Fizz Buzz Problem

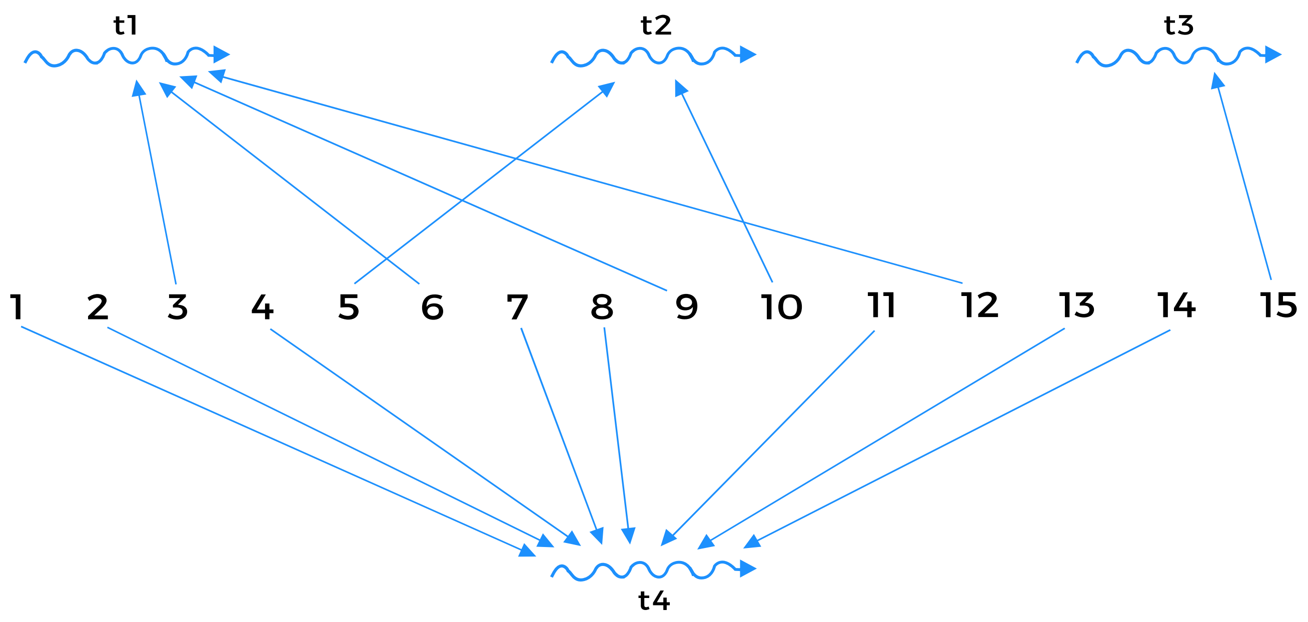
This problem explores a multi-threaded solution to the very common Fizz Buzz programming task

We'll cover the following

Problem Statement#

FizzBuzz is a common interview problem in which a program prints a number series from 1 to *n* such that for every number that is a multiple of 3 it prints "fizz", for every number that is a multiple of 5 it prints "buzz" and for every number that is a multiple of both 3 and 5 it prints "fizzbuzz". We will be creating a multi-threaded solution for this problem.

Suppose we have four threads t1, t2, t3 and t4. Thread t1 checks if the number is divisible by 3 and prints **fizz**. Thread t2 checks if the number is divisible by 5 and prints **buzz**. Thread t3 checks if the number is divisible by both 3 and 5 and prints **fizzbuzz**. Thread t4 prints numbers that are not divisible by 3 or 5. The workflow of the program is shown below:



The code for the class is as follows:

```
class MultithreadedFizzBuzz {

    private int n;

    public MultithreadedFizzBuzz(int n) {
        this.n = n;
    }

    public void fizz() {
        System.out.print("fizz");
    }

    public void buzz() {
        System.out.print("buzz");
    }

    public void fizzbuzz() {
        System.out.print("fizzbuzz");
    }

    public void number(int num) {
        System.out.print(num);
    }
}
```

For an input integer *n*, the program should output a string containing the words fizz, buzz and fizzbuzz representing certain numbers. For example, for n = 15, the output should be: 1, 2, fizz, 4, buzz, fizz, 7, 8, fizz, buzz, 11, fizz, 13, 14, fizzbuzz.

Solution#

We will solve this problem using the basic Java functions; `wait()` and `notifyAll()`. The basic structure of the class is given below.

```
class MultithreadedFizzBuzz {

    private int n;
    private int num = 1;

    public MultithreadedFizzBuzz(int n) {
    }
    public void fizzbuzz() {
    }
    public void fizz() {
    }
    public void buzz() {
    }
    public void number() {
    }
}
```

The `MultithreadedFizzBuzz` class contains 2 private members: `n` and `num`.

`n` is the last number of the series to be printed whereas `num` represents the current number to be printed. It is initialized with 1.

The constructor `MultithreadedFizzBuzz()` initializes `n` with the input taken from the user.

```
public MultithreadedFizzBuzz(int n) {
        this.n = n;
}
```

The second function in the class, `fizz()` prints "fizz" only if the current number is divisible by 3. The first loop checks if `num` (current number) is smaller than or equal to `n` (user input). Then `num` is checked for its divisibility by 3. We check if `num` is divisible by 3 and not by 5 because some multiples of 3 are also multiples of 5. If the condition is met, then "fizz" is printed and `num` is incremented. The waiting threads are notified via `notifyAll()`. If the condition is not met, the thread goes into `wait()`.

```
public synchronized void fizz() throws InterruptedException {
    while (num <= n) {
        if (num % 3 == 0 && num % 5 != 0) {
            System.out.println("fizz");
            num++;
            notifyAll();
        }
        else {
            wait();
        }
    }
}
```

The next function `buzz()` works in the same manner as `fizz()`. The only difference here is the check to see if `num` is divisibile by 5 and not by 3. The reasoning is the same: some multiples of 5 are also multiples of 3 and those numbers should not be printed by this function. If the condition is met, then "buzz" is printed otherwise the thread will `wait()`.

```
public synchronized void buzz() throws InterruptedException {
    while (num <= n) {
        if (num % 3 != 0 && num % 5 == 0) {
            System.out.println("buzz");
            num++;
            notifyAll();
        }
        else {
            wait();
        }
    }
}
```

The next function `fizzbuzz()` prints "fizzbuzz" if the current number in the series is divisible by both 3 and 5. A multiple of 15 is divisible by 3 and 5 both so `num` is checked for its divisibility by 15. After printing "fizzbuzz", `num` is incremented and waiting threads are notified via `notifyAll()`. If `num` is not divisible by 15 then the thread goes into `wait()`.

```
public synchronized void fizzbuzz() throws InterruptedException {
   while (num <= n) {
      if (num % 15 == 0) {
         System.out.println("fizzbuzz");
         num++;
         notifyAll();
      }
      else {
         wait();
      }
   }
}
```

The last function `number()` checks if `num` is neither divisible by 3 nor by 5, then prints the `num`.

```java
public synchronized void number() throws InterruptedException {
    while (num <= n) {
        if (num % 3 != 0 && num % 5 != 0) {
            System.out.println(num);
            num++;
            notifyAll();
        }
        else {
            wait();
        }
    }
}
```

The complete code for `MultithreadedFizzBuzz` is as follows

```
class MultithreadedFizzBuzz {

    private int n;
    private int num = 1;

    public MultithreadedFizzBuzz(int n) {
        this.n = n;
    }

    public synchronized void fizz() throws InterruptedException {
        while (num <= n) {
            if (num % 3 == 0 && num % 5 != 0) {
                System.out.println("fizz");
                num++;
                notifyAll();
            } else {
                wait();
            }
        }
    }

    public synchronized void buzz() throws InterruptedException {
        while (num <= n) {
            if (num % 3 != 0 && num % 5 == 0) {
                System.out.println("buzz");
                num++;
                notifyAll();
            } else {
                wait();
            }
        }
    }

    public synchronized void fizzbuzz() throws InterruptedExceptio
n {
        while (num <= n) {
            if (num % 15 == 0) {
                System.out.println("fizzbuzz");
                num++;
                notifyAll();
            } else {
                wait();
            }
        }
    }

    public synchronized void number() throws InterruptedException
{
        while (num <= n) {
            if (num % 3 != 0 && num % 5 != 0) {
                System.out.println(num);
                num++;
                notifyAll();
            } else {
                wait();
            }
        }
    }
}
```

We will be creating another class `FizzBuzzThread` for multithreading purpose. It takes an object of `MultithreadedFizzBuzz` and calls the relevant method from the string passed to it.

```
class FizzBuzzThread extends Thread {

    MultithreadedFizzBuzz obj;
    String method;

    public FizzBuzzThread(MultithreadedFizzBuzz obj, String method
){
        this.obj = obj;
        this.method = method;
    }

    public void run() {
        if ("Fizz".equals(method)) {
            try {
                obj.fizz();
            }
            catch (Exception e) {
            }
        }
        else if ("Buzz".equals(method)) {
            try {
                obj.buzz();
            }
            catch (Exception e) {
            }
        }
        else if ("FizzBuzz".equals(method)) {
            try {
                obj.fizzbuzz();
            }
            catch (Exception e) {
            }
        }
        else if ("Number".equals(method)) {
            try {
                obj.number();
            }
            catch (Exception e) {
                }
            }

        }
    }
```

To test our solution, we will be making 4 threads: **t1**,**t2**, **t3** and **t4**. Three threads will check for divisibility by 3, 5 and 15 and print **fizz**, **buzz**, and **fizzbuzz** accordingly. Thread **t4** prints numbers that are not divisible by 3 or 5.

```
class MultithreadedFizzBuzz {
    private int n;
    private int num = 1;

    public MultithreadedFizzBuzz(int n) {
        this.n = n;
    }
    public synchronized void fizz() throws InterruptedException {
        while (num <= n) {
            if (num % 3 == 0 && num % 5 != 0) {
                System.out.println("Fizz");
                num++;
                notifyAll();
            } else {
                wait();
            }
        }
    }

    public synchronized void buzz() throws InterruptedException {
        while (num <= n) {
            if (num % 3 != 0 && num % 5 == 0) {
                System.out.println("Buzz");
                num++;
                notifyAll();
            } else {
                wait();
            }
        }
    }

    public synchronized void fizzbuzz() throws InterruptedException {
        while (num <= n) {
            if (num % 15 == 0) {
                System.out.println("FizzBuzz");
                num++;
                notifyAll();
            } else {
                wait();
            }
        }
    }

    public synchronized void number() throws InterruptedException {
        while (num <= n) {
            if (num % 3 != 0 && num % 5 != 0) {
                System.out.println(num);
                num++;
                notifyAll();
            } else {
                wait();
            }
        }

    }
}

class FizzBuzzThread extends Thread {

    MultithreadedFizzBuzz obj;
    String method;

    public FizzBuzzThread(MultithreadedFizzBuzz obj, String method){
        this.obj = obj;
        this.method = method;
    }

    public void run() {
        if ("Fizz".equals(method)) {
            try {
                obj.fizz();
            }
            catch (Exception e) {
            }
        }
        else if ("Buzz".equals(method)) {
            try {
                obj.buzz();
            }
            catch (Exception e) {
            }
        }
        else if ("FizzBuzz".equals(method)) {
            try {
                obj.fizzbuzz();
            }
            catch (Exception e) {
            }
        }
        else if ("Number".equals(method)) {
            try {
                obj.number();
            }
            catch (Exception e) {
            }
        }

    }
}

class main
{
        public static void main(String[] args) {
                MultithreadedFizzBuzz obj = new MultithreadedFizzBuzz(15);
            Thread t1 = new FizzBuzzThread(obj,"Fizz");
            Thread t2 = new FizzBuzzThread(obj,"Buzz");
            Thread t3 = new FizzBuzzThread(obj,"FizzBuzz");
            Thread t4 = new FizzBuzzThread(obj,"Number");

            t2.start();
            t1.start();
            t4.start();
            t3.start();
        }
}
```

Interviewing soon? We've partnered with Hired so that companies apply to y
utm_source=educative&utm_medium=lesson&utm_location=CA&utm_camp