School of Computing, University of Leeds

COMP2221 Networks

Coursework 1: `InetAddress` and IPv4 address ranges

Deadline: Monday March 2nd, 10am.

If you have any queries about this coursework, visit the Yammer page for this module (`COMP2221 1920`). If your query is not resolved by previous answers, post a new message.

Learning objectives

- Use of the `java.net.InetAddress` class.
- To understand the format of IPv4 addresses and IPv4 address ranges.

This item of coursework is worth **15%** of the final module grade.

Task

Write a Java class called `Coursework1` that:

- Reads in exactly two arguments from the command line. The first argument is the `filter`, and the second argument is the `hostname`.
- Checks that the `filter` is a valid IPv4 address, or range of addresses with wildcards (‘`*`’).
- Converts the `hostname` to an instance of `InetAddress`, and checks it is an IPv4 address.
- Returns `true` if the hostname either matches the filter, or belongs to the range of addresses specified by the filter.
- Returns `false` otherwise.
- Returns meaningful error messages for invalid input, specifying *why* the input was invalid.

If both filter and hostname are valid, your solution should only print `true` or `false`. Examples:

```
$ java Coursework1 129.11.144.10 www.comp.leeds.ac.uk
true
```

```
$ java Coursework1 129.11.144.* www.comp.leeds.ac.uk
true
```

```
$ java Coursework1 129.11.144.* www.leeds.ac.uk
false
```

Your code should adhere to the coding standards in `JavaCodingStandards.pdf` on Minerva. Your solution should expect no interaction with the user other than *via* the command line arguments when launched.

Guidance

IPv4 addressing and the use of wildcards (‘`*`’) was covered in Lecture 5. Valid IPv4 filters are *e.g.* `129.11.*.*` or `129.*.*.*`. Note that `129.11.*.2` is **not** valid as it has a value after a ‘`*`’. Your code should identify such cases and return a meaningful error message.

Some methods of `InetAddress` were covered in Lecture 5; the full specification can be easily found online[1]. You can also look at the book *Java Network Programming* by Harold mentioned in Lecture 1.

You may like to base your solution on the `nslookup` Java code that was covered at the start of Lecture 6. You are not expected to use streams as covered in the remainder of Lecture 6, or any of the material covered in Lectures 7 or later, for this item of coursework.

To check your code is working, you may like to know what the actual IP address is for a given hostname. To see this, you can use one of the UNIX command line tools that were mentioned at the end of Lecture 4 (`host`, `dig` *etc.*).

You may find it helpful to use the `split` method for Java strings, to separate the filter string into substrings separated by a ‘`.`’ character: `String components[] = filterString.split("[.]");` (the square brackets in the quotes are necessary as Java interprets this as a regular expression).

IP addresses use unsigned bytes, but Java supports signed integers. If you find you need to convert from an unsigned byte to a signed integer, you can use the following Java expression:

```
int value = ((int)byteValue) & 0xff;
```

If you want to test your solution correctly returns an error message for IPv6 hostnames, you can use `dns6.leeds.ac.uk`.

Marks

6 marks	:	Error messages for incorrect input, giving the reason why the input was invalid.
3 marks	:	Correctly matches filters without wildcards.
3 marks	:	Correctly matches filters with wildcards.
3 marks	:	Structure and commenting. Adherence to the provided standard.
Total: 15		

Submission

If you submit a single file, it should be called `Coursework1.java` and uploaded to Minerva.

If your solution is split between multiple files, they should be all placed in the same directory. To submit, `cd` to that directory and delete all extraneous files (`.class`, any IDE-related files *etc.*), then create an archive `.tar.gz` file:

```
tar cvzf cwk1.tar.gz *
```

This will create the file `cwk1.tar.gz` that you should upload to Minerva.

The following sequence of steps will be performed to assess your submission:

1. Unarchive the submission if it ended `.tar.gz`.

[1] *e.g.* `https://docs.oracle.com/javase/7/docs/api/java/net/InetAddress.html`

2. Compile all files with `javac *.java`
3. Execute your solution, *e.g.* `java Coursework1 129.*.*.* comp.leeds.ac.uk`

Your submission <u>must</u> work when this sequence is followed.

All testing will be performed on a School Linux machine, similar to those in DEC-10.

Disclaimer

This is intended as an individual piece of work and, while discussion of the work is encouraged, what you submit should be entirely your own work. Code similarity tools will be used to check for collusion, and online source code sites will be checked.

The standard late penalty of 5% per day applies for work submitted after the deadline.

Ensure you retain the receipt for your submission as you may be asked to produce it at a later date in the event of a disputed submission time.