**Scenario 2.2 (Factory Design Pattern)**

**Definition**

The **factory method pattern** is a [creational pattern](https://en.wikipedia.org/wiki/Creational_pattern) that uses factory methods to deal with the problem of [creating objects](https://en.wikipedia.org/wiki/Object_creation) without having to specify the exact [class](https://en.wikipedia.org/wiki/Class_(computer_programming)) of the object that will be created.

This is done by creating objects by calling a factory method—either specified in an [interface](https://en.wikipedia.org/wiki/Interface_(object-oriented_programming)) and implemented by child classes, or implemented in a base class and optionally [overridden](https://en.wikipedia.org/wiki/Method_overriding) by derived classes—rather than by calling a [constructor](https://en.wikipedia.org/wiki/Constructor_(object-oriented_programming)).

**Where We Require Factory Pattern?**

The factory design pattern is used **when we have a superclass with multiple sub-classes and based on input, we need to return one of the sub-class**.

This pattern takes out the responsibility of the instantiation of a class from the client program to the factory class.

In our case, we have a superclass named Event and sub-classes named BookingTheEvent and HostingTheEvent.

**Step 1: Create an interface event**

**Text

Description automatically generated**

**Step 2: Create concrete classes implementing the same interface.**

Text

Description automatically generated

Text

Description automatically generated

**Step 3: Create a Factory to generate object of concrete class based on given information.**

**Text

Description automatically generated**

**Step 4: Use the Factory to get object of concrete class by passing an information such as type**.

Text

Description automatically generated