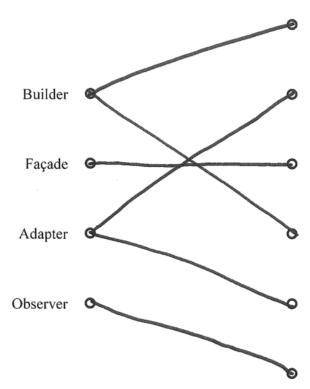**Problem**: Match the design pattern to the situation to which you should apply it.

○ Your application needs to generate HTML files (from scratch).

Builder ○

○ Your program must support switching among several different email libraries, but each one has a slightly different interface.

Façade ○

○ Sending an SMS message requires lots of big, ugly code, involving connection, message, and other objects.

Adapter ○

○ Your program has to create and configure some big, ugly record objects before inserting them into a database.

Observer ○

○ Your company already implemented a component that almost implements the interface that you need, but not quite.

○ Your Pac-Man program needs to listen for presses of the arrow keys and to update Pac-Man's position in the maze accordingly.

Solution:

Builder — Your program has to create and configure some big, ugly record objects before inserting them into a database.

Façade — Sending an SMS message requires lots of big, ugly code, involving connection, message, and other objects.

Adapter — Your application needs to generate HTML files (from scratch).

Observer — Your company already implemented a component that almost implements the interface that you need, but not quite.

Your application needs to generate HTML files (from scratch).

Your program must support switching among several different email libraries, but each one has a slightly different interface.

Sending an SMS message requires lots of big, ugly code, involving connection, message, and other objects.

Your program has to create and configure some big, ugly record objects before inserting them into a database.

Your company already implemented a component that almost implements the interface that you need, but not quite.

Your Pac-Man program needs to listen for presses of the arrow keys and to update Pac-Man's position in the maze accordingly.