










Design Patterns

Engr. Abdul-Rahman Mahmood

DPM, MCP, QMR(ISO9001:2000)

 armahmood786@yahoo.com
 alphapeeler.sf.net/pubkeys/pkey.htm
 pk.linkedin.com/in/armahmood
 www.twitter.com/alphapeeler
 www.facebook.com/alphapeeler
 abdulmahmood-sss  alphasecure
 armahmood786@hotmail.com
 <http://alphapeeler.sf.net/me>

 alphasecure@gmail.com
 <http://alphapeeler.sourceforge.net>
 <http://alphapeeler.tumblr.com>
 armahmood786@jabber.org
 alphapeeler@aim.com
 mahmood_cubix  48660186
 alphapeeler@icloud.com
 <http://alphapeeler.sf.net/acms/>

Façade Design Pattern

More Examples

Example 2

- Let's consider a hotel. This hotel has a hotel keeper. There are a lot of restaurants inside hotel e.g. Veg restaurants, Non-Veg restaurants and Veg/Non Both restaurants.
You, as client want access to different menus of different restaurants . You do not know what are the different menus they have. You just have access to hotel keeper who knows his hotel well. Whichever menu you want, you tell the hotel keeper and he takes it out of from the respective restaurants and hands it over to you. Here, the hotel keeper acts as the **facade**, as he hides the complexities of the system hotel.

Interface of Hotel

```
public interface Hotel {  
    public Menu getMenu();  
}
```

NonVegRestaurant.java, VegRestaurant.java, VegNonBothRestaurant.java

```
public class NonVegRestaurant implements Hotel {  
    public Menu getMenu() {  
        NonVegMenu nv = new NonVegMenu();  
        return nv;  
    }  
}
```

```
public class VegRestaurant implements Hotel {  
    public Menu getMenu() {  
        VegMenu v = new VegMenu();  
        return v;  
    }  
}
```

```
public class VegNonBothRestaurant implements Hotel {  
    public Menu getMenu() {  
        Both b = new Both();  
        return b;  
    }  
}
```

HotelKeeper.java

```
public class HotelKeeper {  
    public VegMenu getVegMenu() {  
        VegRestaurant v = new VegRestaurant();  
        VegMenu vegMenu = (VegMenu)v.getMenus();  
        return vegMenu;  
    }  
  
    public NonVegMenu getNonVegMenu() {  
        NonVegRestaurant v = new NonVegRestaurant();  
        NonVegMenu NonvegMenu = (NonVegMenu)v.getMenus();  
        return NonvegMenu;  
    }  
  
    public Both getVegNonMenu() {  
        VegNonBothRestaurant v = new VegNonBothRestaurant();  
        Both bothMenu = (Both)v.getMenus();  
        return bothMenu;  
    }  
}
```

The Client

```
public class Client
{
    public static void main (String[] args)
    {
        HotelKeeper keeper = new HotelKeeper();

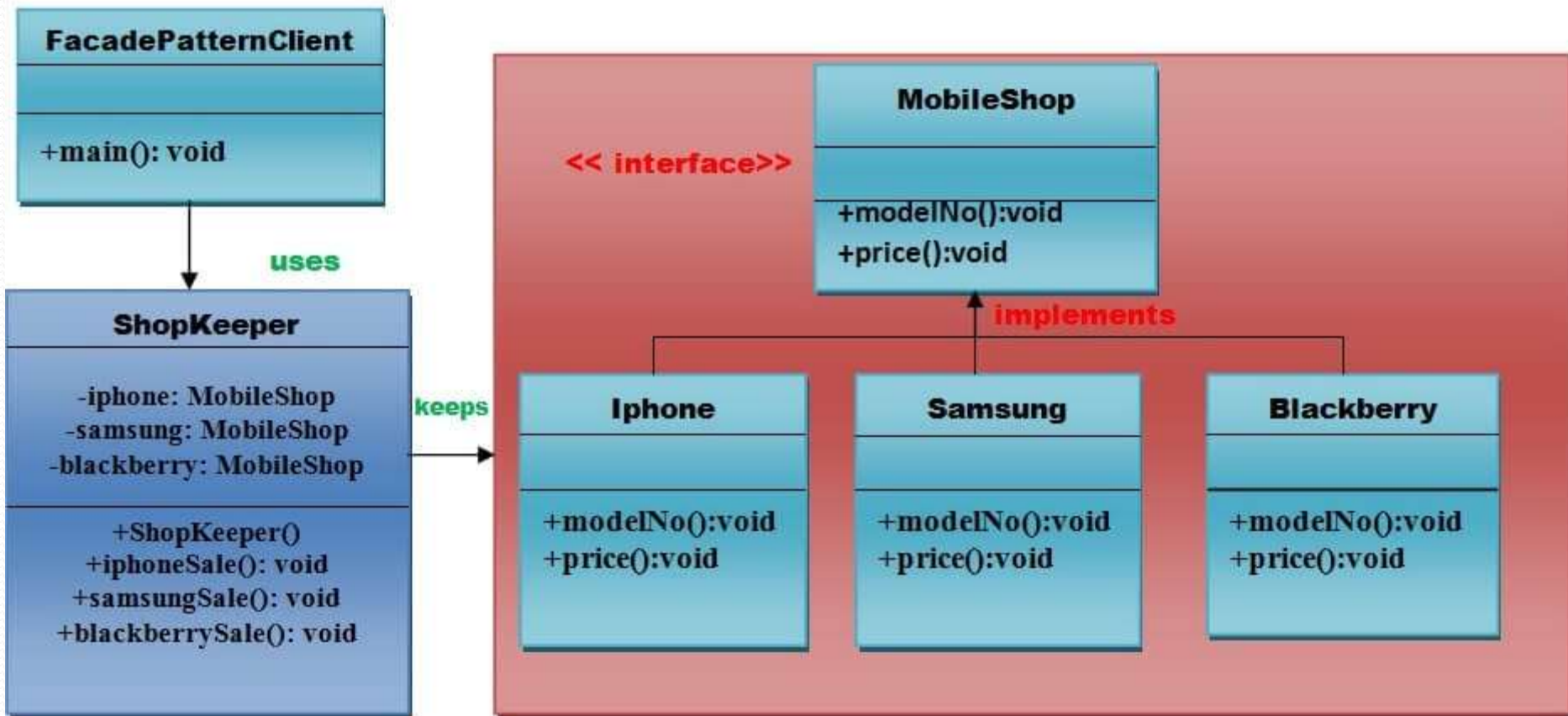
        VegMenu v = keeper.getVegMenu();
        NonVegMenu nv = keeper.getNonVegMenu();
        Both = keeper.getVegNonMenu();

    }
}
```

When Should this pattern be used?

- The facade pattern is appropriate when you have a **complex system** that you want to expose to clients in a simplified way, or you want to make an external communication layer over an existing system which is incompatible with the system. Facade deals with interfaces, not implementation. Its purpose is to hide internal complexity behind a single interface that appears simple on the outside.

Example 3



MobileShop interface

```
public interface MobileShop {  
    public void modelNo();  
    public void price();  
}
```

Classes that implement **Mobileshop** interface

```
public class Iphone implements MobileShop {  
    @Override  
    public void modelNo() {  
        System.out.println(" Iphone 6 ");  
    }  
    @Override  
    public void price() {  
        System.out.println(" Rs 65000.00 ");  
    }  
}
```

```
public class Samsung implements MobileShop {  
    @Override  
    public void modelNo() {  
        System.out.println(" Samsung galaxy tab 3 ");  
    }  
    @Override  
    public void price() {  
        System.out.println(" Rs 45000.00 ");  
    }  
}
```

Classes that implement **Mobileshop** interface

```
public class Blackberry implements MobileShop {  
    @Override  
    public void modelNo() {  
        System.out.println(" Blackberry Z10 ");  
    }  
    @Override  
    public void price() {  
        System.out.println(" Rs 55000.00 ");  
    }  
}
```

ShopKeeper.java

```
public class ShopKeeper {  
    private MobileShop iphone;  
    private MobileShop samsung;  
    private MobileShop blackberry;  
    public ShopKeeper(){  
        iphone= new Iphone();  
        samsung=new Samsung();  
        blackberry=new Blackberry();  
    }  
    public void iphoneSale(){  
        iphone.modelNo();  
        iphone.price();  
    }  
    public void samsungSale(){  
        samsung.modelNo();  
        samsung.price();  
    }  
    public void blackberrySale(){  
        blackberry.modelNo();  
        blackberry.price();  
    }  
}
```

FacadePatternClient.java

```
public class FacadePatternClient {  
    private static int choice;  
    public static void main(String args[]) throws NumberFormatException, IOException{  
        do{  
            System.out.print("===== Mobile Shop ===== \n");  
            System.out.print("        1. IPHONE.          \n");  
            System.out.print("        2. SAMSUNG.        \n");  
            System.out.print("        3. BLACKBERRY.     \n");  
            System.out.print("        4. Exit.           \n");  
            System.out.print("Enter your choice: ");  
  
            BufferedReader br=new BufferedReader(new InputStreamReader(System.in));  
            choice=Integer.parseInt(br.readLine());  
            ShopKeeper sk=new ShopKeeper();  
        }  
    }  
}
```

FacadePatternClient.java

```
        switch (choice) {
        case 1:
            sk.iphoneSale();
            break;
        case 2:
            sk.samsungSale();
            break;
        case 3:
            sk.blackberrySale();
            break;
        default:
            System.out.println("Nothing You purchased");
            return;
        }while(choice!=4);
    }
}
```

output

===== Mobile Shop =====

1. IPHONE.
2. SAMSUNG.
3. BLACKBERRY.
4. Exit.

Enter your choice: 1

Iphone 6

Rs 65000.00

===== Mobile Shop =====

1. IPHONE.
2. SAMSUNG.
3. BLACKBERRY.
4. Exit.

Enter your choice: 2

Samsung galaxy tab 3

Rs 45000.00

===== Mobile Shop =====

1. IPHONE.
2. SAMSUNG.
3. BLACKBERRY.
4. Exit.

Enter your choice: 3

Blackberry Z10

Rs 55000.00

===== Mobile Shop =====

1. IPHONE.
2. SAMSUNG.
3. BLACKBERRY.
4. Exit.

Enter your choice: 4

Nothing You purchased