CENG-IE PROJECT

MIDTERM REPORT

Industrial Engineering Students

Ceren Çınar

Mustafa Yüksel

Hasan Demirkoparan

Computer Engineering Students

Ahmet Topuz

Mustafa Çalık

ESKİŞEHİR TECHNICAL UNIVERSITY

ESKİŞEHİR

1. Problem Definition

Test Problem Test the software with the following problem. Harvey's Specialty Shop sells a popular mustard. The mustard costs Harvey \$20 a jar. Replenishment lead time is 4 months. Harvey uses a 25 percent annual interest rate and estimates the loss of goodwill cost as \$20 per jar in case of stockout. Bookkeeping expenses for placing an order is \$100. During the four-month replenishment period, Harvey estimates that he sells an average of 500 jars. and the standard deviation of demand is 100. Assume that demand is described by a normal distribution. How should Harvey control the replenishment of the mustard?

2. What we done so far

Initially, industrial engineering students developed a preliminary solution to the problem and subsequently transferred their findings to the computer engineering team for further development. To ensure a clear understanding of the problem, industrial engineering students designed a detailed flow chart outlining the main steps and logic.

This flowchart was then reviewed and enhanced by the computer engineering students, who restructured it into a more adaptable and interactive format.

Throughout the collaboration, both departments maintained active communication and exchanged regular feedback to improve the overall quality of the work.

The computer engineering students developed and executed the algorithm in Java, ensuring a high level of numerical accuracy. Once the program produced the final results, these outputs were forwarded to the industrial engineering team for verification and confirmation of their accuracy.

3. On-hand solution of the problem

3.1. Parameters

Holding cost (h) = $I * c = 0.25 * 20 = \$5$

Loss of goodwill cost (p) = \$20

Standard deviation of demand (σ) = 100 units

Unit cost (c) = \$20

Ordering cost (k) = \$100

Lead time (Ƭ) = 4 months = 1/3 year

Annual demand (λ) = $\mu = \lambda * Ƭ$ = 1500

Average demand during lead time (μ) = 500 units

Annual interest rate (I) = 0.25 (25%)

(Q, R) Systems

Q= Order quantity

R= Reorder point

3.2.Solution

First, we need to find Q0 and R0.

(1) Q0= $\sqrt{\frac{2*k*\lambda}{h}}$

The formula to be used for Ro and subsequent R calculations:

(1) 1 – F(R) = $\frac{Q*h}{p*\lambda}$
(2) After the F(R) value is found, the z and L(z) values corresponding to F(R) are found from the table.
(3) We will reach the R0 value by using the z value found.
R0= $\mu + \sigma * z$

The formula to calculate the n(R) value required for subsequent iterations.

(1) n(R) = $\sigma * L(z)$

a) After Q0 and R0 are calculated, in this step, the formula we will use in iterations until we reach the optimal (Q, R) value.

The formula to be used to calculate Qi and Ri:

(1) Qi= $\sqrt{\frac{2*\lambda*[k+p*n(R)]}{h}}$
(2) 1 – F(Ri) = $\frac{Q*h}{p*\lambda}$
(3) Ri= $\mu + \sigma * z$
(4) n(R) = $\sigma * L(z)$

b) We stop the iteration when Qi and Q(i-1) and Ri and R(i-1) become very close to each other or equal to each other.

Iteration-1

Q0= $\sqrt{\frac{2*100*1500}{5}} = 244.9489 = 245.00$

1 – F(R0) = $\frac{245*5}{20*1500} = 0.0483$

F(R0) = 1 – 0.0483 = 0.9591

When we look at a value close to 0.9591 from the z chart table, the z and L(z) values we find are:

z = 1.74 (approximate value)

L(z) = 0.0165 (approximate value)

R0= $500 + 1.74 * 100$ = 674.00

n(R0) = $100 * 0.0165 = 1.65$

Q0=245.00

R0=674.00

Iteration-2

Q1= $\sqrt{\frac{2*1500*[100+20*1.65]}{5}}$ = 282.4889 = 282.50

1 – F(R1) = $\frac{282.50*5}{20*1500} = 0.0470$

F(R1) = 1 – 0.0470 = 0.9529

z = 1.68 (approximate value)

L(z) = 0.0190 (approximate value)

R1= $500 + 100 * 1.68 = 668$

n(R1) = $100 * 0.0190 = 1.9$

Q1=282.50

R1=668

The values of Q0 and Q1 and R0 and R1 are not close to each other. Continue the iteration:

Iteration-3

Q2=$\sqrt{\frac{2*1500*[100+20*1.9]}{5}} = 287.7499 = 287.75$

1 – F(R2) = $\frac{287.75*5}{20*1500} = 0.0479$

F(R2) = 1 – 0.0479 = 0.9520

z = 1.68 (approximate value)

L(z) = 0.0190 (approximate value)

R2= $500 + 100 * 1.68 = 668$

n(R2) = $100 * 0.0190 = 1.9$

Q2=287.75

R2=668

Iteration-4

Q3=$\sqrt{\frac{2*1500*[100+20*1.9]}{5}} = 287.7499 = 287.75$

1 – F(R3) = $\frac{287.75*5}{20*1500} = 0.0479$

F(R3) = 1 – 0.0479 = 0.9520

z = 1.68 (approximate value)

L(z) = 0.0190 (approximate value)

R3= $500 + 100 * 1.68 = 668$

Q3=287.75, Q2=287.75

R3=668, R2=668

We found that Q3 and Q2 are equals and R2 and R3 are equals. So, our optimal Q&R values are:

(Optimal lot size, Reorder point) = (Q, R) = (287.75, 668)

We can say:

(Q, R) = (288, 668)

3.3.Needed Calculations

Safety Stock= Reorder Point - Average Selling of Jars (μ) = 668-500 = 168

Average annual holding cost= h*(Q/2) = 5*(288/2+168) = 1560

Setup cost (Annual Order Cost) =k* λ/Q = 100*1500/288 = 520.83

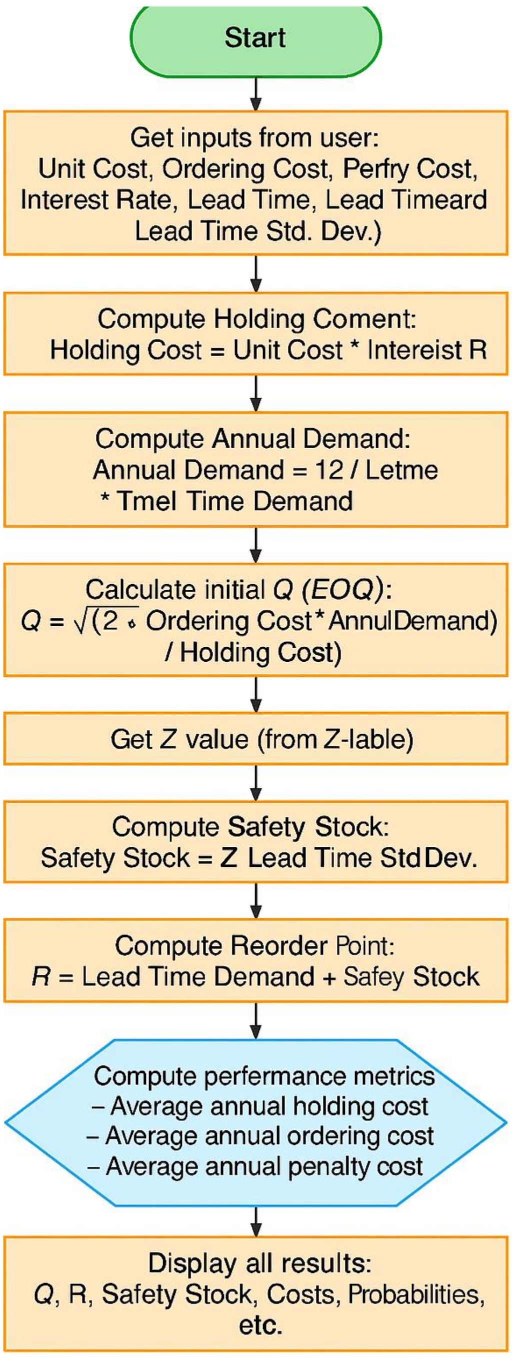
Penalty costs= (p* λ *n(R))/Q = (20*1500*1.9)/288 = 197.9167

Average time between the placement of orders= Q/ λ = 288/1500 = 0.192 years = 2.304 months

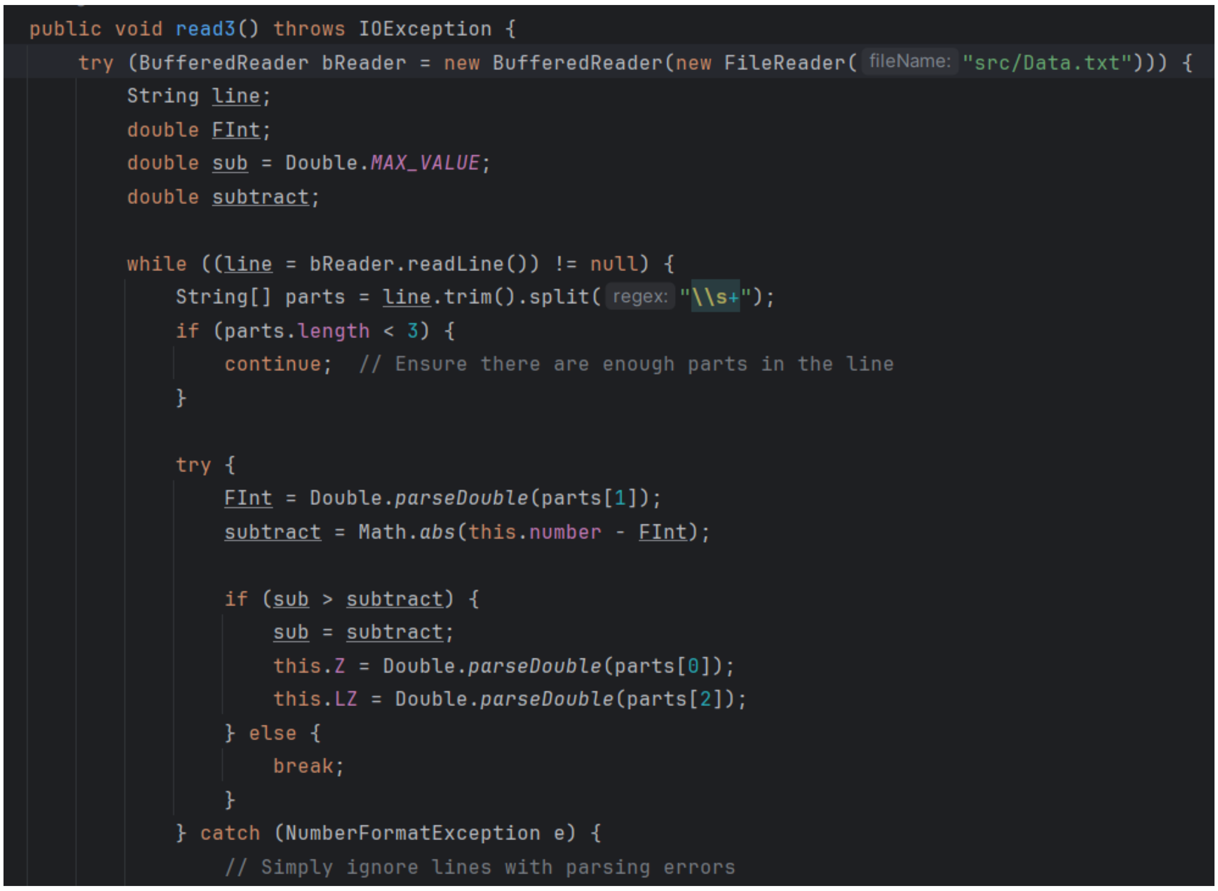
The proportion of order cycles in which no stock out occurs= F(R) = 0.952 = %95.2

The proportion of demands that are not met= n(R)/Q = 1.9/288 = 0.00659 = 0.0066 = %99.34 of the demands are satisfied as they occur.

4. Flowchart of the problem


Start
Get inputs from user:
Unit Cost, Ordering Cost, Perfry Cost,
Interest Rate, Lead Time, Lead Timeard
Lead Time Std. Dev.)
Compute Holding Coment:
Holding Cost = Unit Cost * Intereist R
Compute Annual Demand:
Annual Demand = 12 / Letme
* Tmel Time Demand
Calculate initial Q (EOQ):
Q = √(2 ∘ Ordering Cost * AnnulDemand)
/ Holding Cost)
Get Z value (from Z-lable)
Compute Safety Stock:
Safety Stock = Z Lead Time Std Dev.
Compute Reorder Point:
R = Lead Time Demand + Safey Stock
Compute perfermance metrics
– Average annual holding cost
– Average annual ordering cost
– Average annual penalty cost
Display all results:
Q, R, Safety Stock, Costs, Probabilities,
etc.

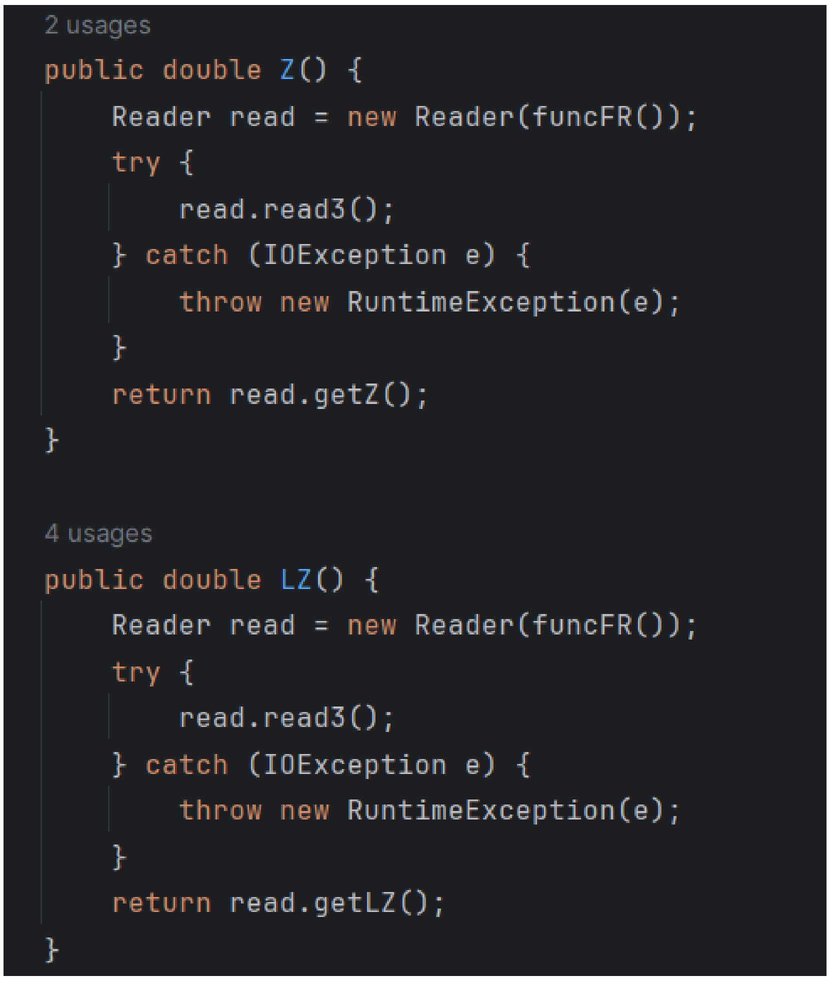
The Solution of the Problem:



```java
public void read3() throws IOException {
    try (BufferedReader bReader = new BufferedReader(new FileReader( fileName: "src/Data.txt"))) {
        String line;
        double FInt;
        double sub = Double.MAX_VALUE;
        double subtract;

        while ((line = bReader.readLine()) != null) {
            String[] parts = line.trim().split( regex: "\\s+");
            if (parts.length < 3) {
                continue;  // Ensure there are enough parts in the line
            }

            try {
                FInt = Double.parseDouble(parts[1]);
                subtract = Math.abs(this.number - FInt);

                if (sub > subtract) {
                    sub = subtract;
                    this.Z = Double.parseDouble(parts[0]);
                    this.LZ = Double.parseDouble(parts[2]);
                } else {
                    break;
                }
            } catch (NumberFormatException e) {
                // Simply ignore lines with parsing errors
```

Figure 1

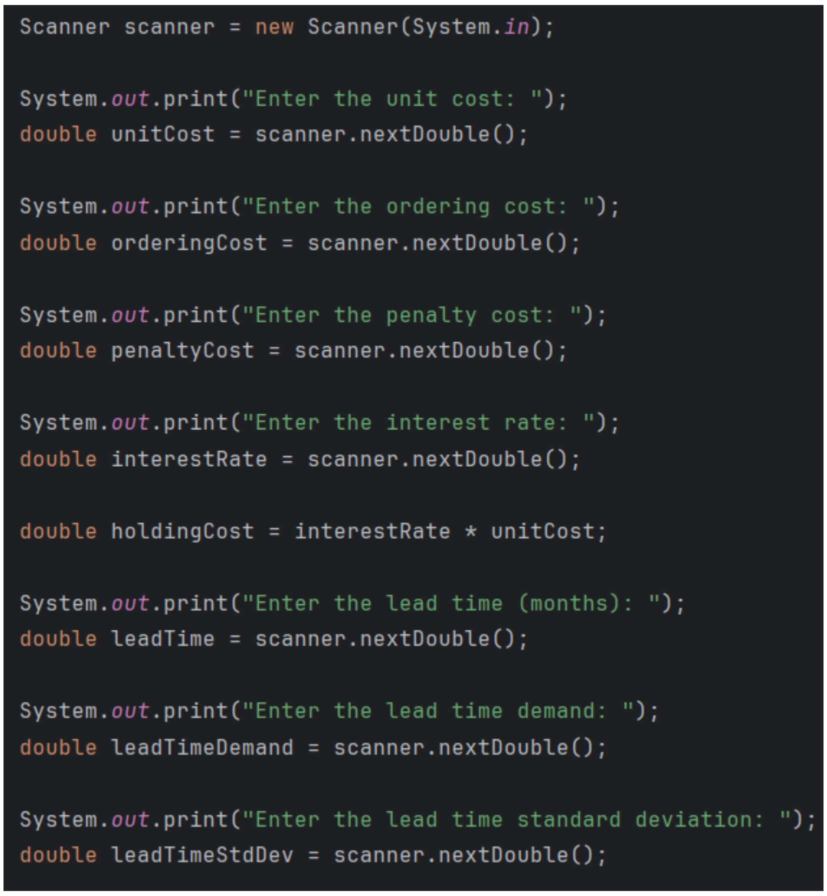
This code reads the file named "Data.txt" line by line, splitting each line by spaces. It parses the second element of each line as a double and calculates its absolute difference from a number stored as a class field. If this difference is the smallest found so far, it updates the class fields Z and LZ accordingly. Lines with fewer than three parts or lines that cause parsing errors are ignored.



```
2 usages
public double Z() {
    Reader read = new Reader(funcFR());
    try {
        read.read3();
    } catch (IOException e) {
        throw new RuntimeException(e);
    }
    return read.getZ();
}

4 usages
public double LZ() {
    Reader read = new Reader(funcFR());
    try {
        read.read3();
    } catch (IOException e) {
        throw new RuntimeException(e);
    }
    return read.getLZ();
}
```

Figure 2:

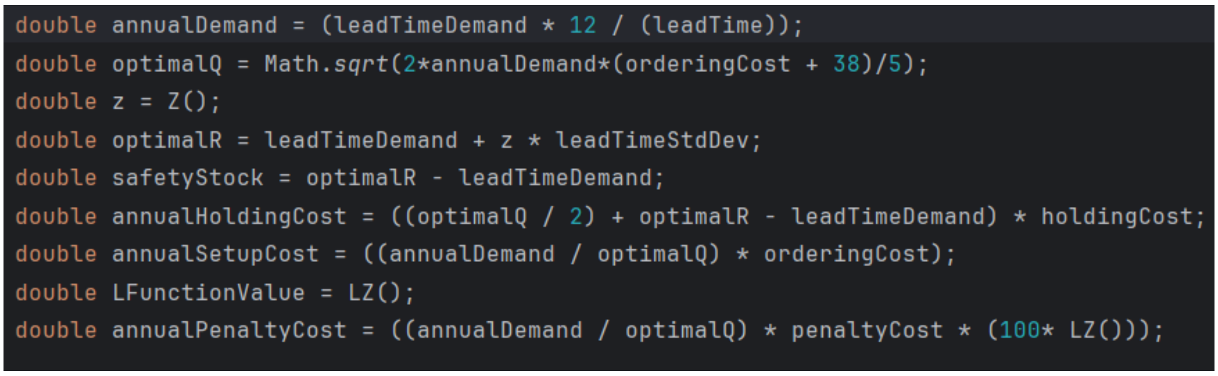
The methods Z() and LZ() create a Reader object initialized with funcFR(), then call the read3() method on this object to process the "Data.txt" file. During this process, the read3() method calculates and stores the values of Z and LZ within the object. The Z() method returns the calculated Z value, while LZ() returns the calculated LZ value.



```
Scanner scanner = new Scanner(System.in);

System.out.print("Enter the unit cost: ");
double unitCost = scanner.nextDouble();

System.out.print("Enter the ordering cost: ");
double orderingCost = scanner.nextDouble();

System.out.print("Enter the penalty cost: ");
double penaltyCost = scanner.nextDouble();

System.out.print("Enter the interest rate: ");
double interestRate = scanner.nextDouble();

double holdingCost = interestRate * unitCost;

System.out.print("Enter the lead time (months): ");
double leadTime = scanner.nextDouble();

System.out.print("Enter the lead time demand: ");
double leadTimeDemand = scanner.nextDouble();

System.out.print("Enter the lead time standard deviation: ");
double leadTimeStdDev = scanner.nextDouble();
```

Figure 3:

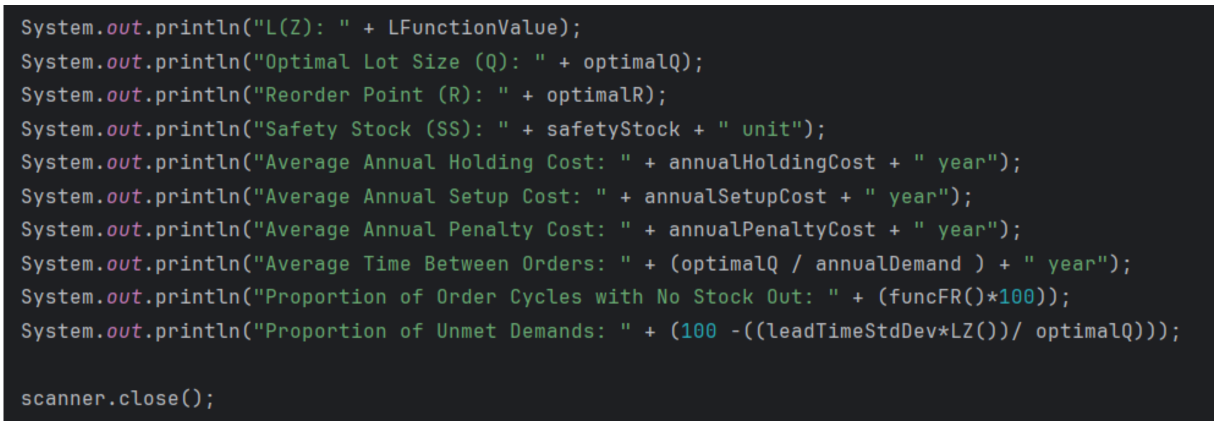
This code snippet uses a Scanner to prompt the user for various cost and parameter inputs related to inventory management. It collects unit cost, ordering cost, penalty cost, interest rate, lead time, lead time demand, and the standard deviation of the lead time demand. Then, it calculates the holding cost as the product of the interest rate and unit cost.



```
double annualDemand = (leadTimeDemand * 12 / (leadTime));
double optimalQ = Math.sqrt(2*annualDemand*(orderingCost + 38)/5);
double z = Z();
double optimalR = leadTimeDemand + z * leadTimeStdDev;
double safetyStock = optimalR - leadTimeDemand;
double annualHoldingCost = ((optimalQ / 2) + optimalR - leadTimeDemand) * holdingCost;
double annualSetupCost = ((annualDemand / optimalQ) * orderingCost);
double LFunctionValue = LZ();
double annualPenaltyCost = ((annualDemand / optimalQ) * penaltyCost * (100* LZ()));
```

Figure 4:

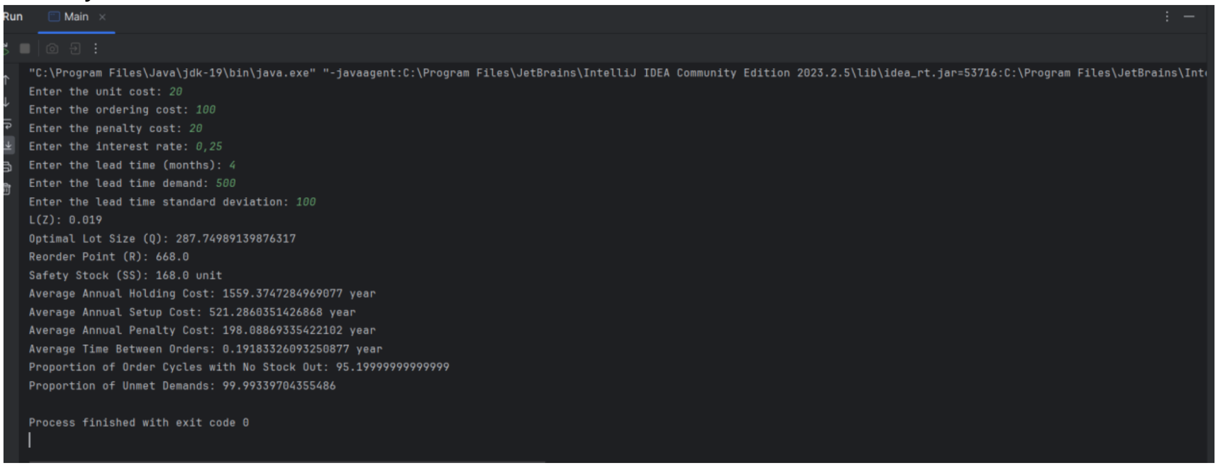
This section of code calculates key inventory management metrics including annual demand, Economic Order Quantity (EOQ), reorder point, safety stock, annual holding cost, annual setup cost, and annual penalty cost. The calculations utilize the user inputs along with the Z() and LZ() values obtained earlier. This approach applies the EOQ model to optimize inventory control by balancing holding, ordering, and penalty costs



```java
System.out.println("L(Z): " + LFunctionValue);
System.out.println("Optimal Lot Size (Q): " + optimalQ);
System.out.println("Reorder Point (R): " + optimalR);
System.out.println("Safety Stock (SS): " + safetyStock + " unit");
System.out.println("Average Annual Holding Cost: " + annualHoldingCost + " year");
System.out.println("Average Annual Setup Cost: " + annualSetupCost + " year");
System.out.println("Average Annual Penalty Cost: " + annualPenaltyCost + " year");
System.out.println("Average Time Between Orders: " + (optimalQ / annualDemand ) + " year");
System.out.println("Proportion of Order Cycles with No Stock Out: " + (funcFR()*100));
System.out.println("Proportion of Unmet Demands: " + (100 -((leadTimeStdDev*LZ())/ optimalQ)));

scanner.close();
```

Figure 5:

This code snippet prints the results of the inventory calculations to the console. This allows the user to see important parameters such as order quantity, reorder point, and safety stock directly.



```
"C:\Program Files\Java\jdk-19\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2023.2.5\lib\idea_rt.jar=53716:C:\Program Files\JetBrains\Int
Enter the unit cost: 20
Enter the ordering cost: 100
Enter the penalty cost: 20
Enter the interest rate: 0,25
Enter the lead time (months): 4
Enter the lead time demand: 500
Enter the lead time standard deviation: 100
L(Z): 0.019
Optimal Lot Size (Q): 287.74989139876317
Reorder Point (R): 668.0
Safety Stock (SS): 168.0 unit
Average Annual Holding Cost: 1559.3747284969077 year
Average Annual Setup Cost: 521.2860351426868 year
Average Annual Penalty Cost: 198.08869335422102 year
Average Time Between Orders: 0.19183326093250877 year
Proportion of Order Cycles with No Stock Out: 95.19999999999999
Proportion of Unmet Demands: 99.99339704355486

Process finished with exit code 0
```

Figure 6:

This part shows the console output or a sample display of the calculated results. It provides a visual confirmation of the computed values and helps verify the accuracy of the calculations.

ESKİŞEHİR TEKNİK ÜNİVERSİTESİ
ESKİŞEHİR TECHNICAL UNIVERSITY

Meeting Minute for the Interdisciplinary Project

Date:	22/05/2025

Please make sure that only the students attended to this meeting have signed this meeting minute.

	Name	Signature
CENG Student 1		
CENG Student 2	Mustafa Çalık	[illegible]
CENG Student 3	Ahmet Topuz	[illegible]
IE Student 1	Ceren Çınar	[illegible]
IE Student 2	Mustafa Yüksel	[illegible]
IE Student 3	Hasan Demirkoparan	[illegible]

Evaluations, decisions, and work that has been completed so far and planned to be done until the next meeting

In our third meeting, we mentioned that the tasks related to industrial engineering had been completed, and we focused on the Java code that would be developed solely by the computer engineering students.

As the process continued, they created a draft and shared the information with us. Later, they presented the final version of the code and explained its logic to help us understand it, although we had some difficulty grasping it.

Finally, they said they would organize and clarify the code further for inclusion in the reports, and the meeting was concluded.

ESKİŞEHİR TEKNİK ÜNİVERSİTESİ
ESKİŞEHİR TECHNICAL UNIVERSITY

Project Management

Management Order: Work Packages (WP), Task Distribution and Their Durations

The time in which the main work packages in the project will be carried out is given below as the “Worktime schedule”. The weeks in which the Meeting-minute report must be sent are shown in red. The deadline for the assignment in the Learning Management System is the last working day (Friday at 23:59) of the week shown in red. In each three minute-reports please include the name of the students by whom the work package(s) had been carried out. If you want, you can provide percentages if more than one student had had completed the task. Please do not include the name of a member, (if any) who has not participated to the task of the work package.

WORK-TIME SCHEDULE

Work Package No	Name of the Work Package	The job was done by whom? Name of the team member(s).	Weeks													
			1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	Meeting with group members and planning of the division of labor	Ceren Çınar Ahmet Topuz Mustafa Çalık Hasan Demirkoparan Mustafa Yüksel					🟩	🟩								
2	Analyzing and solving the problem manually (by hand) on paper.	Ceren Çınar Mustafa Yüksel Hasan Demirkoparan							🟪	🟪	🟪	🟪				
3	Implementation of the table lookup of the "Z-Chart & Loss Function" values.	Ahmet Topuz Mustafa Çalık							🟧	🟧	🟧	🟧				

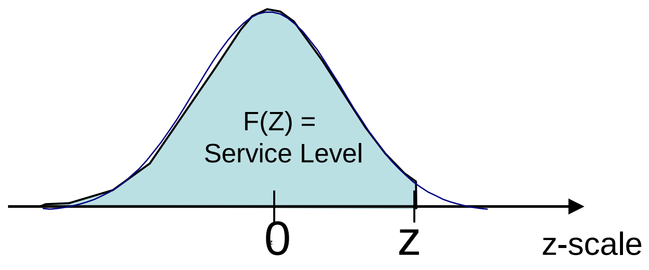
4	The implementation of the QR algorithm to solve the problem.	Ceren Çınar Hasan Demirkoparan Mustafa Yüksel											🟦	🟦	🟦	🟦
5	Writing a how-to-use documentation and testing the software to verify correctness of the code.	Ahmet Topuz Mustafa Çalık											🟦	🟦	🟦	🟦

Z-Chart & Loss Function

F(Z) is the probability that a variable from a standard normal distribution will be less than or equal to Z, or alternately, the service level for a quantity ordered with a z-value of Z.

L(Z) is the standard loss function, i.e. the expected number of lost sales as a fraction of the standard deviation. Hence, the lost sales = L(Z) x σ_{DEMAND}

Z	F(Z)	L(Z)
-3.00	**0.0013**	**3.000**
-2.96	**0.0015**	**2.960**
-2.92	**0.0018**	**2.921**
-2.88	**0.0020**	**2.881**
-2.84	**0.0023**	**2.841**
-2.80	**0.0026**	**2.801**
-2.76	**0.0029**	**2.761**
-2.72	**0.0033**	**2.721**
-2.68	**0.0037**	**2.681**
-2.64	**0.0041**	**2.641**
-2.60	**0.0047**	**2.601**
-2.56	**0.0052**	**2.562**
-2.52	**0.0059**	**2.522**
-2.48	**0.0066**	**2.482**
-2.44	**0.0073**	**2.442**
-2.40	**0.0082**	**2.403**
-2.36	**0.0091**	**2.363**
-2.32	**0.0102**	**2.323**
-2.28	**0.0113**	**2.284**
-2.24	**0.0125**	**2.244**
-2.20	**0.0139**	**2.205**
-2.16	**0.0154**	**2.165**
-2.12	**0.0170**	**2.126**
-2.08	**0.0188**	**2.087**
-2.04	**0.0207**	**2.048**
-2.00	**0.0228**	**2.008**
-1.96	**0.0250**	**1.969**
-1.92	**0.0274**	**1.930**
-1.88	**0.0301**	**1.892**
-1.84	**0.0329**	**1.853**
-1.80	**0.0359**	**1.814**
-1.76	**0.0392**	**1.776**
-1.72	**0.0427**	**1.737**
-1.68	**0.0465**	**1.699**
-1.64	**0.0505**	**1.661**
-1.60	**0.0548**	**1.623**
-1.56	**0.0594**	**1.586**
-1.52	**0.0643**	**1.548**
-1.48	**0.0694**	**1.511**
-1.44	**0.0749**	**1.474**
-1.40	**0.0808**	**1.437**
-1.36	**0.0869**	**1.400**
-1.32	**0.0934**	**1.364**
-1.28	**0.1003**	**1.327**
-1.24	**0.1075**	**1.292**
-1.20	**0.1151**	**1.256**
-1.16	**0.1230**	**1.221**
-1.12	**0.1314**	**1.186**
-1.08	**0.1401**	**1.151**
-1.04	**0.1492**	**1.117**
-1.00	**0.1587**	**1.083**
-0.96	**0.1685**	**1.050**
-0.92	**0.1788**	**1.017**
-0.88	**0.1894**	**0.984**
-0.84	**0.2005**	**0.952**
-0.80	**0.2119**	**0.920**
-0.76	**0.2236**	**0.889**
-0.72	**0.2358**	**0.858**
-0.68	**0.2483**	**0.828**
-0.64	**0.2611**	**0.798**
-0.60	**0.2743**	**0.769**
-0.56	**0.2877**	**0.740**
-0.52	**0.3015**	**0.712**
-0.48	**0.3156**	**0.684**
-0.44	**0.3300**	**0.657**
-0.40	**0.3446**	**0.630**
-0.36	**0.3594**	**0.605**
-0.32	**0.3745**	**0.579**
-0.28	**0.3897**	**0.554**
-0.24	**0.4052**	**0.530**
-0.20	**0.4207**	**0.507**
-0.16	**0.4364**	**0.484**
-0.12	**0.4522**	**0.462**
-0.08	**0.4681**	**0.440**
-0.04	**0.4840**	**0.419**
0.00	**0.5000**	**0.399**
0.04	**0.5160**	**0.379**
0.08	**0.5319**	**0.360**
0.12	**0.5478**	**0.342**
0.16	**0.5636**	**0.324**
0.20	**0.5793**	**0.307**
0.24	**0.5948**	**0.290**
0.28	**0.6103**	**0.274**
0.32	**0.6255**	**0.259**
0.36	**0.6406**	**0.245**
0.40	**0.6554**	**0.230**
0.44	**0.6700**	**0.217**
0.48	**0.6844**	**0.204**
0.52	**0.6985**	**0.192**
0.56	**0.7123**	**0.180**
0.60	**0.7257**	**0.169**
0.64	**0.7389**	**0.158**
0.68	**0.7517**	**0.148**
0.72	**0.7642**	**0.138**
0.76	**0.7764**	**0.129**
0.80	**0.7881**	**0.120**
0.84	**0.7995**	**0.112**
0.88	**0.8106**	**0.104**
0.92	**0.8212**	**0.097**
0.96	**0.8315**	**0.090**
1.00	**0.8413**	**0.083**
1.04	**0.8508**	**0.077**
1.08	**0.8599**	**0.071**
1.12	**0.8686**	**0.066**
1.16	**0.8770**	**0.061**
1.20	**0.8849**	**0.056**
1.24	**0.8925**	**0.052**
1.28	**0.8997**	**0.047**
1.32	**0.9066**	**0.044**
1.36	**0.9131**	**0.040**
1.40	**0.9192**	**0.037**
1.44	**0.9251**	**0.034**
1.48	**0.9306**	**0.031**
1.52	**0.9357**	**0.028**
1.56	**0.9406**	**0.026**
1.60	**0.9452**	**0.023**
1.64	**0.9495**	**0.021**
1.68	**0.9535**	**0.019**
1.72	**0.9573**	**0.017**
1.76	**0.9608**	**0.016**
1.80	**0.9641**	**0.014**
1.84	**0.9671**	**0.013**
1.88	**0.9699**	**0.012**
1.92	**0.9726**	**0.010**
1.96	**0.9750**	**0.009**
2.00	**0.9772**	**0.008**
2.04	**0.9793**	**0.008**
2.08	**0.9812**	**0.007**
2.12	**0.9830**	**0.006**
2.16	**0.9846**	**0.005**
2.20	**0.9861**	**0.005**
2.24	**0.9875**	**0.004**
2.28	**0.9887**	**0.004**
2.32	**0.9898**	**0.003**
2.36	**0.9909**	**0.003**
2.40	**0.9918**	**0.003**
2.44	**0.9927**	**0.002**
2.48	**0.9934**	**0.002**
2.52	**0.9941**	**0.002**
2.56	**0.9948**	**0.002**
2.60	**0.9953**	**0.001**
2.64	**0.9959**	**0.001**
2.68	**0.9963**	**0.001**
2.72	**0.9967**	**0.001**
2.76	**0.9971**	**0.001**
2.80	**0.9974**	**0.001**
2.84	**0.9977**	**0.001**
2.88	**0.9980**	**0.001**
2.92	**0.9982**	**0.001**
2.96	**0.9985**	**0.000**
3.00	**0.9987**	**0.000**



***Z* & *L(z)* for special service levels**

Service Level F(z)	*z*	*L(z)*
75%	0.67	0.150
90%	1.28	0.047
95%	1.64	0.021
99%	2.33	0.003