Design and implement an Event Handling Framework. It handles pre-defined set of events (such as Login, Logout) and notifies Listeners who are interested in getting notification. An Event Handling framework consists of following key interfaces –

**EventManager – *EventManager is main contract for registering, unregistering and notifying listeners. One Event type can have many listeners.***

/\*\*

\* Simple Contract for Event framework. This includes contracts for Registration of listeners, Unregistration of listeners

\* and notifying listeners when events are updated.

\*/

**public** **interface** EventManager {

/\*\*

\* Contract for registration of Listeners.

\*

\* **@param** eventType Type of event for which Listeners are interested in getting notification

\* **@param** eventNotificationType Notification type - synchronous vs asynchronous

\* **@param** eventListener Listener which will get notification

\*/

**public** **void** registerListener( EventType eventType, EventNotificationType eventNotificationType,

EventListener eventListener );

/\*\*

\* Contract for listeners to unregister.

\*

\* **@param** eventType Type of event for which Listeners are interested in getting notification

\* **@param** listener Listener which will get notification

\*/

**public** **void** unregisterListener( EventType eventType, EventListener listener );

/\*\*

\* Contract for client to provide notification to all the listeners who are registered for particular event type.

\*

\* **@param** event Event for which notification is received

\* **@throws** EventException Exception thrown in case of error

\*/

**public** **void** fireEvent( Event<?> event ) **throws** EventException;

}

**Event – *An Event can have information about type and it can contain data*.**

/\*\*

\* Contract for event objects. A event object provides information about Type of event and event data.

\*

\* **@param** <T> The type of Even Data

\*/

**public** **interface** Event<T> {

/\*\*

\* Type of events. This is nothing but predefined events types which system expects (such as Login, Logout etc)

\*

\* **@return**

\*/

**public** EventType getEventType();

/\*\*

\* Data about events.

\*

\* **@return** Event Data

\*/

**public** T getEventData();

}

**EventListener – *Listeners acts on Event after getting notification.***

/\*\*

\* This interface defines the contract to be implemented by event listeners which want to be notified

\* about events. Event listeners can be notified synchronously or asynchronously, which is specified when

\* listeners are registered for events.

\*/

**public** **interface** EventListener {

/\*\*

\* Callback method used by event framework to notify the listener about some event.

\*

\* **@param** event event object

\*/

**public** **void** onEvent( Event<?> event ) **throws** EventException;

}

**Expectations:**

1. Implement an Event handling framework which handles *Login Event*. There are 2 kinds of Listeners which want to subscribe for Login events –
   1. SessionLoginEventListener – this class just prints “Logging data stored to session”. This Listener is notified serially (onEvent(..) is called in same thread as main thread)
   2. LogLoginEventListener – this class just prints “Login information logged”. This Listener is notified in parallel (onEvent(..) is called in different thread than main thread)
2. Implement main method in which create EventManager, register above 2 listeners and fire events on LoginEvent class.
3. Design point to keep in mind – System will have only one Event Manager.