**CSC 3020 – Java Programming**

**Homework 6 – [your name]**

**25 points – Due April 18, 10am**

**Late deadline is April 20, 11:59pm, but 20% off**

**a)** Save this document with your name and the homework number somewhere in the file name.

**b)** Type/paste your answers into the document.

**c)** Submit this document and your .java file(s) to the Blackboard item where you downloaded this document. Do not submit a zip file but individually attach your files.

You have been hired by *Millennial Mart* to write a Java console application that processes customers through their store checkout line. The application has the following two classes:

**Customer.java**

Each object created from this class represents one customer and includes the following fields and methods:

**Fields**

● (static) totalCustomers – count of all distinct customers; initialize to 0 in declaration.

● (static) totalItems – total items purchased by all customers; initialize to 0 in declaration.

● (static) totalCost – total cost of all items purchased by all customers; initialize to 0 in declaration.

● ID – unique number assigned to each customer.

● items – number of items in customer shopping cart.

● cost – cost of all items in customer shopping cart.

● next – pointer to next customer.

**Methods**

● A constructor with no parameters that sets the fields, respectively, to these values:

totalCustomers = totalCustomers + 1

ID = -1

items = -1

cost = -1

next = null

● A constructor with two parameters that sets the fields, respectively, to these values:

totalCustomers = totalCustomers + 1

totalItems = totalItems + items

totalCost = totalCost + cost

ID – set to totalCustomers

items – set from parameter

cost – set from parameter

next = null

● Getter methods for each field (declare the getters for totalCustomers, totalItems, and totalCost static).

● Setter methods for each field (declare the setters for totalCustomers, totalItems, and totalCost static).

● *equals* method that compares ID for equality.

● *toString* method for returning ID, items, and cost values only.

**HW6.java**

This class contains the main method and uses the Customer class to represent each customer in the store checkout line. The line is represented as a queue (first-in first-out). Begin with an empty queue. Create the following three methods:

● **void insertCustomer(Customer ptr)** – this method adds a customer to the queue.

● **Customer removeCustomer()** – this method removes a customer from the queue.

● **void printCustomers()** – this method prints any customers in the queue. It prints the ID, items, and cost for each customer formatted in three columns.

In the **main method**, loop twenty times. Within each loop, either add a customer to the queue or remove a customer from the queue, and then print the customers in the queue. Add a customer if the queue is empty or if a randomly generated value is true (use random function **nextBoolean** for this). Otherwise, remove a customer. Print a message showing the ID of the customer added or removed. When adding a customer (creating an object of the Customer class):

● Generate a random number between 1 and 40 for their number of items.

● Generate a random number between 1 and 200 for their cost.

Keep track of the number of customers in the queue, and the number of customers that have left the queue. At the bottom of each loop, pause execution for one second with **Thread.sleep(1000);** After the loop, print:

● The number of customers that have left the queue.

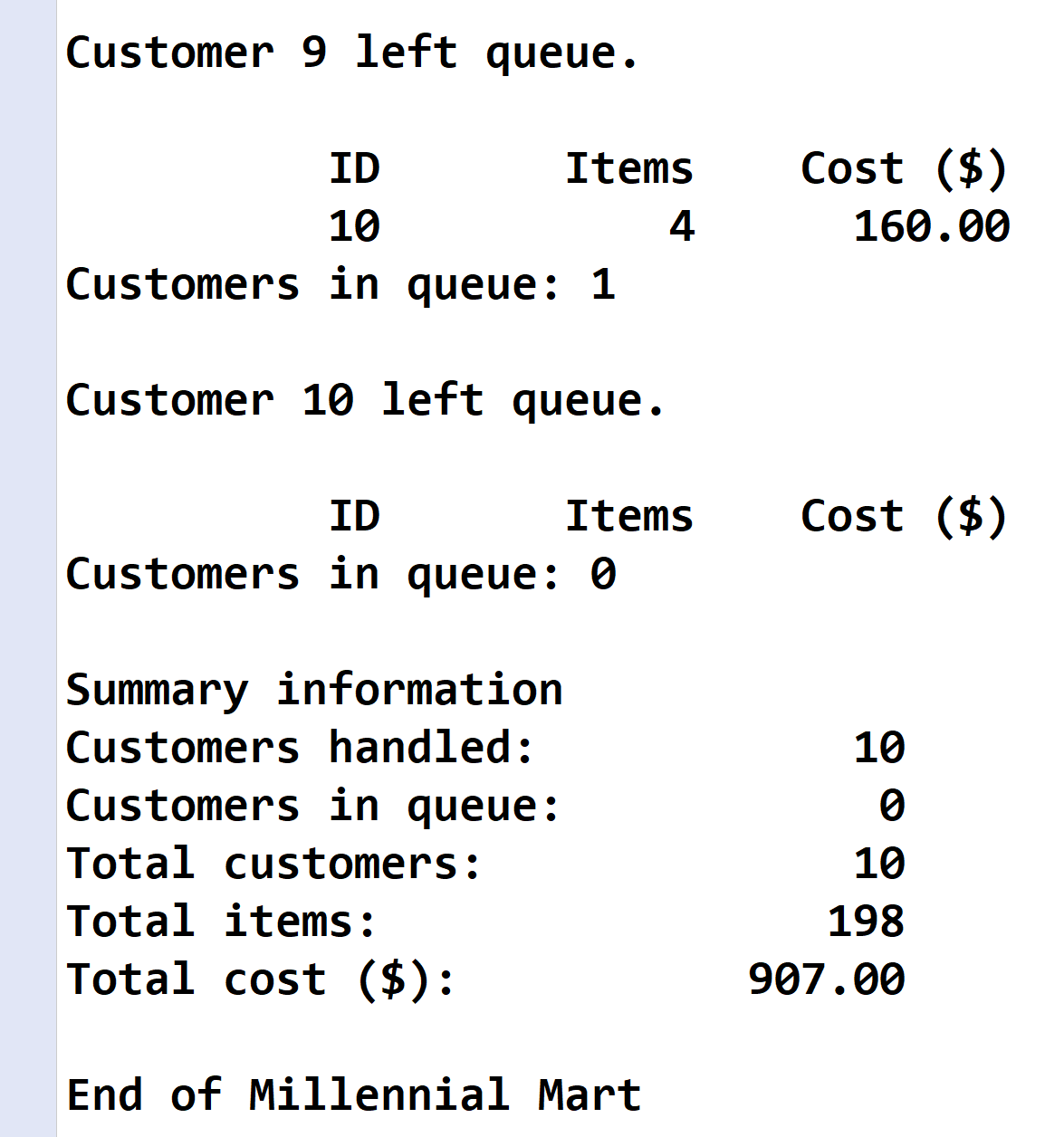
● The number still in the queue.

● The total number of customers handled.

● The total items purchased by all customers.

● The total cost of all items purchased by all customers.

Each run will generate different results. Here is sample output from the end of one run:



**Customer.java**

*[your Customer.java code here]\**

**If possible, format your code like this:**

**Font “Courier New”**

**Font size “9”**

**Bold**

**HW6.java**

*[your HW6.java code here]\**

**If possible, format your code like this:**

**Font “Courier New”**

**Font size “9”**

**Bold**

**Program output**

*[your program output here]\*\**

\* **Copying-and-pasting application code to a Word document**

1) From the program editor window, press **CTRL-A** and press **CTRL-C**.

2) From within the Word document, press **CTRL-V**.

\*\* **Copying-and-pasting application output to a Word document**

1) From the Eclipse main screen, maximize the Console window.

2) From the Console window, press **ALT-PrintScreen**.

3) From within the Word document, press **CTRL-V**.