**Template Method Design Pattern**

Template Method is a **behavioral**[**design pattern**](https://www.journaldev.com/1827/java-design-patterns-example-tutorial). Template Method design pattern is used to create a method stub and deferring some of the steps of implementation to the subclasses. Template method defines the steps to execute an algorithm and it can provide default implementation that might be common for all or some of the subclasses.

Let’s understand this pattern with an example, suppose we want to provide an algorithm to build a house. The steps need to be performed to build a house are – building foundation, building pillars, building walls and windows. The important point is that the we can’t change the order of execution because we can’t build windows before building the foundation. So in this case we can create a template method that will use different methods to build the house.

Now building the foundation for a house is same for all type of houses, whether its a wooden house or a glass house. So we can provide base implementation for this, if subclasses want to override this method, they can but mostly it’s common for all the types of houses.

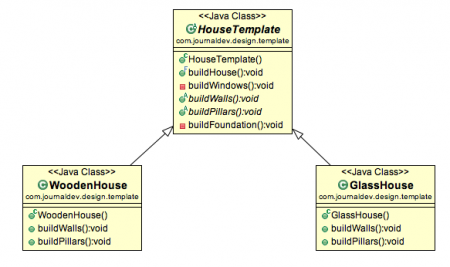
To make sure that subclasses don’t override the template method, we should make it final.

### Template Method Abstract Class

Since we want some of the methods to be implemented by subclasses, we have to make our base class as [abstract class](https://www.journaldev.com/1582/abstract-class-in-java).

Notice that client is invoking the template method of base class and depending of implementation of different steps, it’s using some of the methods from base class and some of them from subclass.

### Template Method Class Diagram

[](https://cdn.journaldev.com/wp-content/uploads/2013/07/template-method-pattern.png)

### Template Method Design Pattern in JDK

* All non-abstract methods of java.io.InputStream, java.io.OutputStream, java.io.Reader and java.io.Writer.
* All non-abstract methods of java.util.AbstractList, java.util.AbstractSet and java.util.AbstractMap.

### Template Method Design Pattern Important Points

1. Template method should consists of certain steps whose order is fixed and for some of the methods, implementation differs from base class to subclass. Template method should be final.
2. Most of the times, subclasses calls methods from super class but in template pattern, superclass template method calls methods from subclasses, this is known as [Hollywood Principle](https://en.wikipedia.org/wiki/Hollywood_principle) – “don’t call us, we’ll call you.”.
3. Methods in base class with default implementation are referred as **Hooks** and they are intended to be overridden by subclasses, if you want some of the methods to be not overridden, you can make them final, for example in our case we can make buildFoundation() method final because if we don’t want subclasses to override it.