(){
        System.out.println("Basic engine start functionality");
    }
}
```

}

<!--SOLID-->

Codifiquemos! {

<Ejemplo/>

}

<!--SOLID-->

Gracias {

<Por="Raquel Martinez/>

}