### 改进的工厂模式

public interface Fruit {
  
 void eat();
  
}
  
  
public class Apple implements Fruit {
  
 @Override
  
 public void eat() {
  
 System.out.println("吃苹果");
  
 }
  
}
  
  
public class Banana implements Fruit {
  
 @Override
  
 public void eat() {
  
 System.out.println("吃香蕉");
  
 }
  
}
  
  
public class Orange implements Fruit {
  
 @Override
  
 public void eat() {
  
 System.out.println("吃橘子");
  
 }
  
}
  
  
public class Factory {
  
 // 获取当前类的包名
  
 private static final String PACKAGE\_NAME = Factory.class.getPackage().getName() + ".";
  
  
 public static Fruit getInstance(String fruitName) throws Exception {
  
 // 忽略大小写
  
 fruitName = fruitName.toLowerCase();
  
 // 首字母大写
  
 fruitName = fruitName.substring(0, 1).toUpperCase() + fruitName.substring(1);
  
  
 Class<? extends Fruit> clazz = Class.forName(PACKAGE\_NAME + fruitName)
  
 .asSubclass(Fruit.class);
  
 Fruit fruit = clazz.getConstructor().newInstance();
  
 return fruit;
  
 }
  
  
 public static void main(String[] args) {
  
 try {
  
 Factory.getInstance("apple").eat();
  
 Factory.getInstance("Banana").eat();
  
 Factory.getInstance("ORANGE").eat();
  
 Factory.getInstance("peach").eat();
  
 } catch (Exception e) {
  
 System.out.println(e);
  
 }
  
 }
  
}

### 反射实现Demo类

class Person {
  
 String name;
  
 int age;
  
 String sex;
  
  
 public Person() {
  
 }
  
  
 public Person(String name, int age, String sex) {
  
 super();
  
 this.name = name;
  
 this.age = age;
  
 this.sex = sex;
  
 }
  
  
 public String toString() {
  
 return "Student [name=" + name + ", age=" + age + ", sex=" + sex + "]";
  
 }
  
  
}
  
  
import java.lang.reflect.Field;
  
  
public class Demo {
  
 // 获取当前类的包名
  
 private static final String PACKAGE\_NAME = Demo.class.getPackage().getName() + ".";
  
  
 // 将obj对象转化成为字符串的序列化方法
  
 public static String toString(Object obj) {
  
 StringBuilder sb = new StringBuilder();
  
 Class<?> objClass = obj.getClass();
  
 Field[] fields = objClass.getDeclaredFields();
  
  
 for (int i = 0; i < fields.length; i++) {
  
 fields[i].setAccessible(true);
  
 try {
  
 String fieldName = fields[i].getName();
  
 Object fieldValue = fields[i].get(obj);
  
 sb.append(fieldName).append("=").append(fieldValue);
  
 if (i < fields.length - 1) {
  
 sb.append(", ");
  
 }
  
 } catch (IllegalAccessException e) {
  
 e.printStackTrace();
  
 }
  
 }
  
  
 return objClass.getSimpleName() + " [" + sb.toString() + "]";
  
 }
  
  
 // 将字符串转换为对象的反序列化方法
  
 public static Object fromString(String str) {
  
 int classNameEnd = str.indexOf("[");
  
 String className = str.substring(0, classNameEnd).trim();
  
 try {
  
 Class<?> objClass = Class.forName(PACKAGE\_NAME + className);
  
 Object obj = objClass.getDeclaredConstructor().newInstance();
  
  
 String[] fieldPairs = str.substring(classNameEnd + 1, str.length() - 1).split(", ");
  
 for (String fieldPair : fieldPairs) {
  
 String[] parts = fieldPair.split("=");
  
 String fieldName = parts[0].trim();
  
 String fieldValue = parts[1].trim();
  
 Field field = objClass.getDeclaredField(fieldName);
  
 field.setAccessible(true);
  
  
 if (field.getType() == int.class) {
  
 field.set(obj, Integer.parseInt(fieldValue));
  
 } else {
  
 field.set(obj, fieldValue);
  
 }
  
 }
  
 return obj;
  
 } catch (Exception e) {
  
 e.printStackTrace();
  
 return null;
  
 }
  
 }
  
  
 public static void main(String[] args) {
  
 Person p1 = new Person("Lucky", 18, "female");
  
 String str = Demo.toString(p1);
  
 // System.out.println(str);
  
 Person p2 = (Person) Demo.fromString(str);
  
 System.out.println(p2);
  
 }
  
}