ADAPTER PATTERN

The adapter pattern is a structural design pattern. In the adapter pattern, a wrapper class (ie, the adapter) is used translate requests from it to another class (ie, the adaptee). In effect, an adapter provides particular interactions with an adaptee that are not offered directly by the adaptee.



The adapter pattern takes two forms. In the first form, a "class adapter" utilizes inheritance. The class adapter extends the adaptee class and adds the desired methods to the adapter. These methods can be declared in an interface (ie, the "target" interface).

In the second form, an "object adapter" utilizes composition. The object adapter contains an adaptee and implements the target interface to interact with the adaptee.

Now let's look at simple examples of a class adapter and an object adapter. First, we have an adaptee class named CelciusReporter. It stores a temperature value in Celcius.

[**CelciusReporter.java**](https://www.avajava.com/tutorials/design-patterns/adapter-pattern/CelciusReporter.java)

**public** **class** CelciusReporter {

 **double** temperatureInC;

 **public** CelciusReporter() {

 }

 **public** **double** getTemperature() {

 **return** temperatureInC;

 }

 **public** **void** setTemperature(**double** temperatureInC) {

 **this**.temperatureInC = temperatureInC;

 }

]

Here is our target interface that will be implemented by our adapter. It defines actions that our adapter will perform.

[**TemperatureInfo.java**](https://www.avajava.com/tutorials/design-patterns/adapter-pattern/TemperatureInfo.java)

**public** **interface** TemperatureInfo {

 **public** **double** getTemperatureInF();

 **public** **void** setTemperatureInF(**double** temperatureInF);

 **public** **double** getTemperatureInC();

 **public** **void** setTemperatureInC(**double** temperatureInC);

}

TemperatureClassReporter is a class adapter. It extends CelciusReporter (the adaptee) and implements TemperatureInfo (the target interface). If a temperature is in Celcius, TemperatureClassReporter utilizes the temperatureInC value from CelciusReporter. Fahrenheit requests are internally handled in Celcius.

[**TemperatureClassReporter.java**](https://www.avajava.com/tutorials/design-patterns/adapter-pattern/TemperatureClassReporter.java)

;

// example of a class adapter

**public** **class** TemperatureClassReporter **extends** CelciusReporter **implements** TemperatureInfo {

 @Override

 **public** **double** getTemperatureInC() {

 **return** temperatureInC;

 }

 @Override

 **public** **double** getTemperatureInF() {

 **return** cToF(temperatureInC);

 }

 @Override

 **public** **void** setTemperatureInC(**double** temperatureInC) {

 **this**.temperatureInC = temperatureInC;

 }

 @Override

 **public** **void** setTemperatureInF(**double** temperatureInF) {

 **this**.temperatureInC = fToC(temperatureInF);

 }

 **private** **double** fToC(**double** f) {

 **return** ((f - 32) \* 5 / 9);

 }

 **private** **double** cToF(**double** c) {

 **return** ((c \* 9 / 5) + 32);

 }

}

TemperatureObjectReporter is an object adapter. It is similar in functionality to TemperatureClassReporter, except that it utilizes composition for the CelciusReporter rather than inheritance.

[**TemperatureObjectReporter.java**](https://www.avajava.com/tutorials/design-patterns/adapter-pattern/TemperatureObjectReporter.java)

// example of an object adapter

**public** **class** TemperatureObjectReporter **implements** TemperatureInfo {

 CelciusReporter celciusReporter;

 **public** TemperatureObjectReporter() {

 celciusReporter = **new** CelciusReporter();

 }

 @Override

 **public** **double** getTemperatureInC() {

 **return** celciusReporter.getTemperature();

 }

 @Override

 **public** **double** getTemperatureInF() {

 **return** cToF(celciusReporter.getTemperature());

 }

 @Override

 **public** **void** setTemperatureInC(**double** temperatureInC) {

 celciusReporter.setTemperature(temperatureInC);

 }

 @Override

 **public** **void** setTemperatureInF(**double** temperatureInF) {

 celciusReporter.setTemperature(fToC(temperatureInF));

 }

 **private** **double** fToC(**double** f) {

 **return** ((f - 32) \* 5 / 9);

 }

 **private** **double** cToF(**double** c) {

 **return** ((c \* 9 / 5) + 32);

 }

}

The AdapterDemo class is a client class that demonstrates the adapter pattern. First, it creates a TemperatureClassReporter object and references it via a TemperatureInfo reference. It demonstrates calls to the class adapter via the TemperatureInfo interface. After this, it creates a TemperatureObjectReporter object and references it via the same TemperatureInfo reference. It then demonstrates calls to the object adapter.

[**AdapterDemo.java**](https://www.avajava.com/tutorials/design-patterns/adapter-pattern/AdapterDemo.java)

**public** **class** AdapterDemo {

 **public** **static** **void** main(String[] args) {

 // class adapter

 System.out.println(**"class adapter test"**);

 TemperatureInfo tempInfo = **new** TemperatureClassReporter();

 testTempInfo(tempInfo);

 // object adapter

 System.out.println(**"\nobject adapter test"**);

 tempInfo = **new** TemperatureObjectReporter();

 testTempInfo(tempInfo);

 }

 **public** **static** **void** testTempInfo(TemperatureInfo tempInfo) {

 tempInfo.setTemperatureInC(0);

 System.out.println(**"temp in C:"** + tempInfo.getTemperatureInC());

 System.out.println(**"temp in F:"** + tempInfo.getTemperatureInF());

 tempInfo.setTemperatureInF(85);

 System.out.println(**"temp in C:"** + tempInfo.getTemperatureInC());

 System.out.println(**"temp in F:"** + tempInfo.getTemperatureInF());

 }

}

The console output of the execution of AdapterDemo is shown here.

[**Console Output**](https://www.avajava.com/tutorials/design-patterns/adapter-pattern/console.txt)

class adapter test

temp in C:0.0

temp in F:32.0

temp in C:29.444444444444443

temp in F:85.0
object adapter test

temp in C:0.0

temp in F:32.0

temp in C:29.444444444444443

temp in F:85.0