TUTORIAL 4: Using API documentation SOLUTION

Look at the following excerpt from the API documentation for the class `InetAddress`, which is part of the `java.net` package. This class represents an Internet Protocol (IP) network address.

Use the documentation to complete the task described over the page.

Method Summary	
boolean	**equals**(Object obj) Compares this object against the specified object.
byte[]	**getAddress**() Returns the raw IP address of this InetAddress object.
static InetAddress[]	**getAllByName**(String host) Given the name of a host, returns an array of its IP addresses, based on the configured name service on the system.
static InetAddress	**getByAddress**(byte[] addr) Returns an InetAddress object given the raw IP address .
static InetAddress	**getByAddress**(String host, byte[] addr) Create an InetAddress based on the provided host name and IP address No name service is checked for the validity of the address.
static InetAddress	**getByName**(String host) Determines the IP address of a host, given the host's name.
String	**getCanonicalHostName**() Gets the fully qualified domain name for this IP address.
String	**getHostAddress**() Returns the IP address string in textual presentation.
String	**getHostName**() Gets the host name for this IP address.
static InetAddress	**getLocalHost**() Returns the local host.
boolean	**isMulticastAddress**() Utility routine to check if the InetAddress is an IP multicast address.
boolean	**isReachable**(int timeout) Test whether that address is reachable.
boolean	**isReachable**(NetworkInterface netif, int ttl, int timeout) Test whether that address is reachable.
boolean	**isSiteLocalAddress**() Utility routine to check if the InetAddress is a site local address.
String	**toString**() Converts this IP address to a String.

Task

Use the documentation for `InetAddress` to help you to do each of the following:

1. Write a line of code to import the `InetAddress` class so that you can use it in your program.

   import java.net.InetAddress;

2. Given the line of code:

   ```
   String input = "www.gcal.ac.uk";
   ```

   Write code which:

   - declares a variable of type `InetAddress` called `ip`
   - calls a method of the class `InetAddress` to assign to this variable an `InetAddresss` object representing the IP address of a network host with the name specified by `input`

   InetAddress ip = InetAddress.getByName(input);

   (see note at end for more information about static methods)

3. Write a line of code which prints out the IP address as text

   System.out.println(ip.getHostAddress());

4. Write a line of code which prints out the host name for the IP address

   System.out.println(ip.getHostName());

5. Write a code segment which checks whether the host is reachable within a timeout period of 2 seconds.

The code should print *"reachable"* if the host is reachable, or else print "*not reachable*" if the host is not reachable.

Does the documentation shown give you enough information to be sure how to do this? What would you do to find further information to help you?

```
if(ip.isReachable(2000))
{
      System.out.println("reachable\n");
}
else
{
      System.out.println("not reachable\n");
}
```

Need to know what units the **timeout** should be specified in. Need to click on name of method in Method Summary to see method detail section of documentation. This will show that timeout is in milliseconds

Code for network test tool:

```
// TASK 1
import java.net.InetAddress;

import java.util.Scanner;

public class InetAddressTest
{
   private Scanner reader;

   public InetAddressTest()
   {
      reader = new Scanner(System.in);
   }

   public void doTest()
   {
      System.out.print("Java network tool\nEnter host:\n>");
      String input = reader.nextLine();
```

```
        while(!input.equals("stop"))
        {
            try
            {
                System.out.println("Looking up DNS...");

                // TASK 2
                InetAddress ip = InetAddress.getByName(input);

                // TASK 3 & 4
                System.out.format("Host: %s, IP address:
                        %s\n",ip.getHostName(), ip.getHostAddress());

                System.out.println("Doing PING...");

                // TASK 5
                if(ip.isReachable(2000))
                {
                    System.out.println("reachable\n");
                }
                else
                {
                    System.out.println("not reachable\n");
                }
            }
            catch(Exception e)
            {
                System.out.println("an error ocurred\n");
            }
            System.out.print("Enter host:\n>");
            input = reader.nextLine();
        }
        System.out.println("Finished!\n");
    }

    public static void main(String[] args)
    {
        InetAddressTest tester = new InetAddressTest();
        tester.doTest();
    }
}
```

Note – static methods

getByName is a static method. This means that it can be called using the class name, rather than using the name of an instance as we usually do when calling a method. The call to the static method is:

```
InetAddress.getByName()
```

This is similar to way you use the Math class, which provides static constants and methods which are useful for mathematical expressions. These can be used conveniently without having to create a Math object.For example:

```
Math.PI         (using a static constant)
Math.sin(x)     (calling a static method)
```

The static methods in InetAddress are an example of a special use for static methods. They are factory methods, which create instances of the class they belong to. Factory methods are an alternative to constructors for providing a way of creating and initialising objects.

Factory methods have some advantages over constructors:

- The methods have meaningful names, e.g. getByName, getByAddress, which indicate how the new object is to be instantiated – you can have more than one constructor in a class, but they all have the same name, i.e. the name of the class

- Factory methods can create more than one object at a time, e.g getAllByName, which creates an array of InetAddress objects.

- Factory methods may create objects of different types at runtime, using polymorphism. For example, when you try the code given in this tutorial, the actual object created is probably an Inet4Address object, which is a subclass of InetAddress which specifically represents an IPv4 address. There is also an Inet6Address subclass – if your computer connected to the host using IPv6 then you would get an instance of Inet6Address instead.