Abstract Factory Pattern
Mar. 8, 2007

What is it?

- Abstract Factory pattern is one level of abstraction higher than the factory pattern.
  - use when you want to return one of several related classes of objects, each of which can return several different objects on request.
- I.e., the Abstract Factory is a factory object that returns one of several factories.

One Classic Application

- Your system needs to support multiple “look-and-feel” user interfaces, such as
  - Windows-9x, Motif or Macintosh.
- Tell the factory that you want your program to look like Windows and it returns a GUI factory which returns Windows-like objects.
  - Then when you request specific objects such as buttons, check boxes and windows,
  - the GUI factory returns Windows instances of these visual interface components.

Simple(r) Example

- You are writing a program to plan the layout of gardens.
- These could be annual gardens, vegetable gardens or perennial gardens.
- However, no matter which kind of garden you are planning, you want to ask the questions:
  - What are good border plants?
  - What are good center plants?
  - What plants do well in partial shade?
  - .......
- Goal: We want a base Garden class that can answer the above questions.
The Base Class

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

- our simple Plant object just contains and returns the plant name

```java
public class Plant {
    String name;
    public Plant(String plant) {
        name = plant;
    }
    public String getName() {
        return name;
    }
}
```

VegieGarden

```java
public class VegieGarden extends Garden {
    public Plant getShade() {
        return new Plant("Broccoli");
    }
    public Plant getCenter() {
        return new Plant("Corn");
    }
    public Plant getBorder() {
        return new Plant("Peas");
    }
}
```

- we need a series of Garden objects, each of which returns one of several Plant objects

Lets Construct our Abstract Factory!

- Returns one of these Garden objects based on the string it is given as an argument

```java
public class GardenMaker {
    // Abstract Factory which returns one of three gardens
    private Garden gd;
    public Garden getGarden(String gtype) {
        gd = new VegieGarden(); // default
        if (gtype.equals("Perennial"))
            gd = new PerennialGarden();
        if (gtype.equals("Annual"))
            gd = new AnnualGarden();
        return gd;
    }
}
```

Lets Use our Abstract Factory!

```java
// get a garden type based on text field
String gardenType = textField.getText();
GardenMaker gardenMaker = new GardenMaker();
garden = gardenMaker.getGarden(gardenType);
centerPlant = garden.getCenter().getName();
borderPlant = garden.getBorder().getName();
shadePlant = garden.getShade().getName();
```
Chars. of the Abstract Factory

- It isolates the concrete classes that are generated.
- The actual class names of these classes are hidden in the factory and need not be known at the client level at all.
- Because of the isolation of classes, you can change or interchange these product class families freely.