

# COT5405: Programming Assignment 1 (Fall 2017)

**Name: Bhagyashree Gawade**

**Pid: 4166520**

## **Purpose of assignment:**

To learn the working and implementation of Algorithms involving scheduling.

**Language Used:** Java

## **Compilation Instructions:**

Compile and Run the Java file on Eclipse IDE or through command prompt using the command:

To compile: `javac filename.java`

To run: `java filename.java`

## **In this scenario:**

To compile: `javac Barber.java`

To run: `java Barber.java`

## **Inputs used for this program are as follows:**

Barbers String Array:

```
private String[] Barbers = {"A", "B", "C", "D", "E"};
```

Time String Array:

```
String[] time = {"8:00", "8:25", "8:05", "8:15", "8:40"};
```

**Platform:** Windows, Ubuntu/Linux, MacOS

## **Tasks:**

### **1. Explanation of the Algorithm and design:**

- This algorithm schedules work for five barbers. The five barbers are taken in a String Array. Each barber arrives at different time. Their arrival times do not differ more than a couple of minutes.
- In my program, 5 barbers sign in at the following timings:
  - A: 8:00 am**
  - B: 8:25 am**
  - C: 8:05 am**
  - D: 8:15 am**
  - E: 8:40 am**
- Now, I sorted the two string arrays together to find the barbers in ascending order of sign in time.

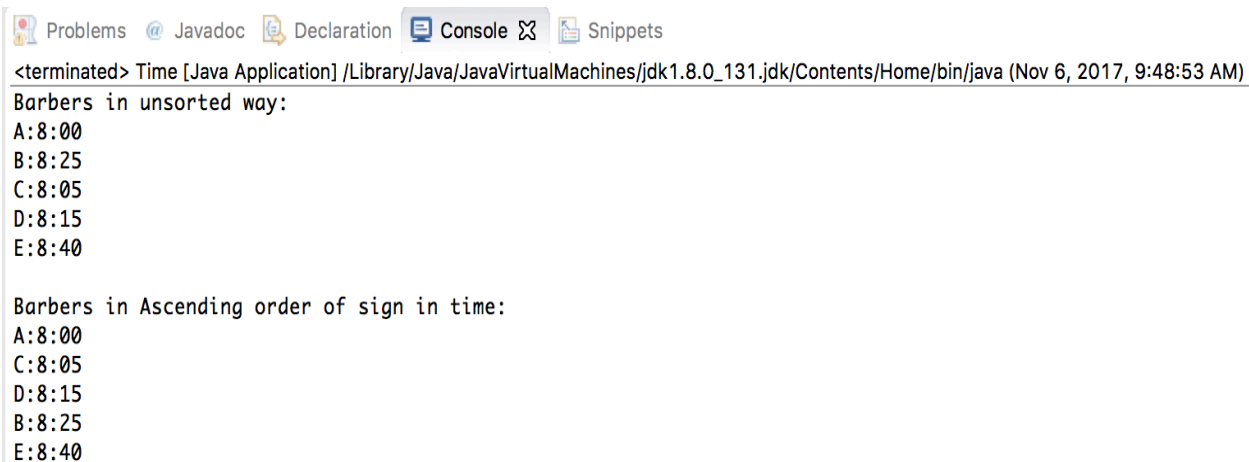
- The following code snippet shows the sorting:

```

public void sortTime(String time[])
{
    System.out.println("Barbers in unsorted way:");
    for(int n=0;n<5;n++)
    {
        System.out.print (Barbers[n] + ":");
        System.out.println (time[n]);
    }
    System.out.println (" ");
    for ( int i = 0; i < 5; i++) {
        for (int j = i + 1; j < 5; j++) {
            String dtmp=null;
            String stmp=null;
            if (time[i].compareTo(time[j]) >0) {
                dtmp = Barbers[i];
                Barbers[i] = Barbers[j];
                Barbers[j] = dtmp;
                stmp = time[i];
                time[i]=time[j];
                time[j]=stmp;
            }
        }
    }
}

```

- The following screenshot shows the output:



Problems @ Javadoc Declaration Console Snippets

<terminated> Time [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0\_131.jdk/Contents/Home/bin/java (Nov 6, 2017, 9:48:53 AM)

Barbers in unsorted way:

A:8:00  
B:8:25  
C:8:05  
D:8:15  
E:8:40

Barbers in Ascending order of sign in time:

A:8:00  
C:8:05  
D:8:15  
B:8:25  
E:8:40

- Once my 5 barbers got sorted in Ascending order of sign in time, I can assign them work in the first round based on FIRST-COME, FIRST-WORK.
- There are four types of services, costing \$10, \$20, \$30, \$40.

- I have Taken an array [] a for these service costs and then picked up the services randomly by using the logic of **Random number generation from a given array**.
- I created an array [] customer\_service which stores the randomly generated services for 5 barbers.
- The following snippet shows the same:

```
public void rndFunc(){
    int[]a= new int[]{10,30,40,20};

    int customer_service[];    //declaring array
    customer_service = new int[customer]; // allocating memory

    for (int i = 0; i < customer_service.length; i++){
        Random rnd= new Random();
        customer_service[i] = a[rnd.nextInt(a.length)];
    }
}
```

- I am calling this function using t.rndFunc();
- The following screenshot shows hoe the customer services are generated randomly and the earning of the 5 barbers after execution of Round 1:

Round 1

[A, C, D, B, E]

Service requested by the customer costs: [40, 30, 20, 30, 10]

Earing of barbers after round: 1

[40, 30, 20, 30, 10]

- After the first Round the Barbers are assigned services considering the money earned so far. The barber with less Earning can be moved ahead of those with more Earning. In my case, the rule is the difference is d=\$20.
- Whenever the barber in front has Earning \$20 or more than the Earning of Barber later in the array, the barber is moved to that position and the barber with more Earning is shifted in the position of Barber with less Earning.
- This way, the barbers with less earning get a chance to get services first.
- Here, I have used a Compare() Method to check the Difference between Earnings. The following snippet shows the execution:

```
private void Compare(){
    for (int i = 0; i < 5; i++) {
        for (int j = i + 1; j < 5; j++) {
            if (Earning[i] >=(Earning[j] +20)) {
                arrange(i,j);
            }
        }
    }
}
```

- Then I used an arrange method to arrange the Barbers and their Earnings according to the difference:

```
private void arrange(int larger, int smaller)
{
    int tempEarning[] = new int[5];
    String tempBarber[] = new String[5];
    //earnings
    System.arraycopy(Earning, 0, tempEarning, 0, 5);
    tempEarning[larger] = Earning[smaller];
    System.arraycopy(Earning, larger, tempEarning, larger+1, smaller-larger);
    System.arraycopy(Earning, smaller+1, tempEarning, smaller+1, Earning.length-smaller-1);
    Earning = tempEarning;
    //barbers
    System.arraycopy(Barbers, 0, tempBarber, 0, 5);
    tempBarber[larger] = Barbers[smaller];
    System.arraycopy(Barbers, larger, tempBarber, larger+1, smaller-larger);
    System.arraycopy(Barbers, smaller+1, tempBarber, smaller+1, Barbers.length-smaller-1);
    Barbers = tempBarber;
}
}
```

The final Output of my program is as shown in the screenshot below:

The output changes every time I run as the customer services are generated randomly,

The following two screenshot show that the final earnings of Barbers differ by \$10.

```
<terminated> Time [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_131.jdk/Contents/Home/bin/java (Nov 6, 2017, 7:12:52 PM)
Barbers in unsorted way:
A:8:00
B:8:25
C:8:05
D:8:15
E:8:40

Barbers in Ascending order of sign in time:
A:8:00
C:8:05
D:8:15
B:8:25
E:8:40

Round 1
[A, C, D, B, E]
Service requested by the customer costs: [30, 20, 30, 20, 10]
Earning of barbers after round: 1
[30, 20, 30, 20, 10]

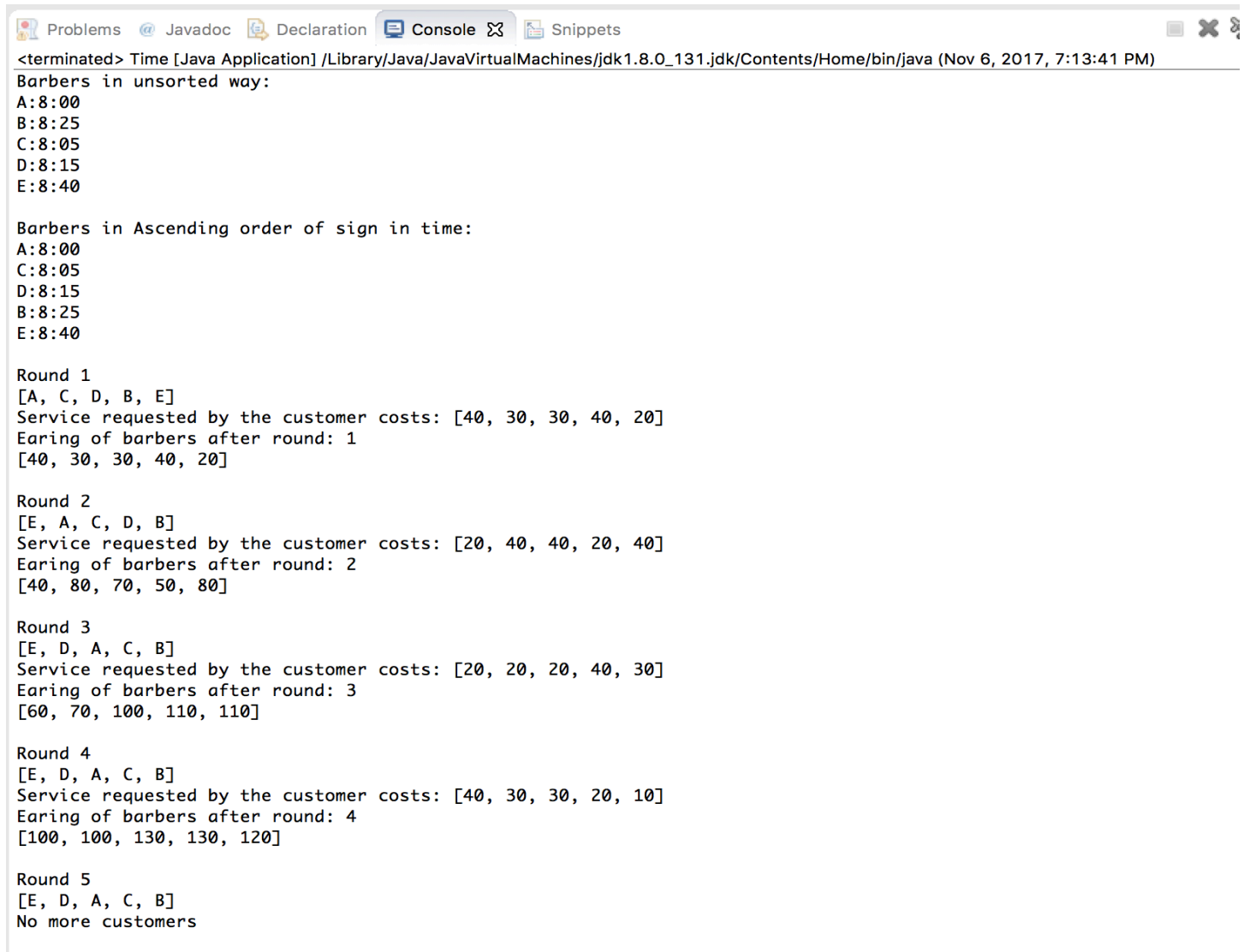
Round 2
[E, A, C, D, B]
Service requested by the customer costs: [30, 10, 20, 30, 20]
Earning of barbers after round: 2
[40, 40, 40, 60, 40]

Round 3
[E, A, C, B, D]
Service requested by the customer costs: [40, 30, 40, 20, 30]
Earning of barbers after round: 3
[80, 70, 80, 60, 90]

Round 4
[B, E, A, C, D]
Service requested by the customer costs: [40, 10, 20, 20, 10]
Earning of barbers after round: 4
[100, 90, 90, 100, 100]

Round 5
[B, E, A, C, D]
No more customers
```

The following two screenshots show that the final earnings of Barbers differ by \$30.



```
<terminated> Time [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_131.jdk/Contents/Home/bin/java (Nov 6, 2017, 7:13:41 PM)
Barbers in unsorted way:
A:8:00
B:8:25
C:8:05
D:8:15
E:8:40

Barbers in Ascending order of sign in time:
A:8:00
C:8:05
D:8:15
B:8:25
E:8:40

Round 1
[A, C, D, B, E]
Service requested by the customer costs: [40, 30, 30, 40, 20]
Earning of barbers after round: 1
[40, 30, 30, 40, 20]

Round 2
[E, A, C, D, B]
Service requested by the customer costs: [20, 40, 40, 20, 40]
Earning of barbers after round: 2
[40, 80, 70, 50, 80]

Round 3
[E, D, A, C, B]
Service requested by the customer costs: [20, 20, 20, 40, 30]
Earning of barbers after round: 3
[60, 70, 100, 110, 110]

Round 4
[E, D, A, C, B]
Service requested by the customer costs: [40, 30, 30, 20, 10]
Earning of barbers after round: 4
[100, 100, 130, 130, 120]

Round 5
[E, D, A, C, B]
No more customers
```

**Q.2) Show that your algorithm can guarantee their income at end of day do not differ more than  $S$  dollars.  $S$  is what your algorithm can guarantee, which can be a function of rounds and/or the price difference of the offered services.**

**Answer:** The above two screenshots show that once the income differs by \$10 and when ran for the second time the income differs by \$30.

So, it depends on the customer services. Screenshot 2 shows that, my algorithm guarantees that the difference of the income at the end of the day will not differ by  $S = \text{Max} - \text{Min}$  Income of barbers in the last round. Here, the Max income of barbers is \$130 and Min income of Barbers is \$100. So, the difference between the income between any barbers will not differ by more than \$30 in the last round.

The algorithm guarantees:  $S < (\text{Max} - \text{Min}) \times \text{Number of rounds}$ .

Where, Max = Maximum cost of services

Min = Minimum cost of services

### Q.3) What is the running time of your algorithm?

**Answer:** The runtime of my algorithm is  $O(n^2)$ .

### Q.4) If the barbers could have more than one assignment in a round, would $S$ be smaller?

**Answer:** Yes, if the barbers have more than one assignment in a round,  $S$  would be smaller. As the MIN earning of barbers would be greater than Min income of barbers we found before. Because, they will receive more number of assignments and hence the difference can be made smaller.

### Q.5) Should the above requirements be modified or dropped, and/or additional requirements be added? What might they be?


The above requirements can be modified and additional requirements can be added.

- 1) This can be done by assigning more than one assignments to barbers in a round only when the difference is more than \$20 and a new assignment of cost \$30 or \$40 comes up. This way the barbers with less income will get the customer service of more cost.
- 2) If we assign the customer service costing \$40 to the barber who is moved ahead of others with more income, then the income of the barbers won't differ much by the end of the day.

### Q.6) Implement your algorithm and try it on some input.

**Answer:**

- 1) I considered 20 customers as my first input. This leads to 4 rounds. The following screenshot shows the output:



```
<terminated> Time [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_131.jdk/Contents/Home/bin/java (Nov 6, 2017, 7:13:41 PM)
Barbers in unsorted way:
A:8:00
B:8:25
C:8:05
D:8:15
E:8:40

Barbers in Ascending order of sign in time:
A:8:00
C:8:05
D:8:15
B:8:25
E:8:40

Round 1
[A, C, D, B, E]
Service requested by the customer costs: [40, 30, 30, 40, 20]
Earning of barbers after round: 1
[40, 30, 30, 40, 20]

Round 2
[E, A, C, D, B]
Service requested by the customer costs: [20, 40, 40, 20, 40]
Earning of barbers after round: 2
[40, 80, 70, 50, 80]

Round 3
[E, D, A, C, B]
Service requested by the customer costs: [20, 20, 20, 40, 30]
Earning of barbers after round: 3
[60, 70, 100, 110, 110]

Round 4
[E, D, A, C, B]
Service requested by the customer costs: [40, 30, 30, 20, 10]
Earning of barbers after round: 4
[100, 100, 130, 130, 120]

Round 5
[E, D, A, C, B]
No more customers
```

- 2) I considered the number to customers to be 40. This leads to 8 rounds. The following 2 screenshots shows the output:

```
Problems Javadoc Declaration Console Snippets
<terminated> Time [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_131.jdk/Contents/Home/bin/java (Nov 6, 2017, 7:55:47 PM)
Barbers in unsorted way:
A:8:00
B:8:25
C:8:05
D:8:15
E:8:40

Barbers in Ascending order of sign in time:
A:8:00
C:8:05
D:8:15
B:8:25
E:8:40

Round 1
[A, C, D, B, E]
Service requested by the customer costs: [20, 10, 40, 30, 10]
Earing of barbers after round: 1
[20, 10, 40, 30, 10]

Round 2
[A, C, E, D, B]
Service requested by the customer costs: [30, 30, 20, 20, 20]
Earing of barbers after round: 2
[50, 40, 30, 60, 50]

Round 3
[E, A, C, D, B]
Service requested by the customer costs: [40, 30, 40, 10, 10]
Earing of barbers after round: 3
[70, 80, 80, 70, 60]

Round 4
[E, B, A, C, D]
Service requested by the customer costs: [20, 30, 10, 10, 10]
Earing of barbers after round: 4
[90, 90, 90, 90, 80]

Round 5
[E, B, A, C, D]
Service requested by the customer costs: [40, 10, 30, 30, 40]
Earing of barbers after round: 5
[130, 100, 120, 120, 120]

Round 6
[B, E, A, C, D]
Service requested by the customer costs: [10, 20, 30, 30, 40]
Earing of barbers after round: 6
[110, 150, 150, 150, 160]

Round 7
[B, E, A, C, D]
Service requested by the customer costs: [40, 10, 20, 40, 20]
Earing of barbers after round: 7
[150, 160, 170, 190, 180]

Round 8
[B, E, A, C, D]
Service requested by the customer costs: [20, 30, 40, 10, 30]
Earing of barbers after round: 8
[170, 190, 210, 200, 210]

Round 9
[B, E, A, C, D]
No more customers
```