

SDN Lab

Experiment 4

Writing SDN Application Modules over Floodlight

Release Date: March 20, 2014

Experiment 4: Writing SDN Application Module over Floodlight.

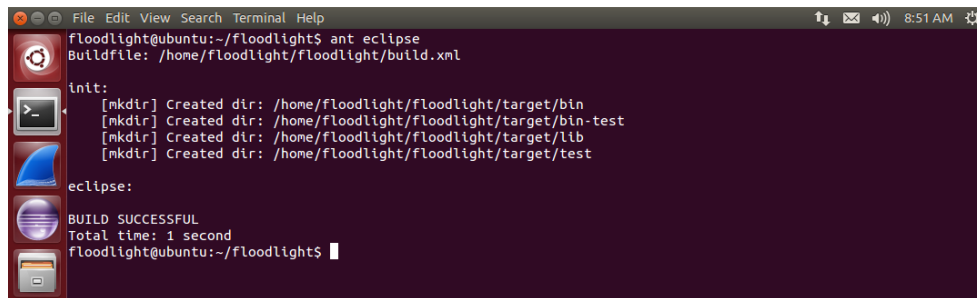
(1) Objective :

In this experiment, the user will know:

- How to create a Floodlight application module

(2) Tutorial :

- Use eclipse to write a floodlight java application
- Run “ant eclipse” to build the floodlight eclipse project file

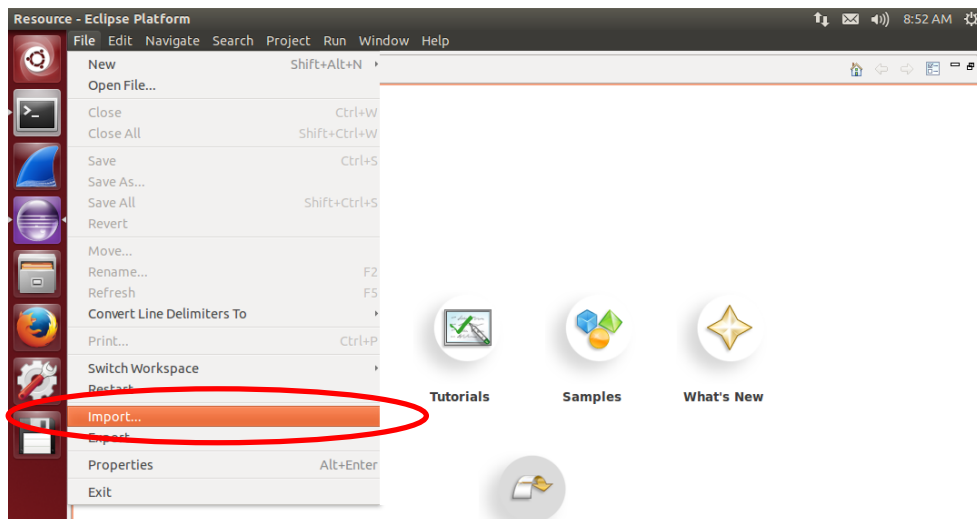


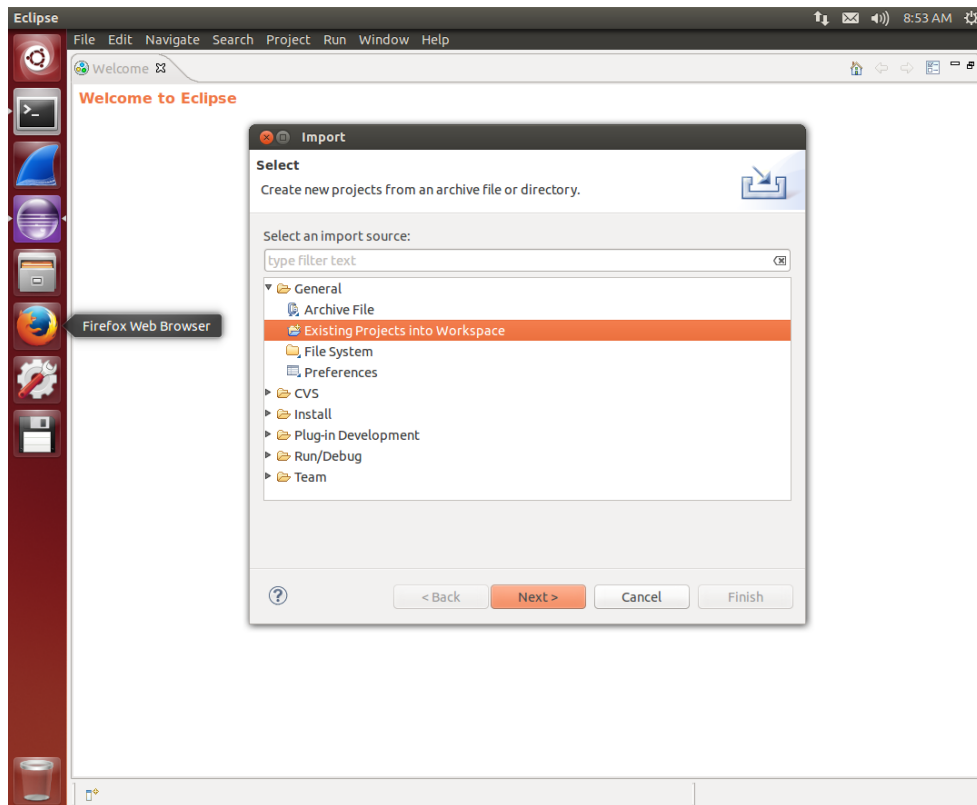
```
floodlight@ubuntu:~/floodlight$ ant eclipse
Buildfile: /home/floodlight/floodlight/build.xml

init:
[mkdir] Created dir: /home/floodlight/floodlight/target/bin
[mkdir] Created dir: /home/floodlight/floodlight/target/bin-test
[mkdir] Created dir: /home/floodlight/floodlight/target/lib
[mkdir] Created dir: /home/floodlight/floodlight/target/test

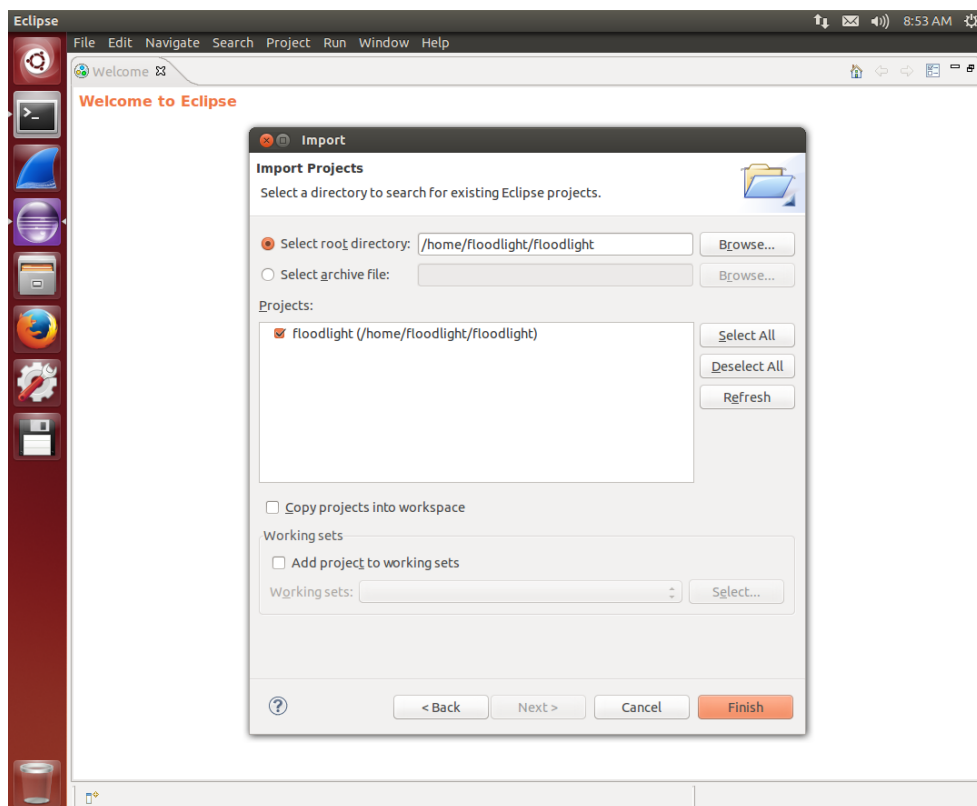
eclipse:
BUILD SUCCESSFUL
Total time: 1 second
floodlight@ubuntu:~/floodlight$
```

- Use File -> Import... -> General -> Existing Projects Into Workspace



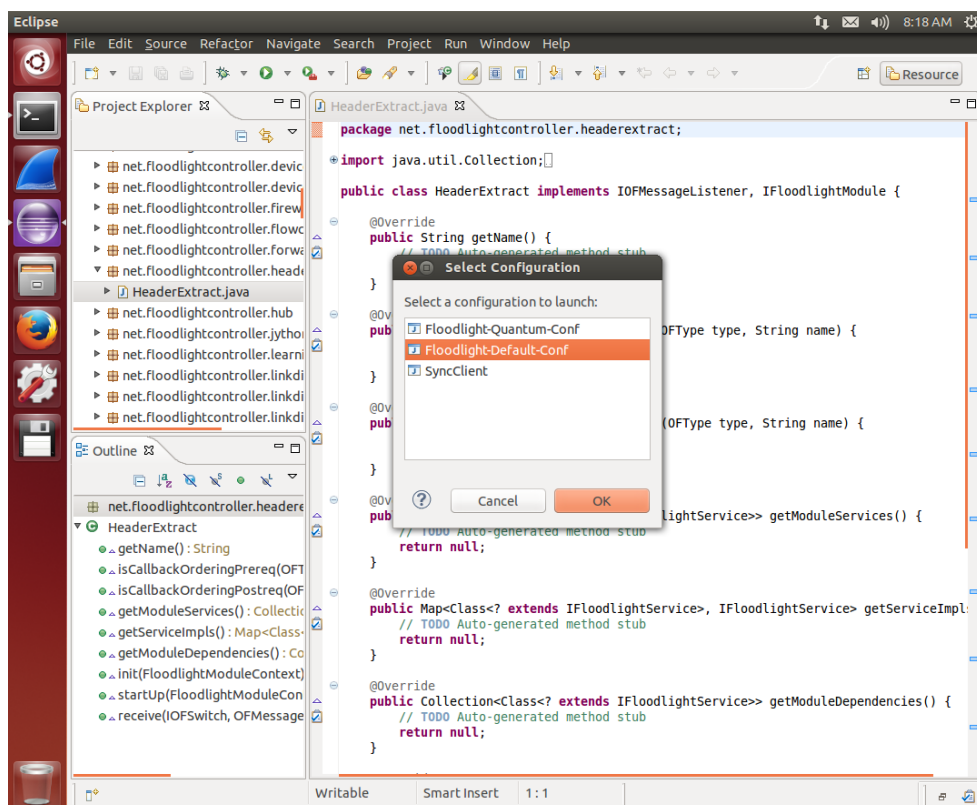
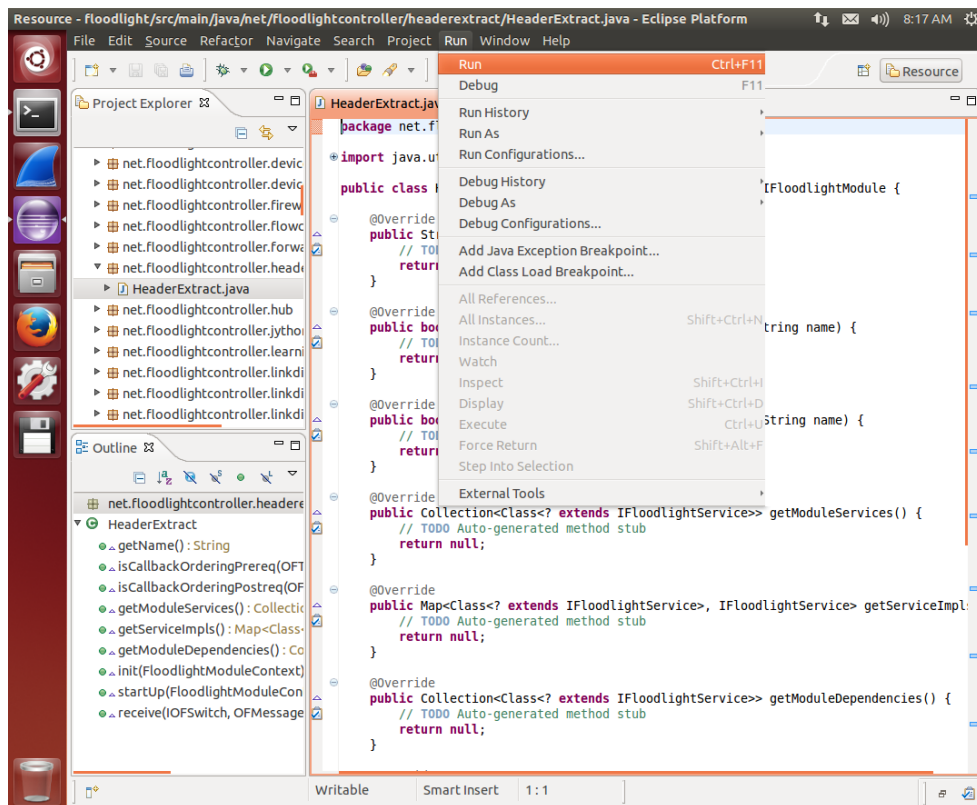


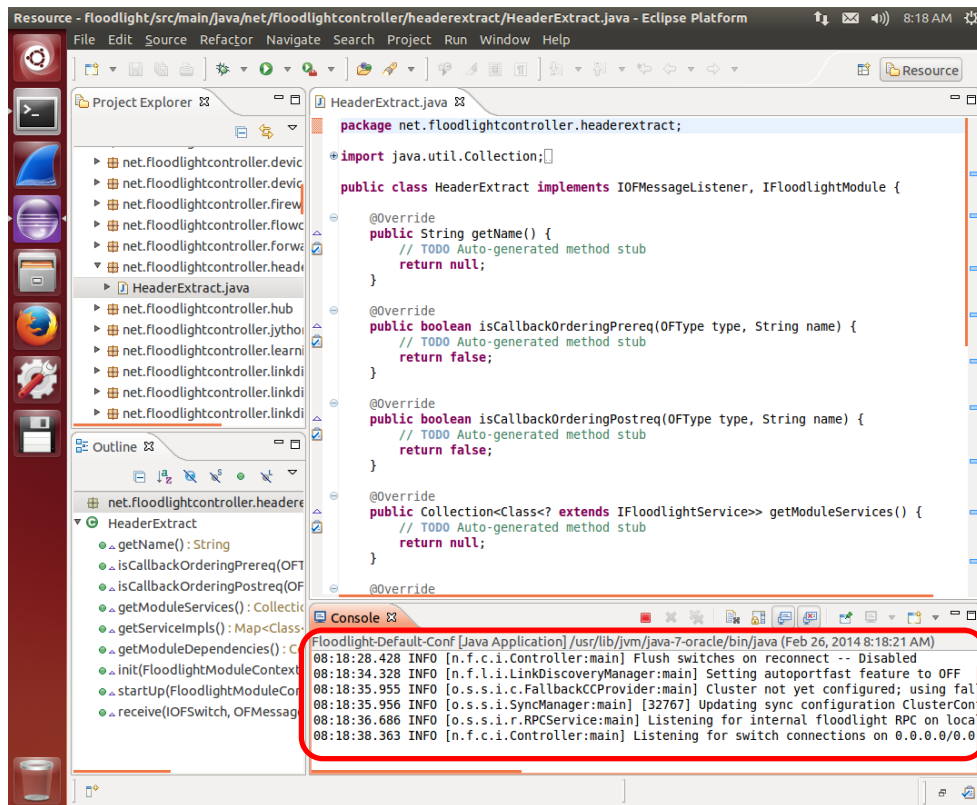
- Choose the root directory and eclipse should find the floodlight project automatically. Then press “**Finish**” to import the floodlight project.



● .

- Use “Run -> Run”, and select “Floodlight-Default-Conf” to run floodlight

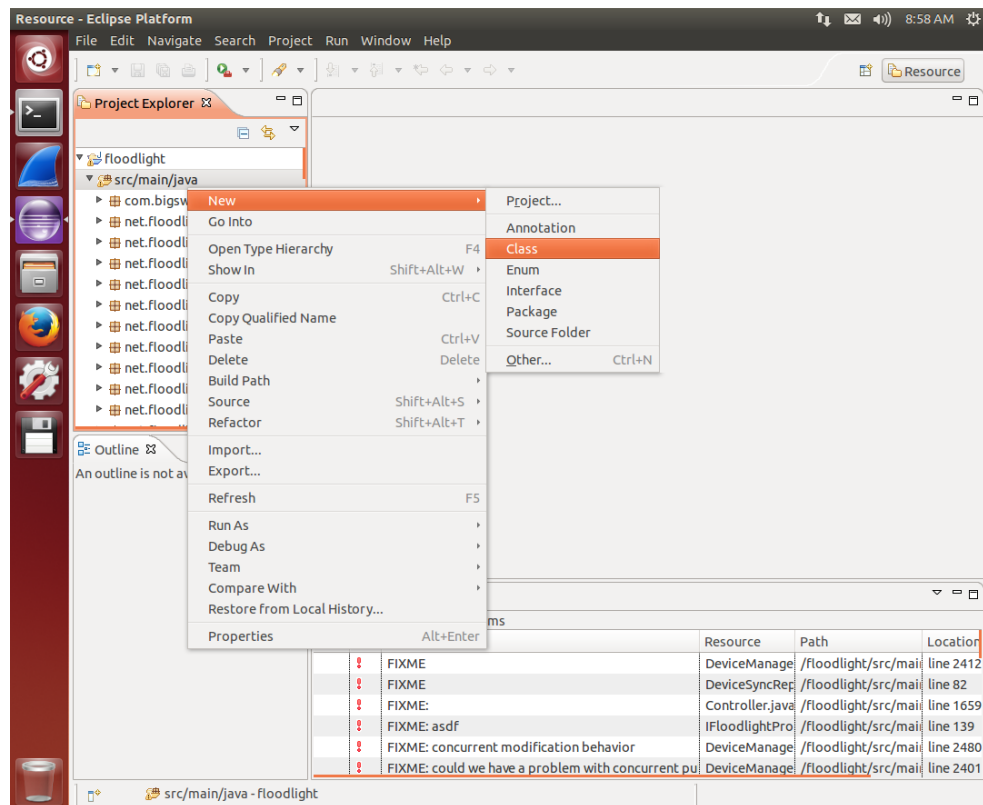




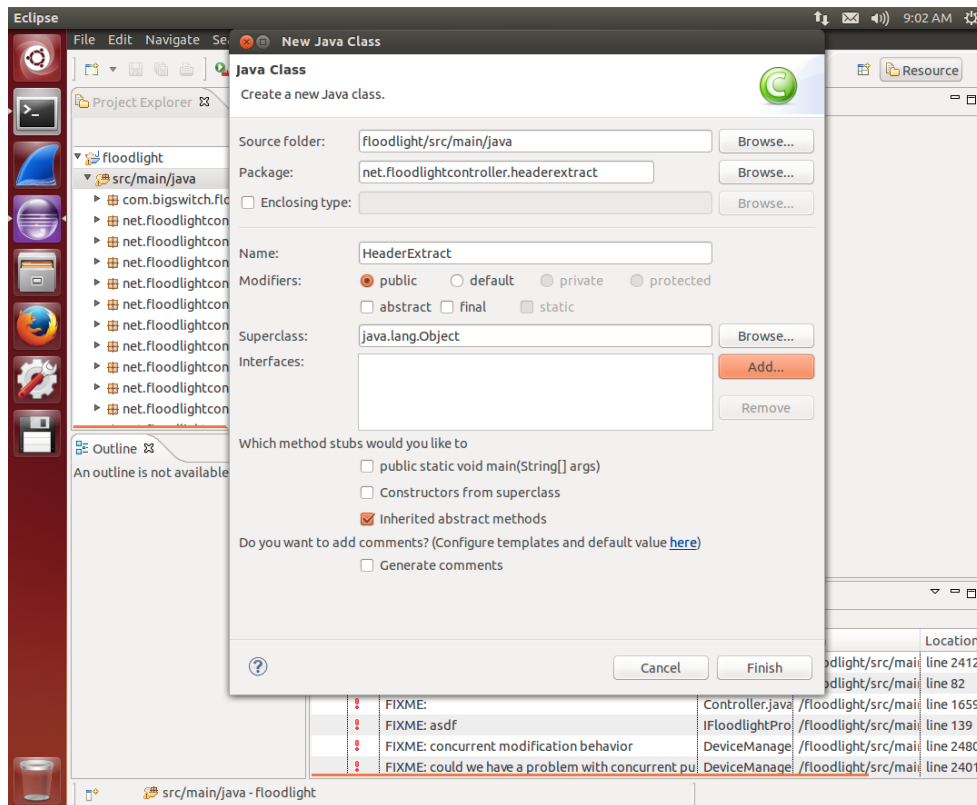
■ How to write a simple module

In this example, we'll write a simple module that will listen and receive PACKET-IN message, and then print the IP and MAC addresses of the packet carried in the PACKET-IN message.

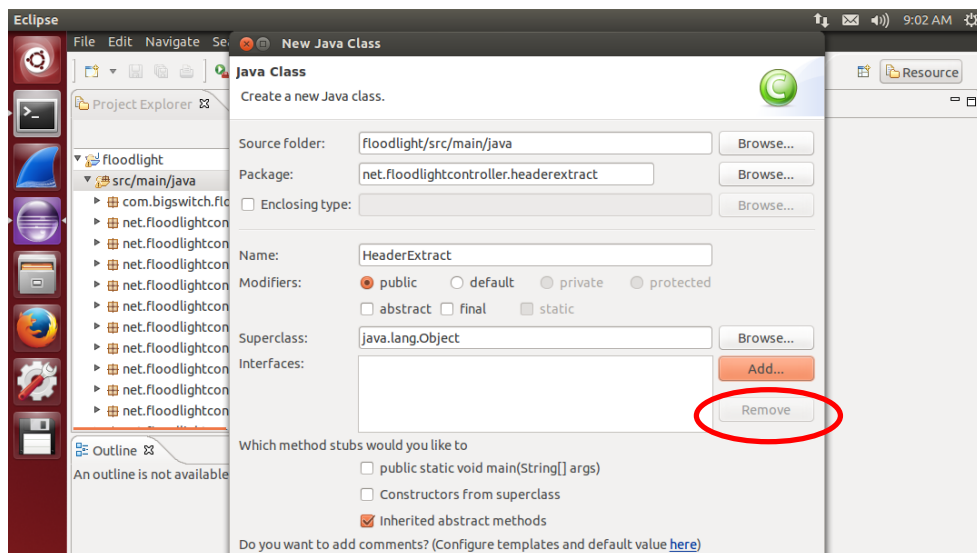
- Right click “src/main/java”, and select “New -> Class”

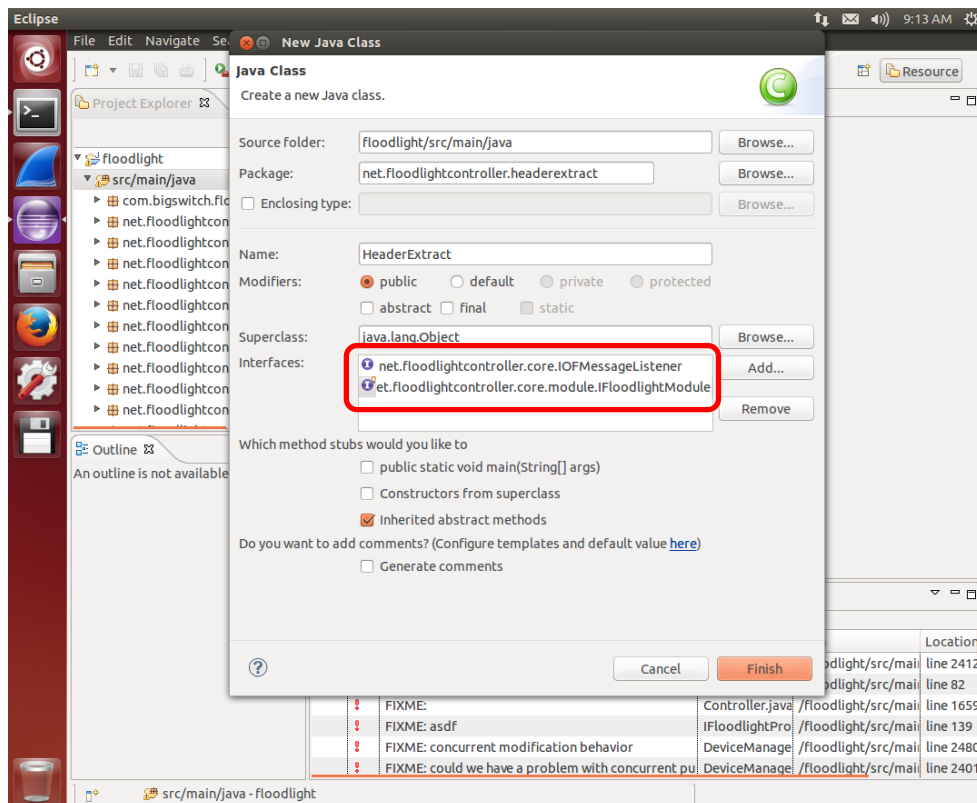
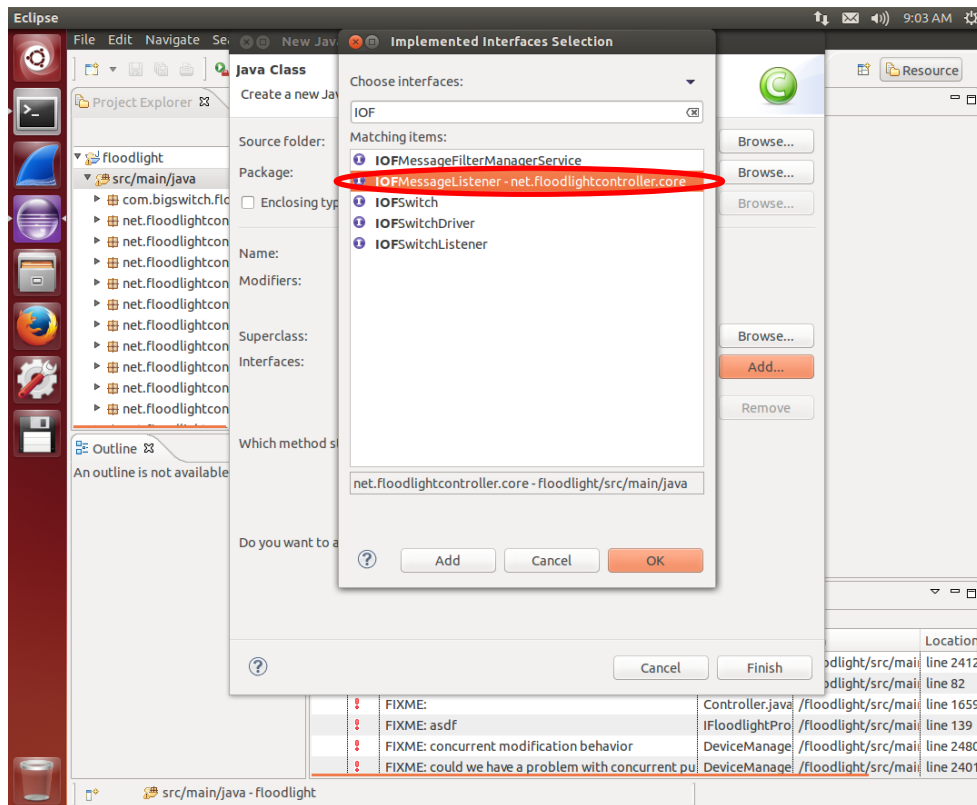


- Input the Package and class name, in this example, we use “net.floodlightcontroller.headerextract”, and “HeaderExtract”

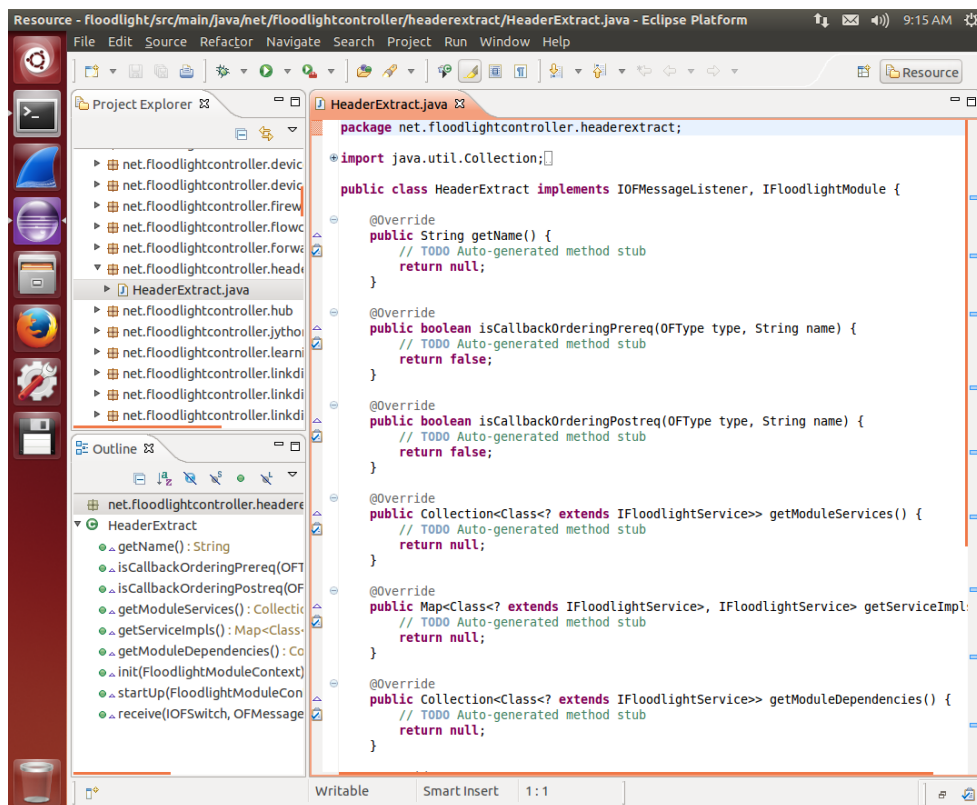


- We need to inherit “IFloodlightModule”. Because we want to “listen” switch-to-controller messages, we need to inherit “IOFMessageListener”, too.





- The default code that was automatically generated by eclipse will look like as follows:



- We can see some auto-generated methods here. They are:

i. getName

Return the name of your module to Floodlight

In HeaderExtract:

```
return "HeaderExtract";
```

ii. isCallbackOrderingPrereq

Check if the module called name is a callback ordering prerequisite for this module. In other words, if this function returns true for the given name, then this listener will be called after that message listener.

Ex: (In HeaderExtract, we do not need to modify this function)

```
return (type.equals(OFType.PACKET_IN) &&
        (name.equals("topology") ||
         name.equals("devicemanager") );
```

iii. isCallbackOrderingPostreq

Check if the module called name is a callback ordering post-requisite for this module. In other words, if this function returns true for the given name, then this listener will be called before that message

listener.,

Ex: (In HeaderExtract, we do not need to modify this function)

```
return (type.equals(OFType.PACKET_IN) &&
name.equals("forwarding"));
```

iv. getModuleServices

Return the list of interfaces that this module implements. All interfaces must inherit IFloodlightService

Ex: (In HeaderExtract, we do not need to modify this function)

```
Collection<Class<? extends IFloodlightService>> l = new
ArrayList<Class<? extends IFloodlightService>>();
l.add(ILoadbalanceRoutingService.class);
return l;
```

v. getServiceImpls

Instantiate (as needed) and return objects that implement each of the services exported by this module. The map returned maps the implemented service to the object. The object could be the same object or different objects for different exported services.

Ex: (In HeaderExtract, we do not need to modify this function)

```
Map<Class<? extends IFloodlightService>, IFloodlightService> m
= new HashMap<Class<? extends IFloodlightService>,
IFloodlightService>();
m.put(ILoadbalanceRoutingService.class, this);
return m;
```

vi. getModuleDependencies

Get a list of modules that this module depends on. The module system will ensure that each of these dependencies is resolved before the subsequent calls to init()

In HeaderExtract:

```
Collection<Class<? extends IFloodlightService>> l = new
ArrayList<Class<? extends IFloodlightService>>();
l.add(IFloodlightProviderService.class);
return l;
```

vii. init

When your module is created, Floodlight calls this method to initialize your module

In HeaderExtract:

```
floodlightProvider =  
context.getServiceImpl(IFloodlightProviderService.class);
```

viii. startUp

After your module starts, it should be prepared to process incoming data.

In HeaderExtract:

```
floodlightProvider.addOFMessageListener(OFType.PACKET_IN,  
this);
```

ix. receive

When Floodlight receives PACKET-IN, it will call this function.

- Declare the following variables in HeaderExtract

```
public final int DEFAULT_CACHE_SIZE = 10;  
protected IFloodlightProviderService floodlightProvider;  
private IStaticFlowEntryPusherService flowPusher;
```

- Add the following code into the receive() function:

```
OFPacketIn pin = (OFPacketIn) msg;  
OFMatch match = new OFMatch();  
match.loadFromPacket(pin.getPacketData(), pin.getInPort());  
System.out.println("$$$$-Get the Destination IP Address-$$$$");  
System.out.println(IPv4.fromIPv4Address(match.getNetworkDestination()));  
System.out.println("$$$$-Mac Address Destination-$$$$");  
System.out.println(HexString.toHexString(sourceMACHash));  
System.out.println("$$$$-PacketIn ARRAY-$$$$");  
System.out.println(Arrays.asList(match));  
return Command.CONTINUE;
```

- Add “**net.floodlightcontroller.headerextract.HeaderExtract**,\” to “src/main/resources/floodlightdefault.properties”, and add “**net.floodlightcontroller.headerextract.HeaderExtract**” to “src/main/resources/META-INF/services/net.floodlightcontroller.core.module.IFloodlightModule” to enable HeaderExtract module.

- The HeaderExtract will print out some information contained in the PACKET-IN message

```

$$$$-Get the Destination IP Address-$$$$
10.0.0.2
$$$$-Mac Address Destination-$$$$
00:00:ff:ff:ff:ff:ff:ff
$$$$-PacketIn ARRAY-$$$$
[OFMatch[in_port=1,dl_dst=ff:ff:ff:ff:ff:ff,dl_src=2e:e5:82:02:a
8:31,dl_type=0x806,dl_vlan=0xffff,dl_vlan_pcp=0,nw_dst=10.0.0.2,
nw_src=10.0.0.1,nw_proto=1,nw_tos=0,tp_dst=1,tp_src=0]]

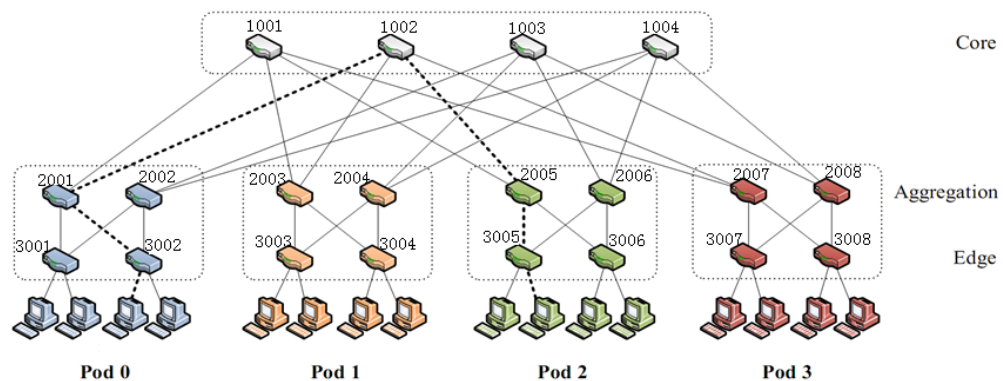
```

■ Useful modules and tips

- You can reference the “forwarding” module for the usage of MATCH, FLOW-MOD, and ACTION
- Use “IRoutingService” to get routing paths
- Use “IDeviceService” to get the list of devices

(3) Experiment :

■ Fat tree topology



- In lab 2, we create a fat tree topology with 4 pods. Each pod has 4 hosts, 2 edge switches, and 2 aggregation switches.
- Your job is to:
 1. Redirect all packets whose destination hosts are different to a single host.

In this lab, we will provide a simple UDP client and server. The server listens on port 5134 to receive packets from the client.

2. Start the UDP server on host 1 (on the leftmost) in this topology, whose IP address is "10.0.0.1"
 3. Redirect any UDP packet that is originally sent by the UDP server to any IP address and port 5134 to the IP address of 10.0.0.1
- For example, we assume the leftmost host in this topology has an IP address of 10.0.0.1, and the rightmost host has an IP address of 10.0.0.16.
- If the provided UDP client running on the rightmost host wants to send a packet to the UDP server running on a host whose IP address is 10.0.0.2, you should redirect this packet to the leftmost host (by changing the destination IP address in the packet header from 10.0.0.2 to 10.0.0.1) and let the server running on the leftmost to receive this packet.
- You do not need to consider the case of sending a packet from 10.0.0.1 to 10.0.0.1.

Bonus:

Allow your module to use REST API to accept commands from the user to specify which host to redirect to.

What you need to turn in:

1. A word file including printscreens to prove you have done the redirection of packet and a report
2. Your module file and a txt file to tell TA how to run with your module

(4) Reference :

- <http://docs.projectfloodlight.org/display/floodlightcontroller/Installation+Guide>
- <http://networkstatic.net/tutorial-to-build-a-floodlight-sdn-openflow-controller-module/>
- Floodlight source code