Brahma App

Sam Kim and Andrew Siegle

# Repository

www.github.com/soggynoose/BrahmaApp.git

# Scrums

## Scrum Week 1

* Planning
* UML
* Week 2 planning

## Scrum Week 2

* Begin implementation
* Develop interfaces and Abstract classes

## Scrum Week 3

* View
* Plugin Loader
* Events and event listener

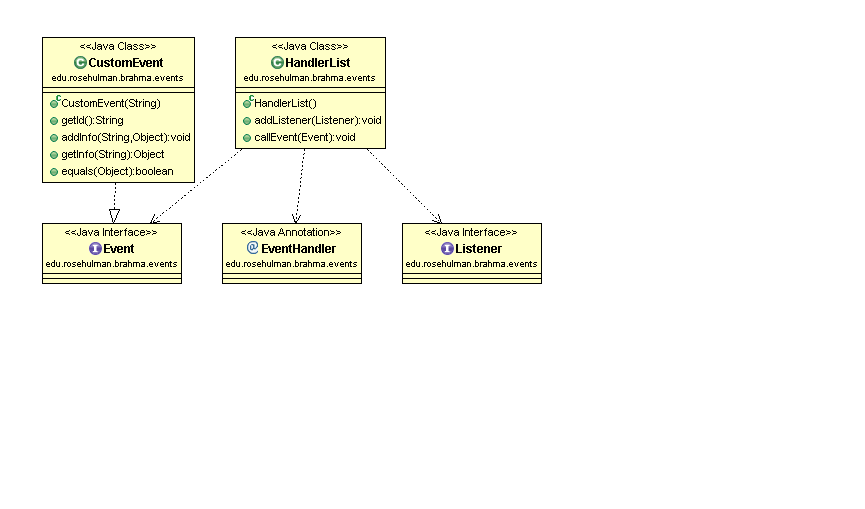
# Design

## First Draft

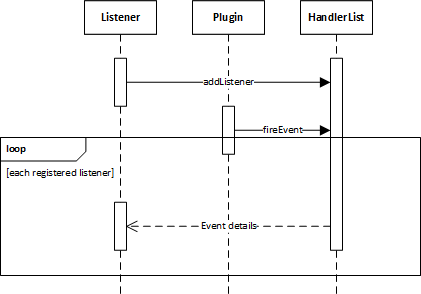
## End Result

### Brahma Package Diagram

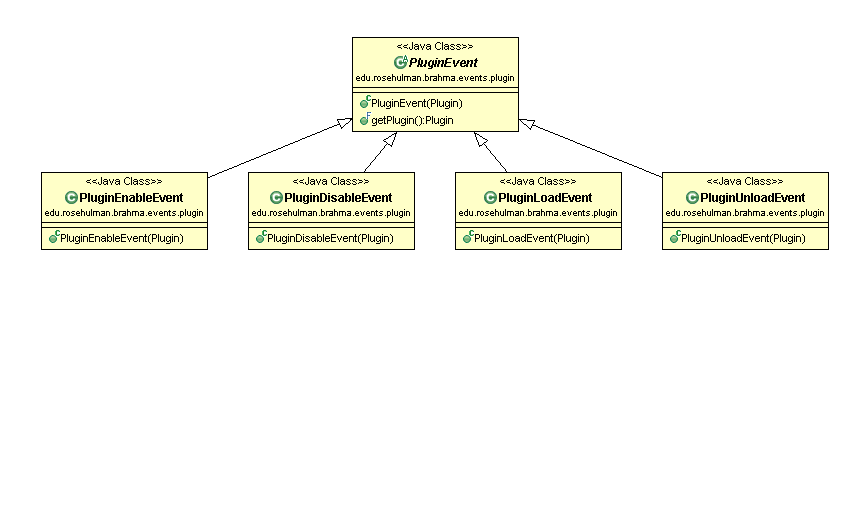
### Event Package Diagram



### Event Sequence Diagram



### Plugin Event Package



# Extensibility

Our Brahma app is very extensible because of the plugin architecture. A user needs only to create a program and compile it into a jar file implementing the Plugin interface supplied with the app and put it in the plugins folder in order for the plugin manager to recognize the plug in and load it into the system. We also allow the users to create new Events to be used with our event handler which notifies all listeners and allows for user-created events.

# Modifiability

We added some modifications for modifiability such as being able to declare the directory path of the plugins directory so you can have multiple plugin mangers watching multiple directories.

# Testability

Most subsystems of the app extend an abstract class of interface which makes the app very testable. Known interfaces and abstractions makes it possible to implement testing code. The plugin architecture mixed with the event-driven architecture makes it easy to create testing plugins which can interact with other plugins.

# Fault Tolerance

Each plugin is self-contained and the plugin manager is equipped fault control during the most vulnerable parts such as loading and starting plugins.

# Usability

The app is extremely usable in the sense that it loads all of the plugins from the specified folder and lists all of the available plugins in the Brahma view window. The plugins can also run in the background, so if you start running one plugin and switch to another, the first one continues running until you switch back to it. All the user needs to do to add a plugin is create a jar and add it to the plugins folder.

# Coolness

The cool features of our app is the event-driven execution of the program and the abstraction which allows for other languages to wrap the java plugin class to create other plugins such as a Python plugin.