Event Bus and Event System

What the EventSystem is used for

Many times there is a need to provide an object on many locations simultaneously. This design pattern is quite similar to Java's Observer, but it is much simpler and readable while having the same functional sense.

Main principle

`EventBus` represents a communication line between multiple components. Objects can be passed through the bus and reach the listening method of another object which is registered on that `EventBus` instance. Hence, the passed object is available as a parameter in the listening method.

Register to the `EventBus`

Any listening method has to be annotated with `@Subscribe` keyword and must have only one accepting parameter. Furthermore, the object which contains such listening method(s) has to be registered using the `register(Object)` method provided by `EventBus`. The listening methods can be overloaded by using different parameter types.

Posting an object

`post(object)` posts an object through the `EventBus` which has been used to register the listening/subscribing methods.

Short example

```
/* Listener.java */

import com.google.common.eventbus.Subscribe;

public class Listener {

   private int value = 0;

   @Subscribe
```

```
    public void listen(int value) {
        this.value = value;
    }

    public int getValue() {
        return this.value;
    }
}
```

```
/* Main.java */

import com.google.common.eventbus.EventBus;

public class Main {
    private static EventBus eventBus = new EventBus();

    public static void main(String[] args) {
        Main main = new Main();
        Listener listener = new Listener();
        eventBus.register(listener);
        eventBus.post(1); // 1 represents the passed event

        // Output should be 1
        System.out.println(listener.getValue());
    }
}
```

Event handling in JabRef

The `event` package contains some specific events which occur in JabRef.

For example: Every time an entry was added to the database a new `EntryAddedEvent` is sent through the `eventBus` which is located in `BibDatabase`.

If you want to catch the event you'll have to register your listener class with the `registerListener(Object listener)` method in `BibDatabase`. `EntryAddedEvent` provides also methods to get the inserted `BibEntry`.
