Working with an Event Filter

Executing Drivers conditionally

org.lcsim allows you to add (Sub-)Drivers to your (Parent-)Driver.

```
add(new SubDriverClass());
```

You can put this **anywhere** in your code!

The Event loop process needs to be told to execute the child Drivers specifically. This is done with

```
super.process()
```

Again, this statement can appear **anywhere** in your code, even in conditional statements.

You can combie these statements to specifically run your Drivers only on certain events. Like so:

```
import java.util.List;

import org.lcsim.event.EventHeader;
import org.lcsim.event.MCParticle;
import org.lcsim.util.Driver;

class PrintDriver1 extends Driver {
    public void process(EventHeader e) {
        System.out.println("PrintDriver1 has been called");
    }
}

class PrintDriver2 extends Driver {
    public void process(EventHeader e) {
        System.out.println("PrintDriver2 has been called");
    }
}

public class FilterExample extends Driver {
    // Permanently add a Sub-Driver to this one
    public FilterExample() {
        add(new PrintDriver1());
    }

    public void process(EventHeader e) {
        List<MCParticle> parts = e.getMCParticles();
        System.out.println("Size: " + parts.size());
        if (parts.size() < 100) {
            // Execute all added Sub-Drivers
            super.process(e);
        } else if (parts.size() < 150) {
            // Add a Driver just for now
            PrintDriver2 p2 = new PrintDriver2();
            add(p2);
            // again, execute ALL Sub-Drivers
            super.process(e);
            // you can even remove a Driver.
            remove(p2);
        } else {
            System.out.println("None is called");
        }
    }
}
```