Inverting your thinking...

Okay, so you need to implement a Pizza Store. What's the first thought that pops into your head?

Right, you start at the top and follow things down to the concrete classes. But, as you've seen, you don't want your pizza store to know about the concrete pizza types, because then it'll be dependent on all those concrete classes!

Now, let's "invert" your thinking...instead of starting at the top, start at the Pizzas and think about what you can abstract.

Right! You are thinking about the abstraction *Pizza*. So now, go back and think about the design of the Pizza Store again.

Close. But to do that you'll have to rely on a factory to get those concrete classes out of your Pizza Store. Once you've done that, your different concrete pizza types depend only on an abstraction, and so does your store. We've taken a design where the store depended on concrete classes and inverted those dependencies (along with your thinking).