Template Pattern

What is it?

- Whenever you write a parent class where you leave one or more of the methods to be implemented by derived classes, you are in essence using the Template pattern.
- The Template pattern formalizes the idea of defining an algorithm in a class, but leaving some of the details to be implemented in subclasses.
- In other words, if your base class is an abstract class, as often happens in these design patterns, you are using a simple form of the Template pattern.

What is it?

- The idea behind the Template pattern is that some parts of an algorithm are well defined and can be implemented in the base class, while other parts may have several implementations and are best left to derived classes.
- Another main theme is recognizing that there are some basic parts of a class that can be factored out and put in a base class so that they do not need to be repeated in several subclasses.

Types of Methods in a Template

- A Template has four kinds of methods that you can make use of in derive classes:
  1. Complete methods that carry out some basic function that all the subclasses will want to use. These are called *Concrete methods.*
  2. Methods that are not filled in at all and must be implemented in derived classes. In Java , you would declare these as *abstract* methods.
  3. Methods that contain a default implementation of some operations, but which may be overridden in derived classes. These are called *Hook* methods. Of course, in Java you can override any public or protected method in the derived class, but Hook methods are intended to be overridden, while Concrete methods are not.
  4. Finally, a Template class may contain methods which themselves call any combination of abstract, hook and concrete methods. These methods are not intended to be overridden, but describe an algorithm without actually implementing its details. They are called Template methods.