COMP 1451 – Assignment #2 (15 points)
Due: 11:59 p.m. the night before Session 8

As we all know, spam (unsolicited junk email) is clogging up the arteries of the internet. In this project you are going to modify the mail-system project from chapter 3 to filter spam.

Here are the specifications for this project. Read them carefully:

- Do Exercise 3.33 from the textbook to become familiar with the mail-system project. Study its functionality carefully.

- Analyze of the project in terms of responsibility-driven design (chapter 7). What is each class responsible for? Are the classes loosely or tightly coupled? Are they cohesive? Are the methods cohesive?

- Modify the MailItem class so that it includes a subject field. The constructor will now expect from, to, subject, message in that order. Provide an appropriate accessor method for the subject, and modify the print method to include it.

- Modify the MailServer so that it uses a HashMap to store MailItems instead of an ArrayList. The keys to the HashMap must be the names of the recipients, and each value must be an ArrayList containing all the MailItems stored for that recipient. (Think of this as that person's mailbox.) The names of the recipients must be case-insensitive, i.e. “darby” and “Darby” and “DARBY” are all the same person. If a recipient does not already have a mailbox a new one must be created for them. All names (both senders and recipients) must be properly formatted, i.e. first letter uppercase and the rest lowercase. (Can you think of a way to do this without duplicating a lot of code? A static method in one of the classes perhaps?)

- Create a separate class called SpamFilter. Any email whose subject starts with the word "spam" (any case) or whose subject contains the phrases “Online Pharmacy”, "Cheap Viagra", “Generic Viagra”, “Fake Watches”, “Replica Watches” (capitalized or not) is spam, and must not be posted by the MailServer. Note that a subject such as “I hate spam” is ok according to our criteria.

- Add a method to the MailServer class called printMessagesSortedByRecipient. This method sorts the recipients (the users who have mailboxes) and prints each recipient's name, and then prints out all the emails sent to that person without removing them from the server. Be sure this method calls the appropriate method from the MailItem class. If there is a user with an empty mailbox print a message to that effect, e.g. “No mail.”

- Do not change the signatures of any of the existing MailServer methods. Do not make any changes to MailClient method signatures except to add a subject to

the mail being sent. You may add one or more methods, keeping in mind responsibility-driven design.

- For the MailServer class and the SpamChecker, create test classes with unit tests to thoroughly test the modified project. Test a variety of scenarios, including the case where you request mail for a user who has not yet received any mail. Document (Javadoc comments) your test cases to explain what you are testing and whether it is a positive or a negative test.

- Think carefully about your modified code. Do not use a loop where it is no longer required. Do not duplicate functionality. Marks will be given for:

- Style – see the style guide Appendix J of the textbook.

- Analysis – project is analyzed in terms of responsibility-driven design.

- Correctness and completeness – system meets functional requirements.

- Unit tests – system is tested thoroughly and the test cases are documented with comments.

Name the project with your ID#. Create a .zip file containing your entire project. Name the .zip file with your name and the assignment number, e.g. “A00..._Assign2.zip”. Upload the file to the D2L assignment dropbox before the cutoff time.