A GardenMaker Factory

Let's consider a simple example where you might want to use the Abstract factory in your application.

Suppose you are writing a program to plan the layout of gardens. These could be annual gardens, vegetable gardens, or perennial gardens. No matter the type of garden, we want to ask the same questions:

1. What are good center plants?
2. What are good border plants?
3. What plants do well in partial shade?

. . . and probably many other plant questions that we'll omit in this simple example.

We want a base Garden class that can answer these questions.

```
public abstract class Garden {
   public abstract Plant getCenter();
   public abstract Plant getBorder();
   public abstract Plant getShade();
}
```

In this case, the Plant object just contains and returns the plant name.

```
public class Plant {
String name;
   public Plant(String pname) {
     name = pname;            //save name
   }
   public String getName() {
      return name;
   }
}
```

In *Design Patterns* terms, the abstract Garden class is the Abstract Factory. It defines the methods of a concrete class that can return one of several classes, in this case one each for center, border, and shade-loving plants. The Abstract Factory could also return more-specific garden information, such as soil pH and recommended moisture content.

In a real system, for each type of garden we would probably consult an elaborate database of plant information. In this example, we'll return one kind of plant from each category. So, for example, for the vegetable garden we write the following:

```
public class VeggieGarden extends Garden {
    public Plant getShade() {
        return new Plant("Broccoli");
    }
```