# Building a simple pizza factory

We'll start with the factory itself. What we're going to do is define a class that encapsulates the object creation for all pizzas. Here it is...

Here's our new class, the SimplePizzaFactory. It has one job in life: creating pizzas for its clients.

First we define a createPizza() method in the factory. This is the method all clients will use to instantiate new objects.

```
public class SimplePizzaFactory {

    public Pizza createPizza(String type) {
        Pizza pizza = null;

        if (type.equals("cheese")) {
            pizza = new CheesePizza();
        } else if (type.equals("pepperoni")) {
            pizza = new PepperoniPizza();
        } else if (type.equals("clam")) {
            pizza = new ClamPizza();
        } else if (type.equals("veggie")) {
            pizza = new VeggiePizza();
        }
        return pizza;
    }
}
```

Here's the code we plucked out of the orderPizza() method.

This code is still parameterized by the type of the pizza, just like our original orderPizza() method was.

## there are no Dumb Questions

**Q: What's the advantage of this? It looks like we're just pushing the problem off to another object.**

A: One thing to remember is that the SimplePizzaFactory may have many clients. We've only seen the orderPizza() method; however, there may be a PizzaShopMenu class that uses the factory to get pizzas for their current description and price. We might also have a HomeDelivery class that handles pizzas in a different way than our PizzaShop class but is also a client of the factory.

So, by encapsulating the pizza creating in one class, we now have only one place to make modifications when the implementation changes.

And, don't forget, we're also just about to remove the concrete instantiations from our client code.

**Q: I've seen a similar design where a factory like this is defined as a static method. What's the difference?**

A: Defining a simple factory as a static method is a common technique and is often called a static factory. Why use a static method? Because you don't need to instantiate an object to make use of the create method. But it also has the disadvantage that you can't subclass and change the behavior of the create method.