COMP125: Assignment 1 - IP Addresses

Due: **23:45, Sunday 11th September**

Worth 5%

Updates

1. (21st August) Corrected the bit just before the ObjectAid class diagram. There was a reference to "Semester marks" class from an old assignment.
2. (22nd August) Updated `IPAddressClient` comments to remove some inconsistencies.
3. (22nd August) Updated setter javadoc to explicitly remind students to call the reset method before doing anything else, otherwise you'll get `NullPointerException` because of operating on a `null` array.

All queries via the Assignment Discussion Forum on iLearn or email gaurav.gupta@mq.edu.au.

Learning Outcomes

The first assignment assesses students' skills on the following topics,

- Reading Javadoc
- Writing Java class definitions
- Programming to tests
- Debugging
- Creating objects and using them
- Problem solving and developing algorithms

Pre-requisite

In order for your assignment to be eligible for being marked, you must have passed the `Classes and Objects` quizzes (and all quizzes before that one) on IQ System. The quizzes have dependencies, so you will need to pass all quizzes before Classes and Objects for it to become available. The quizzes are:

1. operations
2. variables
3. expressions
4. conditions
5. loops
6. functions - 1
7. arrays
8. classes and objects

Your assignment will NOT be marked if you haven't passed the Classes and Objects quiz (and by definition, all quizzes before it).

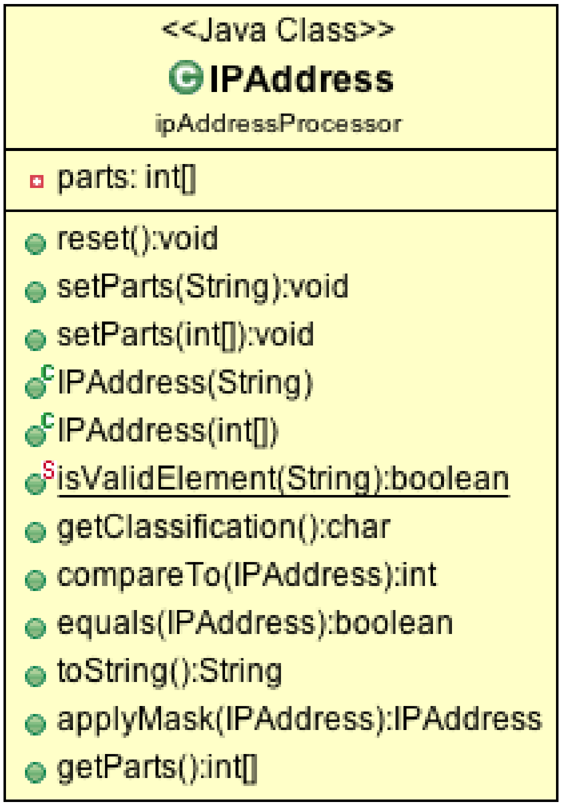
Template

The starting template has been provided to you in `ipAddressProcessorStartingPoint`. Import it into Eclipse by choosing File -> Import -> General -> Existing projects into workspace -> Select archive file -> Browse for and choose `ipAddressProcessorStartingPoint.zip` -> Finish. You **DO NOT** need to modify any file except `IPAddress.java` and `IPAddressClient.java`.

Explanation of tasks

You are required to define a class for an IP Address (specifically an IPv4 address).

The class diagram, drawn using ObjectAid, is given below



<<Java Class>>
IPAddress
ipAddressProcessor

parts: int[]

reset():void
setParts(String):void
setParts(int[]):void
IPAddress(String)
IPAddress(int[])
isValidElement(String):boolean
getClassification():char
compareTo(IPAddress):int
equals(IPAddress):boolean
toString():String
applyMask(IPAddress):IPAddress
getParts():int[]

Description of tasks and marks for each task

The description for each method is provided as a javadoc comment above the method. Once you import the project into Eclipse, go to `ipAddressProcessor/doc/index.html` to access the javadoc in easy navigation form. Pay attention to the javadoc method comment to clearly understand the requirement from the method.

The order in which we suggest you complete the methods is below. Marks for passing the test for each method is specified next to the method.

1. `reset()`: 5 marks
2. `isValidElement(String)`: 15 marks
3. `setParts(String)`: 15 marks
4. `setParts(int[])`: 15 marks
5. `getParts()`: 5 marks
6. `compareTo(IPAddress)`: 5 marks
7. `equals(IPAddress)`: 5 marks
8. `getClassification()`: 5 marks
9. `sameSubnet(IPAddress, IPAddress)`: 5 marks

The two constructors, `IPAddress(String)` and `IPAddress(int[])` have been completed and should not be modified. Neither should the method `toString()` be changed.

Once again, specifications for each method is provided as javadoc.

There are 15 marks for completion of `IPAddressClient.java` as per the description in the class. Sample output, to which the client comments refer to, are as follows,

```
153.12.60.2 masked by 255.255.0.0 = 153.12.0.0
153.12.224.72 masked by 255.255.0.0 = 153.12.0.0
153.12.60.2 masked by 255.255.192.0 = 153.12.0.0
153.12.224.72 masked by 255.255.192.0 = 153.12.192.0
153.12.60.2 and 153.12.224.72 in the same subnet for mask 255.255.0.0
153.12.60.2 and 153.12.224.72 in different subnet for mask 255.255.192.0
153.12.60.2 is lesser than 153.12.224.72
153.12.224.72 is greater than 153.12.60.2
153.12.60.2 and 153.12.60.2 are equal
```

There are 10 marks for code style, including commenting, indentation, and variable naming.

Late submissions

Late submissions will not be accepted unless extension is granted upon disruption to study application via ask.mq.edu.au system.

Submission guidelines

Follow these steps for making a correct submission,

- Click on `IPAddress.java submission` link on iLearn and drag and drop `IPAddress.java` into the submission box. **Ensure you submit `IPAddress.java` and not `IPAddress.class` .**

- Click on IPAddressClient.java submission link on iLearn and drag and drop IPAddressClient.java into the submission box. **Ensure you submit IPAddressClient.java and not IPAddressClient.class 20 out of 100 marks will be deducted if the assignment is submitted in any way other than the one suggested above**.

Procedure to deal with plagiarism

Any code that is flagged for plagiarism would be reported to the faculty disciplinary committee and once this is done, the convenor will have no authority to influence the outcome of the case. I would suggest the students to be extremely careful in not sharing code / pseudocode otherwise you stand to face serious consequences.

Some tips,

- We will remove all import statements from IPAddress.java so using built-in libraries is futile.
- Do not hard-code to the tests. We will use different ip addresses in the final test used for marking the assignment.
- Use internet as your resource tool. If you need to find out on how to split a String on the dot character, Google it! ("how to split a String on the dot character"). You should read the results returned carefully since a simple copy-paste might not suit your purpose.

Help program

A program to help with the assignment was written during week 3 Friday lecture. The code for that week is in package helpProgram along with the starting point.