```java
public class ClassActivity1 {

    /**
     * Prints Student Name, Group ID
     * @param args[0] is Student first and last name
     * Example: java ClassActivity1 "Ben Holland"
     */
    public static void main(String[] args) {
        String studentName = args[0];
        char[] characters = studentName
                .toUpperCase()
                .replaceAll("\\s+","")
                .toCharArray();
        int groupID = 0;
        if(characters.length >= 3) {
            int maxGroups = 5;
            // Note that 'A' == 65, 'A' = 0x41
            int asciiSumFirst3Chars = (int) characters[0]
                                    + (int) characters[1]
                                    + (int) characters[2];
            groupID = asciiSumFirst3Chars % maxGroups;
        }
        System.out.println("Student: " + studentName
                + ", Group: " + (groupID+1));
    }

}
```

Announcements

- Open Help Hours
  - Wednesdays/Fridays 1:10-2:00pm in Gilman 1810
  - One addition help hour TBD
- Using Piazza
  - Use it! Use your peers. Help your peers.
  - Generally will not be answering requests for help in email
- Reminder of course theme
  - Emphasis is on critical thinking!
  - Tutorials to complete assignments do not exist
  - Experiment, learn, and document your thinking (including failures)!
  - Assignments must be *professionally* typed!

Group Activity 1

- Assignment 1
  - https://github.com/SE421/assignment1/blob/master/assignment1.pdf
  - Problem 1: 15 minutes
  - Problem 2: 15 minutes
  - Problem 3: 20 minutes

- Group Participation
  - Nominate group representative
  - Complete attendance sheet online (one per group)
    - https://goo.gl/forms/m7WLXnH49denNojC3
  - At the end of the activity group representative should summarize group thinking
  - *System.out.print("Presenting Group: " + new Random().nextInt(maxGroups) + 1);*

Exercise (2014): Refactoring CVE-2012-4681

- “Allows remote attackers to execute arbitrary code via a crafted applet that bypasses SecurityManager restrictions…”
- CVE Created August 27th 2012 (~2 years old…)

Sample	**Notes**	**Score (2014’s positive detections)**
Original Sample	http://pastie.org/4594319	30/55
Technique A	Changed Class/Method names	28/55
Techniques A and B	Obfuscate strings	16/55
Techniques A-C	Change Control Flow	16/55
Techniques A-D	Reflective invocations (on sensitive APIs)	3/55
Techniques A-E	Simple XOR Packer	0/55

Exercise (2014): Refactoring CVE-2012-4681

Three main approaches that were demonstrated in class

1) Refactoring strings that appear in bytecode of compiled classes
2) Use of Java reflection to indirectly invoke functions
   - https://gist.github.com/benjholla/1a219f30397c2608065f
3) Use of Java class loaders to load new runtime class definitions

Problem 1 (15 minutes)

b) What are YARA rules? How can we develop YARA rules to detect known malware?

c) What evasion techniques have you tried / thought of?
   - What were the preliminary results?
   - What resources have you found in your research so far?

Problem 2 (15 minutes)

- Discuss Reflections on Trusting Trust Paper
  - What is the described attack?
  - Why is it interesting?
  - How could we detect it?

Problem 3 (20 minutes)

- How to write a quine program?
- How to write a quine-relay program?