工厂模式

参考：<https://blog.csdn.net/jason0539/article/details/44976775>

<https://blog.csdn.net/yuanzeyao/article/details/26062579>

1. 简单工厂模式：一个工厂生产多种产品；

public interface Product {  
 void execute();  
}  
  
public class MemoryChip implements Product {  
 @Override  
 public void execute() {  
 System.*out*.println("生产内存条");  
 }  
}  
  
public class Displayer implements Product {  
 @Override  
 public void execute() {  
 System.*out*.println("生产显示器");  
 }  
}  
  
public class SimpleFactory {  
 public static Product createProduct(String productName) {  
  
 Product product = null;  
 switch(productName) {  
 case "memoryChip":  
 product = new MemoryChip();  
 break;  
 case "displayer":  
 product = new Displayer();  
 break;  
 }  
 return product;  
 }  
}

public static void main(String[] args) throws IOException {  
 //从控制台获取用户需求  
 BufferedReader br=new BufferedReader(new InputStreamReader(System.*in*));  
 String input=br.readLine();  
  
 Product product = SimpleFactory.*createProduct*(input);  
 product.execute();  
}

1. 工厂方法模式：一个工厂只生产一种产品，工厂实现同一个接口；

public interface Product {  
 void execute();  
}  
  
public class MemoryChip implements Product {  
 @Override  
 public void execute() {  
 System.*out*.println("生产内存条");  
 }  
}  
  
public class Displayer implements Product {  
 @Override  
 public void execute() {  
 System.*out*.println("生产显示器");  
 }  
}  
  
private interface IFactory {  
 Product createProduct();  
}  
  
public class MemoryChipFactory implements IFactory {  
 @Override  
 public Product createProduct() {  
 return new MemoryChip();  
 }  
}  
  
public class DisplayerFactory implements IFactory {  
 @Override  
 public Product createProduct() {  
 return new Displayer();  
 }  
}  
  
public static void main(String[] args) throws IOException {  
 //从控制台获取用户需求  
 BufferedReader br=new BufferedReader(new InputStreamReader(System.*in*));  
 String input=br.readLine();  
  
 IFactory iFactory = null;  
 switch (input) {  
 case "memoryChip":  
 iFactory = new MemoryChipFactory();  
 break;  
 case "displayer":  
 iFactory = new DisplayerFactory();  
 break;  
 }  
  
 Product product = iFactory.createProduct();  
 product.execute();  
}

1. 抽象工厂模式：一个工厂生产多种产品，工厂实现同一个接口。

public interface Product {  
 void execute();  
}  
  
public static class SamsungMemoryChip implements Product {  
 @Override  
 public void execute() {  
 System.*out*.println("生产三星内存条");  
 }  
}  
  
public static class SamsungDisplayer implements Product {  
 @Override  
 public void execute() {  
 System.*out*.println("生产三星显示器");  
 }  
}  
  
public static class LGMemoryChip implements Product {  
 @Override  
 public void execute() {  
 System.*out*.println("生产LG内存条");  
 }  
}  
  
public static class LGDisplayer implements Product {  
 @Override  
 public void execute() {  
 System.*out*.println("生产LG显示器");  
 }  
}  
  
private interface IFactory {  
 Product createDisplayer();  
 Product createMemoryChip();  
}  
  
public class SamsungFactory implements IFactory {  
 @Override  
 public Product createMemoryChip() {  
 return new SamsungMemoryChip();  
 }  
  
 @Override  
 public Product createDisplayer() {  
 return new SamsungDisplayer();  
 }  
}  
  
public class LGFactory implements IFactory {  
 @Override  
 public Product createMemoryChip() {  
 return new LGMemoryChip();  
 }  
  
 @Override  
 public Product createDisplayer() {  
 return new LGDisplayer();  
 }  
}  
  
public static void main(String[] args) throws IOException {  
 //从控制台获取用户需求  
 BufferedReader br=new BufferedReader(new InputStreamReader(System.*in*));  
 String input=br.readLine();  
 String[] inputs=input.split("\_");  
  
 IFactory iFactory = null;  
 Product product = null;  
 switch (inputs[0]) {  
 case "Samsung":  
 iFactory = new SamsungFactory();  
 break;  
 case "LG":  
 iFactory = new LGFactory();  
 break;  
 }  
  
 switch (inputs[1]) {  
 case "memoryChip":  
 product = iFactory.createMemoryChip();  
 break;  
 case "displayer":  
 product = iFactory.createDisplayer();  
 break;  
 }  
  
 product.execute();  
}