#Part 2 java monitor

##Team Member:
shengbo lou: 28530995

zhongce ji: 28551884

yucheng lu: 28411887

##Compile:
use eclipse or javac

##Run:
Main class

##Description:
This is the second part of the lab2, here we will use monitor to do the same program as the Semaphores will do. In this part we will use java to implement. We are requested to have one producer and multiple consumers. The producer will produce the requests, and consumers will get the requests and execute them.

####Request.java:
This file contains a class called Request, this class has a constructor which takes one argument which is ID, and this will represent the id of the request. We will use this class to create the requests. Then we have length and ID variables:

- **length**: This will be the length of the requests, it's a random variable
- **ID**: This represents the id of the requests.

There is also one method:

- **length()**: This will return the length of the request, in the function, we use the Random to create the random length of the request.

####Consumer.java:
This file is the implementation of consumer, there is a constructor which takes two arguments: one is ID which will represents the ID of the consumer, the other one is producer, which will produce the request for this consumer.

The class consumer will extends Thread. Then we override the run function, in the run function, first we get the ID and assign it to this consumer thread, the we use producer's getRequest fucntion to get the request, then process the request, finally we print out the output of the programs with the real time of the machine.

####Producer.java:
This file is the implementation of the producer. There is a class called Producer which extends Threads, so that the producer can be the thread. In this class, we have some arguments which are private so that other class may not access it. The variables:

```
MAX_REQUEST: int, this will represent the max number of requests in the queue
sleep_time: int , this will represent the time for producer to sleep after
creating a request. This is also final variable so that we can not change it
total_count: int, this will represent the count for requests in total. This is
also final variable so that we can not change it
count: int, this will represent the count for requests in total
max: int, this will represent the max number of requests in total
queRequests: Queue<Request>, this will represent the queue of the request.
```

Then we have a constructor which takes three arguments:

- **Input**: represent max number of requests we will create and we will assign it to MAX_REQUEST.
- **sleep_time**: represent the time the producer will sleep after creating a new request, and we will assign it to sleep_time.
- **max**: this will represent the max number of requests in total so we can assign it to max
  Then we Override the run function and in this function, we have a while loop, if the total_count is less than max
  which means the queue is not full so we put the request in the queue and increase the total_count and count.

And also we have some synchronized method:

- **putRequest**: This method is void. In the function, first we will check whether the queue is full, if it is full,
  we will wait, if not we will create a new request and give the id which is total_count, then we add the request
  in the queue.Then we get new real time of the machine, then print out the output of the program. After that we
  will sleep the producer for the sleep_time. Then we notify other waiting threads.

- **getRequest**: This method will return Request type. In the function first we will create the request, and then notify
  the waiting thread. if the queue is empty then we need to wait. Then after check those things, we will take the
  first request from the request queue, then we decrease the count because one of the request is out. then we get
  the length of the request by call length method of request. After that we get the real time of the machine and print
  out the output of the program.

Then we have a method called consumer which will return the array of the consumer. This method is not synchronized.
This method will take two arguments which are input and producer. The input will indicates how many consumers we will have
and producer is the producer which will produce the request for the consumer. In the method, we will create an array, and
then we use a for loop to create consumers and put each new consumer in the array. Then we will return the consumer array.

####Main.java:
This file have a main class where we will create producer to run the whole program. In the main function, first we

create new producer which will take 10,1000,20 as input, 10 is max requests number in the queue, 1000 means 1 second for
each sleep, 20 means 20 max requests in total. Then we run the producer, and then we use consumer to create 5 consumers.