

Exercise - Adapter Pattern (25 minutes)

The provided code for the Adapter Pattern exercise contains the following classes:

1. A class named `ContactManager` that contains a collection of `Contacts` (as in personal contacts). Each `Contact` object contains the name and contact information for a person.
2. A class named `Table` that can be used to display a table of data in the console (i.e., shell). `Table` can display any data/object that implements the `TableData` interface.

Write a program that uses the `Table` class to display the contents of a `ContactManager` object in the console. This will require you to write an adapter class that wraps a `ContactManager` object and implements the `TableData` interface.

Exercise - Decorator Pattern (25 minutes)

1. Define an interface named `StringSource` that represents an object that produces strings. Your interface should look something like this (modified as needed for your language):

```
interface StringSource { String next(); }
```

2. Write at least two classes that implement the `StringSource` interface, and that return some interesting strings. The strings may be hard-coded, come from a file, come from the keyboard, or wherever you like.
3. Implement the Decorator Pattern by creating at least three decorator classes each of which performs some kind of transformation on a `StringSource`. For example, a decorator might reverse strings or manipulate whitespace or add punctuation or whatever you can think of. Be creative.
4. Write a program that demonstrates your string decorators in action.

Zip up your code for both exercises and submit it on Canvas.