**Façade pattern example:**

Another good example can be the startup of a computer. When a computer starts up, it involves the work of cpu, memory, hard drive, etc. To make it easy to use for users, we can add a facade which wrap the complexity of the task, and provide one simple interface instead

Now Let’s try and understand the facade pattern better using a simple example. Let’s consider a hotel. This hotel has a hotel keeper. There are a lot of restaurants inside hotel e.g. Veg restaurants, Non-Veg restaurants and Veg/Non Both restaurants.  
You, as client want access to different menus of different restaurants . You do not know what are the different menus they have. You just have access to hotel keeper who knows his hotel well. Whichever menu you want, you tell the hotel keeper and he takes it out of from the respective restaurants and hands it over to you. Here, the hotel keeper acts as the **facade**, as he hides the complexities of the system hotel.

**Singleton pattern example**:

for example, a single database object shared by different parts of the program. The **Singleton pattern** disables all other means of creating objects of a **class** except for the special creation method.