В этом примере есть 2 класса: ElectricBulb, ElectricCettle, которые имплементируют интерфейс Swichable

в котором есть только 2 метода.

Также есть класс Switch который своими методами делает что-то с экземплярами классов которые имплементируют

интерфейс Switchable.

Дак вот чтобы не создавать много обьектов разных классов в классе Switch с которыми он что то делает, можно

зделать имплементацию этими классами интерфейса. А в параметры методов которые что-то делают передовать именно

Swichable, т.е. ссылку на интерфейс. В этом случае switch-у без разницы с каким именно обьектом он работает

главное чтобы класс этого обьекта имплементировал интерфейс Switchable.

Смотри класс Switch

Класс может реализовывать более чем 1 интерфейс

В этом случае он должен реализовывать все методы из каждого интерфейса.

1)-создадим ещё один интерфейс Dimable, только с одним методом.

2)-создадим класс реализующий сразу и Swichable и Dimable, -ElectricDimableBulb

В классе Switch добавим метод который может обрабатывать Dimable.

При этом так как ElectricDimableBulb реализует и Swichable в нём пропишим методы из интерфейса Swichable,

значит обьект от этого класса может одновременно быть и Swichable и Dimable.

По итогу обьект класса switch сможет одинаково обрабатывать и Swichable и Dimable обьекты.

Важно!

мы можем иметь какие-то разные обьекты но Благодаря имплементации их классов какогото интерфейса эти обьекты

схожи по поведению и это нам позволяет хранить их в коллекциях указав тип данных <название\_интерфейса>. И так

как все они реализуют какоето общее поведение мы можем к примеру вызвать это поведение у каждого в цикле и пр.

пример:

List <Swichable> myList = new ArrayList();

myList.add(electricBulb);

myList.add(electricCettle);

myList.add(electricDimableBulb);

Важно!

также если классы имплементируют не один интерфейс, а нам надо сабрать их все в один тип данных то можно

создать интерфейс который будет включать в себя методы из каждого нужного нам класса, тем самым обьеденив их

по поведению, но это не отменяет что помимо этого у них есть ещо другое поведение (так как некоторые из них

могут реализовывать разные интерфейсы). Другими словами мы можем обьединить эти обьекты по разным поведениям.

Важно!

Интерфейсы можно наследовать друг от друга. Соответственно если класс реализует конечный интерфейс то этот же

класс реализует и предыдущий интерфейс.

Cоздадим интерфес Combined и унаследуемся от Dimable. Теперь создадим класс ElctroBike и реализуем в нём

интерфейс Combined для этого нам придётся реализовать все методы из Dimable и из Combined.