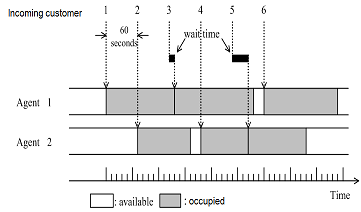
## Read the following description of a C# program and the program itself, and then answer Subquestions 1 and 2.

[Program Description]

This program is a simulator that simulates the airline ticket sell operation of Company A. Company A uses this simulator to estimate the time required for a customer to buy airline tickets. A customer is always serviced by a single seller .

Assume for this simulator that as long as a seller is available, no customer has to wait. For instance, assume six customers arrived to the ticket sell center at 60-second intervals when two sellers were available. The duration of each customer’s incomming was 130 seconds, 100 seconds, 150 seconds, 90 seconds, 110 seconds, and 140 seconds respectively. This means that the wait time of the third customer was 10 seconds, the fifth customer 30 seconds, and the others 0 seconds. (See Figure 1.)

Note that this simulator simulates one second in the real world in 0.1 second.



**Fig.1 Customer’s wait time**

This program consists of the following classes:

(1) SellCenter

This class represents the ticket sell center. It performs simulation after specifying the number of sellers, array of ticket selling times, and intervals that generate customers in the arguments of the constructor. It consists of the methods described below.

public static void main(String[] args)

This method starts the simulator for testing.

Customer getCustomer()

This method is called by method sellTicket(). It returns a Customer object . This method makes the thread wait until a customet can be allocated to the seller . If there remains no customet to be allocated after all customers generated, it returns null.

void sellTicket()

This method simulates the seller, and is processed in an individual thread. . It repeats the processing until all the customers generated by SellCenter have been allocated.

(2) Customer

This class represents a customer. The duration of ticket buying time is specified in the argument of the constructor. This class consists of the methods described below.

public void BuyTicket()

This method simulates the status during the buying ticket between a seller and the customer. This method displays the wait time of a customer allocated to the seller in second units (rounded off), and then stops the selling thread only for the time specified as ticket buying time.

Figure 2 shows the results of the program when the example of Figure 1 was simulated.

Note that (s) represents seconds.

|  |
| --- |
| 0(s)  0(s)  10(s)  0(s)  30(s)  0(s) |

Fig. 2 Results of simulation

[Program 1]

class SellCenter

{

private static Queue<Customer> waitingList;

private static bool running = true;

static void Main(string[] args)

{

int noSellers = 2;

int[] durations = { 130, 100, 150, 90, 110, 140 };

int interval = 60; // Interval for incomming customer (seconds)

new SellCenter(noSellers, durations, interval);

}

public SellCenter(int noSellers, int[] durations, int interval)

{

for (int i = 0; i < noSellers; i++)

{

Thread t = new Thread(new ThreadStart(sellTicket));

t.Start();

}

waitingList = new Queue<Customer>();

long nextTime = DateTime.Now.Ticks / TimeSpan.TicksPerMillisecond;

for (int i = 0; i < durations.Length; i++)

{

lock (waitingList)

{

waitingList.Enqueue(new Customer(durations[i]));

Monitor.Pulse(waitingList);

}

// Wait until next customer is generated.

nextTime += interval \* 100; // Operates 10 times faster.

// A

long sleeping = nextTime - DateTime.Now.Ticks / TimeSpan.TicksPerMillisecond;

if (sleeping > 0) Thread.Sleep((int)sleeping);

}

// End all incomming customers.

running = false;

// B

lock (waitingList)

{

Monitor.PulseAll(waitingList);

}

}

Customer getCustomer()

{

lock (waitingList)

{

// C

while (waitingList.Count == 0 && running == true)

{

// D

Monitor.Wait(waitingList);

}

if (waitingList.Count == 0) return null;

return waitingList.Dequeue();

}

}

void sellTicket()

{

Customer c;

// E

while ((c = getCustomer()) != null)

{

c.BuyTicket();

}

}

}

[Program 2]

class Customer

{

private readonly int duration;

private readonly long startTime;

public Customer(int duration)

{

startTime = DateTime.Now.Ticks / TimeSpan.TicksPerMillisecond;

this.duration = duration;

}

public void BuyTicket()

{

// F

long elapsed = DateTime.Now.Ticks / TimeSpan.TicksPerMillisecond - startTime;

Console.WriteLine("{0}s", elapsed / 100);

Thread.Sleep(duration \* 100);

}

}

Subquestion 1

Fill into the blanks  in the above programs.

Subquestion 2

A SellCenter instance was generated, as shown below, what is the number of sellers occupied after 80 seconds (8 seconds in real time) have elapsed in the simulator.

new SellCenter(3, new long[]{70, 90, 100, 110}, 30);