HOMEWORK 4 - Fall 2018

HOMEWORK 4 - due Tuesday, October 23rd no later than 6:00PM

REMINDERS:

- **Be sure your code follows the coding style for CSE214.**
- **Make sure you read the warnings about academic dishonesty. *Remember, all work you submit for homework or exams MUST be your own work.***
- **Login to your grading account and click "Submit Assignment" to upload and submit your assignment.**
- **You may use any Java API class that you wish.**
- **You may use Scanner, InputStreamReader, or any other class that you wish for keyboard input.**

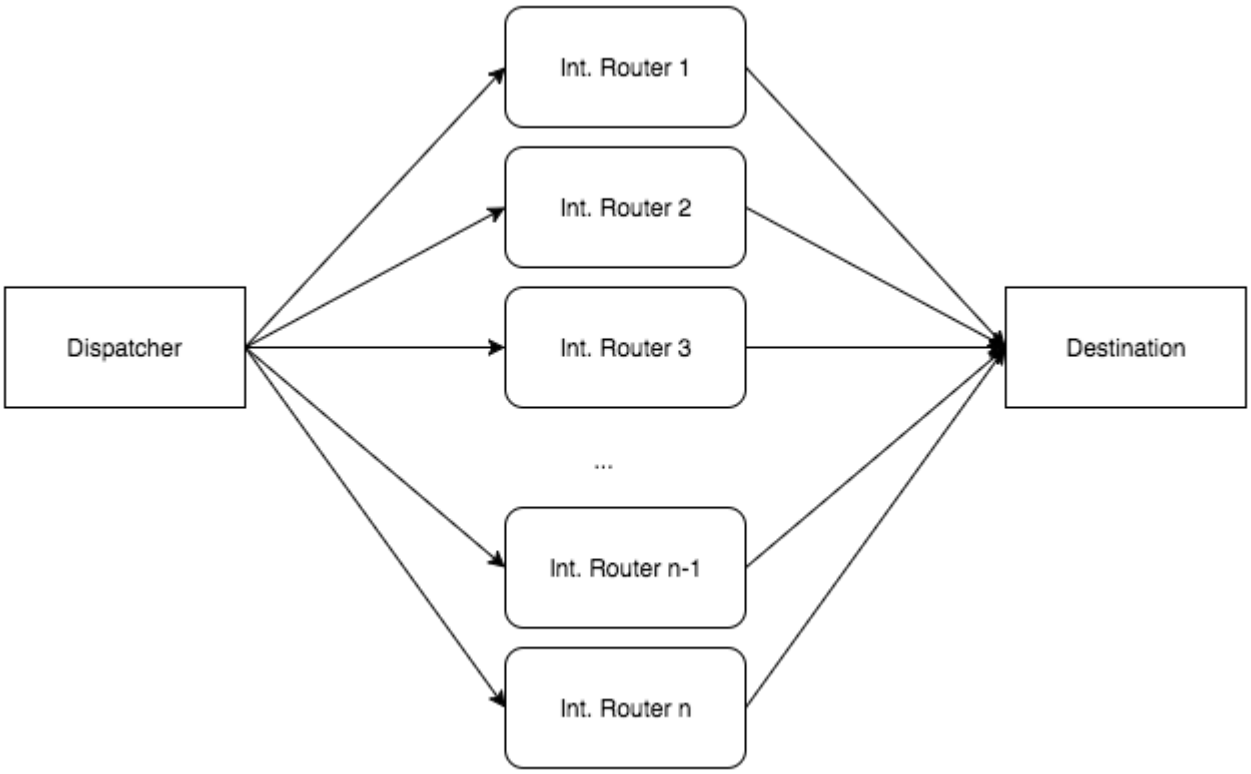
In the field of networking, information is packed into a single entity called a packet. A packet contains the sender's information, and also extra overhead bits that describes certain properties of the packet. Some of these bits are used to determine the path it takes to maneuver through the network. Packets travel through routers across the network, until they have arrived to their specified destination.

In this assignment, you will imitate a simple network of routers by writing a simulator, similar to the CarSimulator discussed in lecture. Note, this is a simple representation of the actual network implemented today.

BRIEF:
The router topology implemented in today's world is far too complicated to model in a small assignment. Therefore, we will simplify this model for our simulation. Assume we have 3 levels of routers:

- **Level 1: Dispatcher Router**
  All packets in our simulation will arrive at this router. The job of this router is to send off the queued packets to one of the available routers in the second level. You will write a simple algorithm that determines which Level 2 routers a packet should jump to. (Discussed later.)
- **Level 2: Intermediate Hop Routers**
  These routers will accept a packet from the dispatcher router. There will be a user-determined amount of these routers. Depending the size of the packet, it takes a variable amount of time to process the packet. Once the arrived packets have been processed, the router can send those packets to their destination.
- **Level 3: Destination Router**
  To simplify our simulation, we assume all packets have the same destination. The job of this router is to accept incoming packets after they have been sent from the second level. However, due to a limited bandwidth (bottleneck in the network), this router can only accept a limited amount of packets at a given moment. This router will not be used in our simulation, but note its purpose.

Here is a picture of our network topology:



BACKGROUND:
Packets first arrive at the Dispatcher router. Each packet can arrive at a probability, prob, that is determined by the user. For each simulation time unit, a maximum of 3 packets can arrive at the Dispatcher. When a Packet arrives at the Dispatcher, a new Packet object should be created, and its data fields will be determined systematically based on rules we will describe later in the Packet class. On the same time unit, the Dispatcher will decide which Intermediate router a specified packet should be forwarded to. You are expected to write a simple algorithm that determines the routing table. (Algorithm specified later in the specs.)
When a Packet arrives at an Intermediate router, it is placed onto its corresponding queue. Only the first packet in the queue will be processed, while the others remain in the queue. However, if the queue is full at the time a packet comes in, we consider that as a buffer overflow, and the network will drop this packet. Once the router finds that a packet is ready to be sent, it will forward it to its final destination.
The final destination receives all incoming packets from the Intermediate routers. However, due to limited bandwidth, the destination router can only accept a limited amount of packets, limit, which is determined by the user. For example, if 3 packets have been finished processing by the Intermediate routers, but the limit is 2, only 2 packets can arrive at the destination in a simulation unit. The third packet must arrive in the next simulation unit.

NOTE:
Your implementation must consider fairness. That is, Intermediate routers must take turns to send packets. For example, if Intermediate routers 1, 2, 4, and 5 can send a packet at a given simulation unit, and the bandwidth is 2, routers 1 and 2 can send their packet. If, in the next simulation unit, Intermediate router 3 can also send a packet, routers 4 and 5 should send, not 3.

To summarize, for each simulation unit, implement the cases in the following order:

- **Decide whether packets have arrived at the Dispatcher. A maximum of 3 can arrive at a given time.**
- **If the Dispatcher contains unsent packets, send them off to one of the Intermediate routers. You will write the method `sendPacketTo(Collection intRouters)` to decide which router the packet should be forwarded to.**
- **Decrement all packets counters in the beginning of the queue at each Intermediate router.**
- **If any packets are ready to be forwarded to the Destination router, do so.**

- **Once a packet has arrived at the Destination router, take note of its arrival by recording the total time in the network. (Be careful of the bandwidth.)**

NOTE: Unlike the CarSimulator discussed in lecture (at most 1 car can arrive at a given time), a maximum of 3 packets can arrive per time unit. Create and use the variable `public static final int MAX_PACKETS = 3`. You can use the following code to simulate random packet arrival:

```
for i in MAX_PACKETS
    if Math.random() < arrivalProb
        generate new Packet
        // each packet has the probability, arrivalProb, to arrive
```

Implement a simulator given the following guidelines. The goal of the simulation is to determine how various parameters can change the efficiency of the network. You will be able to analyze the trends as you experiment with the simulation inputs. You will be graded on how closely you follow the trends. You may add additional variables/methods as you see fit.

1. `public class Packet` - This class represents a packet that will be sent through the network. This object contains the following data fields:

- **`static int packetCount` - this value is used to assign an id to a newly created packet. It will start with the value 0, and every time a new packet object is created, increment this counter and assign the value as the id of the Packet.**
- **`int id` - a unique identifier for the packet. This will be systematically determined by using packetCount.**
- **`int packetSize` - the size of the packet being sent. This value is randomly determined by the simulator by using the Math.random() method.**
- **`int timeArrive` - the time this Packet is created should be recorded in this variable**
- **`int timeToDest` - this variable contains the number of simulation units that it takes for a packet to arrive at the destination router. The value will start at one hundredth of the packet size, that is: `packetSize/100`. At every simulation time unit, this counter will decrease. Once it reaches 0, we can assume that the packet has arrived at the destination.**

Implement the following methods:

- **constructor (you may also include an overloaded constructor)**
- **getter/setters for all variables**
- **toString method that prints the Packet object in the format:**
  `[id, timeArrive, timeToDest]`
  E.g. Packet #18 arrive at simulation unit 11 with packet size 1234 should initially look like: `[18, 11, 12]`

2. `public class Router` - This class represents a router in the network, which is ultimately a queue. If you use a Java API class, you must use inheritance (extend a Java API class) to simplify the class definition. To make it simple, we will use the same definitions for all the different types of routers in the network. Include the following methods for queue functionality:

- **`public Router` - constructor (you may overload the constructor)**
- **`public void enqueue(Packet p)` - adds a new Packet to the end of the router buffer.**
- **`public Packet dequeue()` - removes the first Packet in the router buffer.**
- **`public Packet peek()` - returns, but does not remove the first Packet in the router buffer.**
- **`public int size()` - returns the number of Packets that are in the router buffer.**
- **`public boolean isEmpty()` - returns whether the router buffer is empty or not.**

- **`public String toString()` - returns a String representation of the router buffer in the following format:**
  {[packet1], [packet2], ... , [packetN]} (See sample I/O if you are confused.)

Now implement additional methods for our customized functionality:

- **`public static int sendPacketTo(Collection routers)` - this method should loop through the list Intermediate routers. Find the router with the most free buffer space (contains least Packets), and return the index of the router. If there are multiple routers, any corresponding indices will be acceptable. If all router buffers are full, throw an exception. You must handle this in your code.**

NOTE: You will need to group all Intermediate routers together in a group. A simple array or a Collection is an abstraction of the idea. You may use any data structure you are familiar with and comfortable using. This includes Router[], ArrayList, LinkedList, etc. Each router has a corresponding number, which is the index it resides in on your list. For the scope of this assignment, indices should start at 1.

3. `public class Simulator` - This class contains the main method that tests your simulation. You should not use hard-coded numbers. Instead, all values should be received from user input. The following contains the list of parameters you must keep track of:

- **`Router dispatcher` - Level 1 router**
- **`Collection<Router> routers` - Level 2 routers**
- **`int totalServiceTime` - contains the running sum of the total time each packet is in the network. The service time per packet is simply *the time it has arrived to the Destination minus the time when the packet was created*. When a packet counter reaches 0, dequeue it from the router queue and add the time to the total time. Ignore the leftover Packets in the network when simulation time is up.**
- **`int totalPacketsArrived` - contains the total number of packets that has been successfully forwarded to the destination. When a packet counter reaches 0, dequeue it from the router queue and increase this count by 1.**
- **`int packetsDropped` - records the number of packets that have been dropped due to a congested network.**
  Note: this can only happen when `sendPacketTo(Collection routers)` throws an exception.
- **`double arrivalProb` - the probability of a new packet arriving at the Dispatcher.**
- **`int numIntRouters` - the number of Intermediate routers in the network.**
- **`int maxBufferSize` - the maximum number of Packets a Router can accommodate for.**
- **`int minPacketSize` - the minimum size of a Packet**
- **`int maxPacketSize` - the maximum size of a Packet**
- **`int bandwidth` - the maximum number of Packets the Destination router can accept at a given simulation unit**
- **`int duration` - the number of simulation units**

Implement the following methods:

- **`public double simulate(/* any arguments above you find fit */)` - runs the simulator as described in the specs. Calculate and return the average time each packet spends within the network.**
- **`private int randInt(int minVal, int maxVal)` - this will be your helper method that can generate a random number between minVal and maxVal, inclusively. Return that randomly generated number.**
- **`public static void main(String[] args)` - the main() method will prompt the user for inputs to the simulator. It will then run the simulator, and outputs the result. Prompt the user whether he or she wants to run another simulation.**

4. Any exception class that you find fit.

SAMPLE INPUT/OUTPUT:

```
// comment in green, input in red, output in black

// Your output should following the following out format. Do not expect the numbers to be the same.

Starting simulator...

Enter the number of Intermediate routers: 4

Enter the arrival probability of a packet: 0.5

Enter the maximum buffer size of a router: 10

Enter the minimum size of a packet: 500

Enter the maximum size of a packet: 1500

Enter the bandwidth size: 2

Enter the simulation duration: 25

Time: 1
Packet 1 arrives at dispatcher with size 576.
Packet 2 arrives at dispatcher with size 1044.
Packet 1 sent to Router 1.
Packet 2 sent to Router 2.
R1: {[1, 1, 5]}
R2: {[2, 1, 10]}
R3: {}
R4: {}

Time: 2
Packet 3 arrives at dispatcher with size 922.
Packet 3 sent to Router 3.
R1: {[1, 1, 4]}
R2: {[2, 1, 9]}
R3: {[3, 2, 9]}
R4: {}

Time: 3
Packet 4 arrives at dispatcher with size 1301.
Packet 5 arrives at dispatcher with size 574.
Packet 4 sent to Router 4.
Packet 5 send to Router 1.
R1: {[1, 1, 3], [5, 3, 5]}
R2: {[2, 1, 8]}
R3: {[3, 2, 8]}
R4: {[4, 3, 13]}
```

```
Time: 4
Packet 6 arrives at dispatcher with size 1283.
Packet 7 arrives at dispatcher with size 552.
Packet 6 sent to Router 2.
Packet 7 send to Router 3.
R1: {[1, 1, 2], [5, 3, 5]}
R2: {[2, 1, 7], [6, 4, 12]}
R3: {[3, 2, 7], [7, 4, 5]}
R4: {[4, 3, 12]}

Time: 5
No packets arrived.
R1: {[1, 1, 1], [5, 3, 5]}
R2: {[2, 1, 6], [6, 4, 12]}
R3: {[3, 2, 6], [7, 4, 5]}
R4: {[4, 3, 11]}

Time: 6
Packet 8 arrives at dispatcher with size 900.
Packet 8 sent to Router 4.
Packet 1 has successfully reached its destination: +5
R1: {[5, 3, 5]}
R2: {[2, 1, 5], [6, 4, 12]}
R3: {[3, 2, 5], [7, 4, 5]}
R4: {[4, 3, 11], [8, 6, 9]}

// +5 means it took 5 simulation units for the packet to travel through the
network.

Time: 7
Packet 9 arrives at dispatcher with size 1410.
Packet 10 arrives at dispatcher with size 913.
Packet 9 sent to Router 1.
Packet 10 sent to Router 1.
R1: {[5, 3, 4], [9, 7, 14], [10, 7, 9]}
R2: {[2, 1, 4], [6, 4, 12]}
R3: {[3, 2, 4], [7, 4, 5]}
R4: {[4, 3, 10], [8, 6, 9]}

// According to our selection algorithm, both packets should end up in
Router 1.

// Time 8 - 10 not shown in sample i/o

Time: 11
Packet 17 arrives at dispatcher with size 830.
Packet 17 sent to Router 3.
Packet 5 has successfully reached its destination: +8
Packet 2 has successfully reached its destination: +10
R1: {[9, 7, 14], [10, 7, 9], ... ... ...}
R2: {[6, 4, 12], ... ... ...}
```

```
R3: {[3, 2, 0], [7, 4, 5], ... ... ...} // Packet 3 will stay in the router buffer
R4: {[4, 3, 6], [8, 6, 9], ... ... ...}

// Packet 3 will not be accepted by the Destination router due to limited
bandwidth. It will stay in the router queue until processed in the next
simulation unit.

Time: 12
Packet 18 arrives at dispatcher with size 1201.
Packet 19 arrives at dispatcher with size 667.
Packet 20 arrives at dispatcher with size 920.
Packet 18 sent to Router 1.
Packet 19 sent to Router 2.
Packet 20 sent to Router 4.
Packet 3 has successfully reached its destination: +10
R1: {[9, 7, 13], [10, 7, 9], ... ... ...}
R2: {[6, 4, 11], ... ... ...}
R3: {[7, 4, 5], ... ... ...}
R4: {[4, 3, 5], [8, 6, 9], ... ... ...}

// Time 13 - 23 not shown in sample i/o

Time: 24
Packet 45 arrives at dispatcher with size 1008.
Packet 46 arrives at dispatcher with size 573.
Packet 45 sent to Router 3.
Network is congested. Packet 46 is dropped.
Packet 6 has successfully reached its destination: +20
R1: {[9, 7, 13], [10, 7, 9], ... ... ...}
R2: {[[11, 8, 7], ... ... ...}
R3: {[12, 8, 6], ... ... ...}
R4: {[8, 6, 2], ... ... ...}

// If all router queues are full, we drop the packet. It cannot be sent
through the network.

Time: 25
Packet 47 arrives at dispatcher with size 710.
Packet 48 arrives at dispatcher with size 993.
Packet 47 sent to Router 2.
Network is congested. Packet 48 is dropped.
R1: {[9, 7, 12], [10, 7, 9], ... ... ...}
R2: {[[11, 8, 6], ... ... ...}
R3: {[12, 8, 5], ... ... ...}
R4: {[8, 6, 1], ... ... ...}

Simulation ending...
Total service time: 70
Total packets served: 6
Average service time per packet: 11.67
Total packets dropped: 2
```

```
// This is the final result output. You will be graded on the trends
through simulations with various parameters. You can find the requirements
on the grading sheet for this assignment.

Do you want to try another simulation? (y/n): n

Program terminating successfully...
```

Course Info | Schedule | Sections | Announcements | Homework | Exams | Help/FAQ | Grades | HOME