Homework 4 for CSE 216: Reverse engineering: from code to sequence diagrams (10 points)

Due date: **Feb 22nd (Monday) 10am in class**.

Your task in this homework is to create a sequence diagram by reverse engineering the set of classes supplied below.

- The "found" message that starts the sequence is a doGet(req,res) message sent to the ControllerServlet class.
- Ignore any code in the catch blocks.
- Use the notation presented in Larman Ch 15.
- Include parameters and return values in the messages.
- Use the return value = message name(parameters) message form to show return values; don't draw a separate line.
- In the lifeline boxes, always use the class name and use named instances where appropriate.
- Getting the sequence of messages correct is much more important than notational details like filling in the arrow heads on the message arrows.

```
1 /*
2 * ControllerServlet.java
3 */
4
5 public class ControllerServlet extends HttpServlet {
6   private ActionFactory factory = new ActionFactory();
7
8   ...
```

```
 9
10    protected void processRequest
11      (HttpServletRequest req, HttpServletResponse res)
12      throws ServletException, IOException {
13        try {
14          String actionClassName = getActionClass(req);
15          Action action =
16            factory.getAction
17              (actionClassName, getClass().getClassLoader());
18          ActionRouter router = action.perform(this, req, res);
19          router.route(this, req, res);
20        } catch (Exception ex) {
21          throw new ServletException(ex);
22        }
23      }
24
25    ...
26
27    protected void doGet
28      (HttpServletRequest request, HttpServletResponse response)
29      throws ServletException, IOException {
30        processRequest(request, response);
31      }
32
33    ...
34
35  }
36
37
38  /*
39  * ActionFactory.java
40  *
41  */
42
43  public class ActionFactory {
44    public ActionFactory() { }
45
46    public Action getAction(String className, ClassLoader loader)
47      throws ClassNotFoundException, IllegalAccessException,
48      InstantiationException {
```

```
49        Class klass = loader.loadClass(className);
50        action = (Action) klass.newInstance();
51        return action;
52      }
53    }
54
55  /*
56  * Action.java
57  *
58  */
59
60  public interface Action {
61    public ActionRouter perform(HttpServlet servlet,
62                                HttpServletRequest req,
63                                HttpServletResponse res)
64      throws IOException, ServletException;
65  }
66
67  /*
68  * ActionRouter.java
69  *
70  */
71
72  public class ActionRouter {
73    private String key;
74    private final boolean isForward;
75
76    public ActionRouter(String key) {
77      this(key, true);
78    }
79
80    public ActionRouter(String key, boolean isForward) {
81      this.key = key;
82      this.isForward = isForward;
83    }
84
85    public void route(HttpServlet servlet,
86                      HttpServletRequest req,
87                      HttpServletResponse res)
88      throws ServletException, IOException {
```

```
89          ...
90      }
91
92  }
```

Submission

Please use the free tool bouml (http://bouml.free.fr/) to draw your sequence diagram; print your diagram on paper and submit it at the beginning of the class on Feb 22nd.

Instructions for creating sequence diagrams using bouml: create a project; right click the project to create a new Use Case View or Class View; right click the view to create a new sequence diagram.