CS421 - Computer Networks
Programming Assignment 1
VendingMachine

Due: 13/3/2016

In this programming assignment, you are asked to implement a server that simulates a vending machine and a client that requests service from the server in Java. Your program must be a console application (no graphical user interface (GUI) is required) and should be named as *VendingMachine* (i.e., the name of the class that includes the main method should be *VendingMachine*). Your program should run with the

```
java VendingMachine [<IP_address>] <port_number>
```

command, where `<IP_address>` is the IP address of the host to which your program should connect and `<port_number>` is the port number to which the socket is bound. The first command line argument, i.e., `<IP_address>`, is **optional**. If it is not given, the executed program behaves as the server. It opens a server socket at port `<port_number>` and listens for a connection. If it is given, the executed program behaves as the client. It connects to the host with IP address `<IP_address>` and port `<port_number>`.

The client The client simulates a customer getting service from a vending machine. There are two types of messages that the client may send to the server.

- **GET ITEM LIST**
  The client sends

  ```
  GET ITEM LIST\r\n
  \r\n
  ```

  message to the server to request the details of the items in the vending machine.

- **GET ITEM**
  The client sends

  ```
  GET ITEM\r\n
  <item_id> <item_count>\r\n
  \r\n
  ```

  message to the server to get `<item_count>` number of items with id `<item_id>`. The fields `<item_id>` and `<item_count>` should be separated by one space character.

The server The server simulates a vending machine. When the server starts, it reads the details of the items from the file `item_list.txt` residing in the folder including the source file(s). Each line of this file includes the **unique** integer id and the name of an item, and the number of items with that id in the virtual vending machine. After the file is read and the details of the items in the machine are obtained, the server listens to the server socket opened at port `<port_number>`. When a client is connected to the server, the server waits for the client to send a message and responds to the client with an appropriate message. There are three types of messages the server may send to the client.

- **ITEM LIST**
  If the server receives a `GET ITEM LIST` message, it responds to the client with an `ITEM LIST` message including the details of the items in the machine. Assuming that there are M types of items in the machine and the machine includes `<item_count_m>` items with item id `<item_id_m>` and item name `<item_name_m>` for $m = 1, \ldots, M$, then the following message is sent to the client:

  ```
  ITEM LIST\r\n
  <item_id_1> <item_name_1> <item_count_1>\r\n
  ...
  <item_id_M> <item_name_M> <item_count_M>\r\n
  \r\n
  ```

  The fields `<item_id_m>`, `<item_name_m>`, and `<item_count_m>` should be separated by one space character for $m = 1, \ldots, M$.

- **SUCCESS**
  If the server receives a `GET ITEM` message and the machine includes the requested items, the following message is sent to the client:

  ```
  SUCCESS\r\n
  \r\n
  ```

- **OUT OF STOCK**
  If the server receives a `GET ITEM` message and the machine does not include the requested number of items of that type, the following message is sent to the client:

  ```
  OUT OF STOCK\r\n
  \r\n
  ```

Other issues

- The file including the initial item list should reside in the folder including the source file(s) and be named as `item_list.txt`. This file is read once when the server starts. The fields of the file need to be separated by space or tab characters. You may assume that each line of the file includes a different type of item and the number of items is positive.
- The name of an item should not include a space character.
- The message to be sent by the client is determined by the user input interactively, that is, the client should prompt the user to get input. The format of user input is not specified; so, you may choose an appropriate format.
- The server should send an `OUT OF STOCK` message if the client requests an item with id that is not found in the `item_list.txt` file.
- The server should print received messages and sent messages to the command line.
- The client should prompt the user to terminate the connection. When the user wants to terminate the connection, the client should summarize the details of the items successfully received from the machine and terminate the connection.

- The connection between the server and a client is terminated when the client program ends. The server should handle an exception thrown when the client program is terminated.
- After the connection between a client and a server is terminated, the server should wait for another client's connection.
- You may test your program by executing the application more than once on the same computer.
- The messages sent by the client and the server should be in the format described above. You will lose points if the messages in another format are sent.

Example

Assume that the content of the `item_list.txt` be as follows

```
101   cola           15
102   diet-cola       8
103   chocolate      12
104   potato-chip     4
```

and let

```
java VendingMachine 8080
```

and

```
java VendingMachine 111.111.111.111 8080
```

be two respectively executed commands, where the former runs the server program listening to port 8080 on a computer with IP address 111.111.111.111 and the latter runs the client program trying to connect to the server.

Now, let the client request the list of the items in the machine, respectively gets 4 chocolate items, 5 potato-chip items, 3 cola items, 20 chocolate items, and 3 chocolate items, and terminates the connection. Then the following messages are sent:

1. client → server

```
GET ITEM LIST\r\n
\r\n
```

2. server → client

```
ITEM LIST\r\n
101 cola 15\r\n
102 diet-cola 8\r\n
103 chocolate 12\r\n
104 potato-chip 4\r\n
\r\n
```

3. client → server

```
GET ITEM\r\n
103 4\r\n
\r\n
```

4. server → client

```
SUCCESS\r\n
\r\n
```

5. client → server

```
GET ITEM\r\n
104 5\r\n
\r\n
```

6. server → client

```
OUT OF STOCK\r\n
\r\n
```

7. client → server

```
GET ITEM\r\n
101 3\r\n
\r\n
```

8. server → client

```
SUCCESS\r\n
\r\n
```

9. client → server

```
GET ITEM\r\n
103 20\r\n
\r\n
```

10. server → client

```
OUT OF STOCK\r\n
\r\n
```

11. client → server

```
GET ITEM\r\n
103 3\r\n
\r\n
```

12. server → client

```
SUCCESS\r\n
\r\n
```

After the connection is terminated, the command-lines of the client and the server may be as follows:

Command-line of the client:

```
The connection is established.
Choose a message type (GET ITEM (L)IST, (G)ET ITEM, (Q)UIT): L
The received message:
  ITEM LIST
  101 cola 15
  102 diet-cola 8
  103 chocolate 12
  104 potato-chip 4
Choose a message type (GET ITEM (L)IST, (G)ET ITEM, (Q)UIT): G
Give the item id: 103
Give the number of items: 4
The received message:
  SUCCESS
Choose a message type (GET ITEM (L)IST, (G)ET ITEM, (Q)UIT): G
Give the item id: 104
Give the number of items: 5
The received message:
  OUT OF STOCK
Choose a message type (GET ITEM (L)IST, (G)ET ITEM, (Q)UIT): G
Give the item id: 101
Give the number of items: 3
The received message:
  SUCCESS
Choose a message type (GET ITEM (L)IST, (G)ET ITEM, (Q)UIT): G
Give the item id: 103
Give the number of items: 20
The received message:
  SUCCESS
Choose a message type (GET ITEM (L)IST, (G)ET ITEM, (Q)UIT): G
Give the item id: 103
Give the number of items: 3
The received message:
  SUCCESS
Choose a message type (GET ITEM (L)IST, (G)ET ITEM, (Q)UIT): Q
The summary of received items:
  103 7
  101 3
```

Command-line of the server:
item_list.txt is read
The current list of items:
  101 cola 15
  102 diet-cola 8
  103 chocolate 12
  104 potato-chip 4

Waiting for a client... A client is connected.
The received message:
  GET ITEM LIST
Send the message:
  ITEM LIST
  101 cola 15
  102 diet-cola 8
  103 chocolate 12
  104 potato-chip 4
The received message:
  GET ITEM
  103 4
Send the message:
  SUCCESS
The received message:
  GET ITEM
  104 5
Send the message:
  OUT OF STOCK
The received message:
  GET ITEM
  101 3
Send the message:
  SUCCESS
The received message:
  GET ITEM
  103 20
Send the message:
  OUT OF STOCK
The received message:
  GET ITEM
  103 3
Send the message:
  SUCCESS
The client has terminated the connection.
The current list of items:
  101 cola 12
  102 diet-cola 8
  103 chocolate 5
  104 potato-chip 4

Waiting for a client...

Submission rules

You need to apply all of the following rules in your submission. **You will lose points if you do not obey the submission rules below or your program does not run as described in the assignment above**.

- The assignment should be submitted as an e-mail attachment sent to **morhan[at]bilkent.edu.tr**. Any other methods (Disk/CD/DVD) of submission will not be accepted.
- The subject of the e-mail should start with *[CS421_PA1]*, and include your name and student ID. For example, the subject line must be

  [CS421_PA1]AliVelioglu20111222

  if your name and ID are *Ali Velioglu* and *20111222*. If you are submitting an assignment done by two students, the subject line should include the names and IDs of both group members. The subject of the e-mail should be

  [CS421_PA1]AliVelioglu20111222AyseFatmaoglu20255666

  if group members are *Ali Velioglu* and *Ayse Fatmaoglu* with IDs *20111222* and *20255666*, respectively.
- All the files must be submitted in a zip file whose name is the same as the subject line **except** the [CS421_PA1] part. The file must be a **.zip** file, not a **.rar** file or any other compressed file.
- All of the files must be in the root of the zip file; directory structures are **not** allowed. Please note that this also disallows organizing your code into Java packages. The archive should **not** contain:
  - Any class files or other executables,
  - Any third party library archives (i.e. jar files),
  - Any text files (including the `item_list.txt` file you have used in testing your program),
  - Project files used by IDEs (e.g., JCreator, JBuilder, SunOne, Eclipse, Idea or NetBeans etc.). You may, and are encouraged to, use these programs while developing, but the end result must be a clean, IDE-independent program.

- The standard rules for plagiarism and academic honesty apply, if in doubt refer to 'Student Disciplinary Rules and Regulation', items 7.j, 8.l and 8.m.