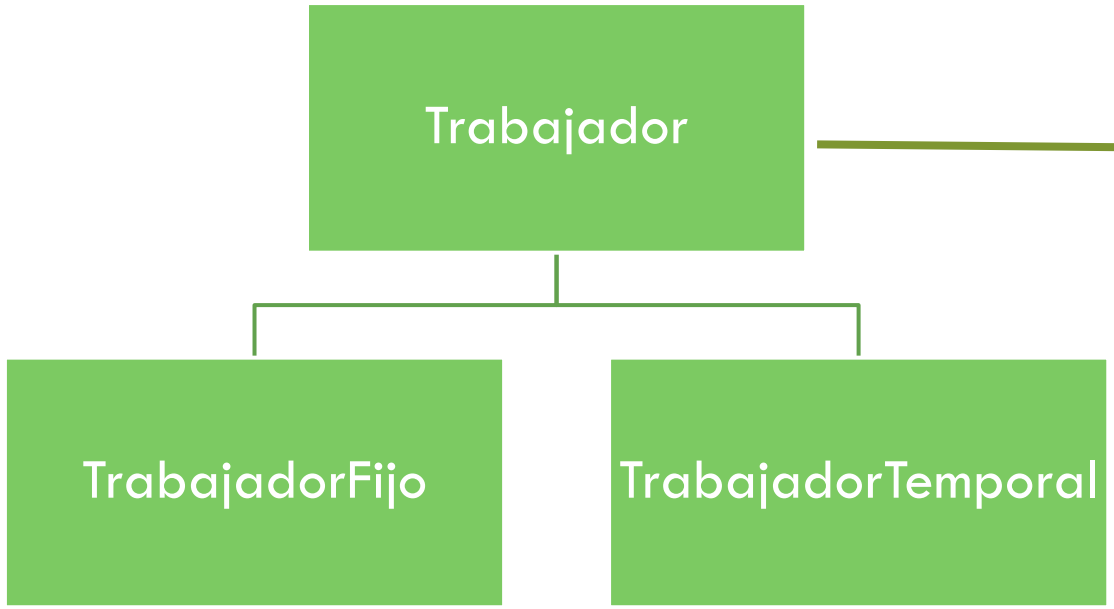
Clases abstractas y métodos abstractos

- Son clases genéricas.
- Conveniente para utilizarlas como superclases en alguna jerarquía de herencia ➔ programas más reutilizables y extensibles por el uso de la herencia y el polimorfismo.



Los trabajadores que manejemos caerán obligatoriamente en una de estas dos categorías ➔ no se van a instanciar objetos de ella

Clases abstractos y métodos abstractos

- **No se permite crear objetos de estas clases.**
- Una clase abstracta puede implementar **métodos abstractos** y **no abstractos.**
- Un método que no proporciona **ninguna implementación** recibe el nombre de **método abstracto** → palabra clave **abstract**

```
[private | protected | public] abstract tipo_retorno nombre_método(parámetros) excepciones;
```

- Si una clase tiene un método abstracto → es una clase abstracta.

```
[public] abstract class Nombre_de_la_clase [extends superclase] implements lista_de_interfaces
```

Clases abstractos y métodos abstractos

- Un método **abstract no puede ser static**.
- Es posible que una subclase de una clase abstracta sea también abstracta.
- Las clases terminales de una jerarquía de herencia serán obligatoriamente **clases concretas**.
- Los constructores no se heredan, no pueden ser abstractos.
- Las subclases que implementan esta clase abstracta tendrán que redefinir estos métodos o bien declararlos también como **abstract**.

Una clase abstracta que tiene un método abstracto

```
public abstract class Vehiculo{
        private int peso;

        public void setPeso(int peso){
            this.peso=p;
        }
        public abstract int getVelocidadActual();
}
```

Características de las clases abstractas

- Una clase con métodos abstractos **obligatoriamente** se define como **clase abstracta**.
- Las clases abstractas sólo especifican lo que sus subclases tienen en común ➔ superclases en las jerarquías de herencia
- **No** se pueden utilizar para **instanciar objetos** ➔ carecen de implementación en algún método.
- Sus subclases deberán implementar los métodos abstractos y sí podrán instanciar objetos.
- **TrabajadorFijo** y **TrabajadorTemporal**, implementan el método abstracto **getSueldo()** heredado de la superclase abstracta **Trabajador**.

El método `getSueldo()` es un método abstracto heredado de la clase Trabajador e implementado en la clase TrabajadorFijo, por lo que aparece un destornillador verde a la izquierda del nombre del método.

```
146
147
148
149
    public double getSueldo(){
151
152         int posicion;
153
154         // El sueldo del trabajador depende de su categoría profesional, por lo que averiguamos la
155         // categoría profesional del trabajador
156         posicion = this.posicionCategoria(getCategoriaProfesional());
157
158         // posicion será mayor o igual a cero si la categoría del trabajador es una categoría válida y será
159         // un número negativo si la categoría del trabajador no forma parte de las categorías registradas
160         // en la variable de clase categorias
161         if (posicion >= 0)
162             // El sueldo del trabajador será igual al sueldo base más el complemento asociado a su categoría
163             // profesional, complemento que viene recogido también en la variable de clase categorias
164             return (TrabajadorFijo.getComplementoCategoria(posicion) + TrabajadorFijo.getSalarioBase() +
165                     50 * this.getNumHijos() + 20 * this.getAntigüedad());
166         else
167             return (TrabajadorFijo.getSalarioBase() + 50 * this.getNumHijos() + 20 * this.getAntigüedad());
168     }
169
```

Ahora, cada vez que se invoque el método sobre un objeto de la clase TrabajadorFijo, se ejecutará este código