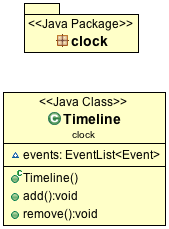
**Documentation**

**Packages**

* [clock](https://github.com/hhsnopek/csc341-final/tree/master/docs#clock)
  + [Timeline](https://github.com/hhsnopek/csc341-final/tree/master/docs#Timeline)
* [event](https://github.com/hhsnopek/csc341-final/tree/master/docs#event)
  + [Event](https://github.com/hhsnopek/csc341-final/tree/master/docs#Event)
  + [EventFactory](https://github.com/hhsnopek/csc341-final/tree/master/docs#EventFactory)
  + [EventList](https://github.com/hhsnopek/csc341-final/tree/master/docs#EventList)
  + [Alarm](https://github.com/hhsnopek/csc341-final/tree/master/docs#Alarm)
  + [Alert](https://github.com/hhsnopek/csc341-final/tree/master/docs#Alert)
  + [Reminder](https://github.com/hhsnopek/csc341-final/tree/master/docs#Reminder)
* [record](https://github.com/hhsnopek/csc341-final/tree/master/docs#record)
  + [Record](https://github.com/hhsnopek/csc341-final/tree/master/docs#Record)
  + [Records](https://github.com/hhsnopek/csc341-final/tree/master/docs#Records)
* [report](https://github.com/hhsnopek/csc341-final/tree/master/docs#report)
  + [Report](https://github.com/hhsnopek/csc341-final/tree/master/docs#Report)
  + [ReportFactory](https://github.com/hhsnopek/csc341-final/tree/master/docs#ReportFactory)
  + [SleepReport](https://github.com/hhsnopek/csc341-final/tree/master/docs#SleepReport)

[**clock**](https://github.com/hhsnopek/csc341-final/blob/master/src/clock)

UML: [](https://github.com/hhsnopek/csc341-final/blob/master/docs/uml/clock.png)

[**Timeline**](https://github.com/hhsnopek/csc341-final/blob/master/src/clock/Timeline.java)

clock.Timeline is where the clock is initialized, counts, and holds all upcoming events. Think of the Timeline as a real timeline, only [events](https://github.com/hhsnopek/csc341-final/tree/master/docs#event): [event.Alarm](https://github.com/hhsnopek/csc341-final/tree/master/docs" \l "Alarm), [event.Alert](https://github.com/hhsnopek/csc341-final/tree/master/docs" \l "Alert), & [event.Reminder](https://github.com/hhsnopek/csc341-final/tree/master/docs" \l "Reminder) are added to the timeline. After an event is triggered it is removed from the timeline, unless the event has specified to reoccur.

Timeline + - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - +

Hour 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24

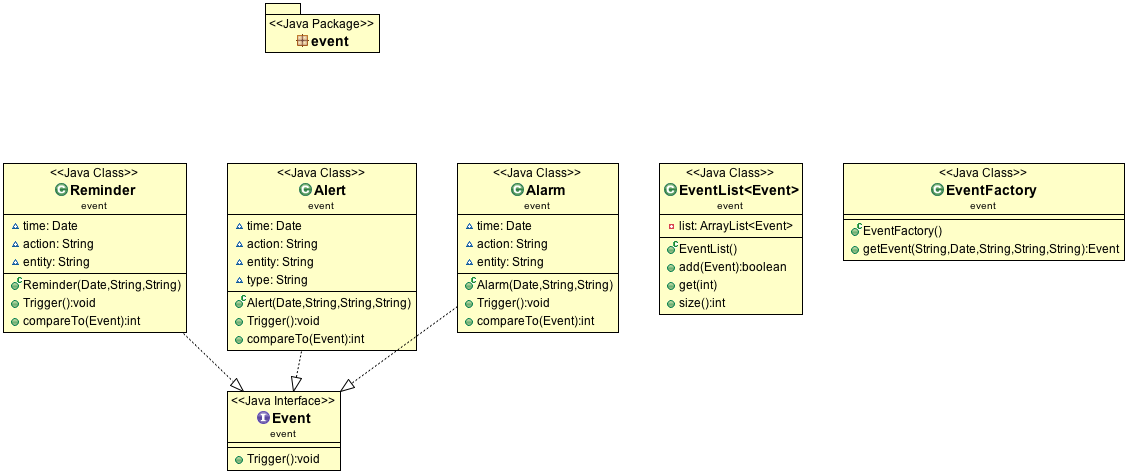
Event Alarm Reminder Alert

**Supported methods:**

* add(Event):void
* remove(Event):void

[**event**](https://github.com/hhsnopek/csc341-final/blob/master/src/event)

The event package contains all the events: [Alarm](https://github.com/hhsnopek/csc341-final/tree/master/docs#Alarm), [Alert](https://github.com/hhsnopek/csc341-final/tree/master/docs#Alert), & [Reminder](https://github.com/hhsnopek/csc341-final/tree/master/docs#Reminder); which can be added to [clock.Timeline](https://github.com/hhsnopek/csc341-final/tree/master/docs" \l "Timeline). To create a new event, create the [EventFactory](https://github.com/hhsnopek/csc341-final/tree/master/docs" \l "EventFactory) and specify the event you'd like to create.

UML: [](https://github.com/hhsnopek/csc341-final/blob/master/docs/uml/event.png)

[**Event**](https://github.com/hhsnopek/csc341-final/blob/master/src/event/Event.java)

This is the interface that is used to create [Alarm](https://github.com/hhsnopek/csc341-final/tree/master/docs#Alarm), [Alert](https://github.com/hhsnopek/csc341-final/tree/master/docs#Alert), & [Reminder](https://github.com/hhsnopek/csc341-final/tree/master/docs#Reminder). Each Event will have a trigger that sends out anActionEvent. This allows your interface to listen to the timeline events. If you're to create your own event it must trigger anActionEvent. The recommended practice is to add your event before it is suppose to be triggered. You may, but not suggested, call the ActionEvent's separate from the event. The action will still be recorded in [record.Records](https://github.com/hhsnopek/csc341-final/tree/master/docs" \l "Records)

**Must implement methods:**

* trigger():void
* compareTo():int

[**EventFactory**](https://github.com/hhsnopek/csc341-final/blob/master/src/event/EventFactory.java)

This is where you'll want to create your Events that will be added to [clock.Timeline](https://github.com/hhsnopek/csc341-final/tree/master/docs" \l "Timeline). You may extend this class to add custom Events that aren't supported by default.

[**EventList**](https://github.com/hhsnopek/csc341-final/blob/master/src/event/EventList.java)

This is what the Timeline uses to hold all the Events for the future. It will hold any amount and always sort the events to the closest current time.

[**Alarm**](https://github.com/hhsnopek/csc341-final/blob/master/src/event/Alarm.java)

This is one of three default events, this triggers the Ring event. You must pass three parameters into the class: Date time,String action, & String entity.

[**Alert**](https://github.com/hhsnopek/csc341-final/blob/master/src/event/Alert.java)

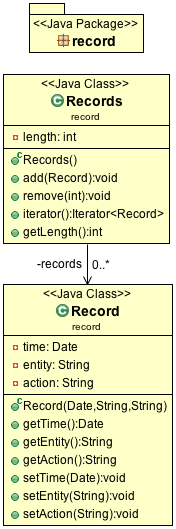
This is the second of three default events, this triggers the Notify event and also trigger the type of event. The type is either RED or YELLOW (case insensitive). You must pass four parameters into the class: Date time, String action, String entity & String type.

[**Reminder**](https://github.com/hhsnopek/csc341-final/blob/master/src/event/Reminder.java)

This is the third of the default events, this triggers the Ring event. You must pass three parameters into the class: Date time, String action, & String entity.

[**record**](https://github.com/hhsnopek/csc341-final/blob/master/src/record)

The record package holds the all the "events" that take place during the clock. These events are different fromclock.Timeline & event.Event events. The recorded events should be any ActionEvent's, User, Clock, and API/Server interaction.

UML: [](https://github.com/hhsnopek/csc341-final/blob/master/docs/uml/record.png)

[**Record**](https://github.com/hhsnopek/csc341-final/blob/master/src/record/Record.java)

The record holds any events that take place during the clocks duration of being active. As stated before this includes anyActionEvent's, User, Clock, and API/Server interaction. Record takes three parameters: Date time, String entity, &String action. See example below for what to specify within the Action.

**Note:** On creation of a Record, you can only read the record afterwards. The attempt at setting anything will throw anUnsupportedOperationException.

13:00 - User, Add Alarm for 8:00

**Supported methods:**

* getTime():Date
* getEntity():String
* getAction():String

[**Records**](https://github.com/hhsnopek/csc341-final/blob/master/src/record/Records.java)

Records is what holds all your Records.

**Supported methods:**

* add(Record):void
* remove(int):void
* iterator():Iterator
* getLength():int

Time - Entity, Action

13:00 - User, Add Alarm for 8:00

13:03 - User, Add Reminder for 18:00

16:00 - API, Add Alert for 19:00

18:00 - Clock, Reminder Executes

18:10 - User, Enters Sleep

...

7:30 - User, Exits Sleep

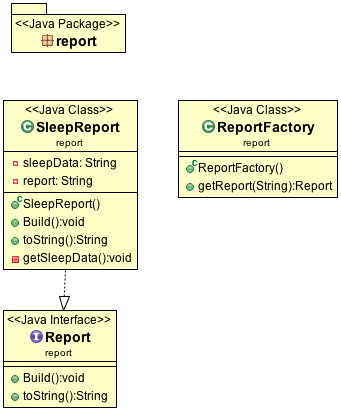
8:00 - Clock, Alarm Executes

8:01 - User, Stops Alarm

...

[**report**](https://github.com/hhsnopek/csc341-final/blob/master/src/report)

The report package contains everything you'll need to generate and build reports that are requested from the User. The Assumption here is that the Researchers have their own way of generating reports that are specific for themselves. This package only contains reports for the User.

UML: [](https://github.com/hhsnopek/csc341-final/blob/master/docs/uml/report.png)

[**Report**](https://github.com/hhsnopek/csc341-final/blob/master/src/report/Report.java)

This is the interface that you can use to build custom reports. You must have build():void and toString():String methods. When a report is created it must trigger an ActionEvent and called Generating Report [reportName]. Within your report you must request all data from the researchers, as all records are stored on their side.

[**ReportFactory**](https://github.com/hhsnopek/csc341-final/blob/master/src/report/ReportFactory.java)

This is where you'll request all your reports that you'd like to be created. It is preferred when adding custom reports to extend this class.

[**SleepReport**](https://github.com/hhsnopek/csc341-final/blob/master/src/report/SleepReport.java)

This is the only default report that is provided. This will grab all the sleeping data: Sleep Time, Alerts to Sleep, Wake up Alarms, Wake up Time, and a bunch more information.