一个典型的复杂对象其类代码示例如下：

```
public class Product {
  private String partA; //可以是任意类型
  private String partB;
  private String partC;
  //partA的Getter方法和Setter方法省略
  //partB的Getter方法和Setter方法省略
  //partC的Getter方法和Setter方法省略
}
```

抽象建造者类中定义了产品的创建方法和返回方法，其典型代码如下：

```
public abstract class Builder {
  protected Product product = new Product();
  public abstract void buildPartA();
  public abstract void buildPartB();
  public abstract void buildPartC();

  public Product getResult() {
    return product;
  }
}
```

建造者模式的结构中还引入了一个指挥者类 Director。该类的作用主要有两个：一方面它**隔离了客户与生产过程；**另一方面它**负责控制产品的生成过程**。指挥者针对抽象建造者编程，客户端只需要知道具体建造者的类型，即可通过指挥者类调用建造者的相关方法，返回一个完整的产品对象。

指挥者类的代码示例如下：

```
public class Director{
  private Builder builder;

  public Director(Builder builder){
    this.builder = builder;
  }

  public void setBuilder(Builder builder){
    this.builder = builer;
  }

  public Product construct(){
    builder.buildPartA();
    builder.buildPartB();
    builder.buildPartC();
    return builder.getResult();
  }
}
```

客户端类代码片段：

```
...
Builder builder = new ConcreteBuilder();
Director director = new Director(builder);
Product product = director.construct();
...
```

在客户端代码中，无须关心产品对象的具体组装过程，只需确定具体建造者的类型即可，建造者模式将复杂对象的构建与对象的表现分离开来，这样使得同样的构建过程可以创建出不同的表现。