Adaptor design pattern :

<https://www.journaldev.com/1487/adapter-design-pattern-java>

<https://www.youtube.com/watch?v=FsS-jWCPtQI&t=306s> ( good example)

The Adapter Pattern provides an implementation of the Dependency Inversion design principle.

Examples:

1. WebDriver : Chrome and IE
2. TimeZone : IST, UST
3. Adaptor : Mobile

**Rules of thumb**

* Adapter makes things work after they're designed; Bridge makes them work before they are.
* Bridge is designed up-front to let the abstraction and the implementation vary independently. Adapter is retrofitted to make unrelated classes work together.
* Adapter provides a different interface to its subject. Proxy provides the same interface. Decorator provides an enhanced interface.
* Adapter is meant to change the interface of an existing object. Decorator enhances another object without changing its interface. Decorator is thus more transparent to the application than an adapter is. As a consequence, Decorator supports recursive composition, which isn't possible with pure Adapters.
* Facade defines a new interface, whereas Adapter reuses an old interface. Remember that Adapter makes two existing interfaces work together as opposed to defining an entirely new one.