The test program

|  |
| --- |
| import java.util.Scanner;  public class SupermarketQueue {      public static void main(String args[]) {          try {              Scanner scanner = new Scanner(System.in);              System.out.println("--------------SETUP SIMULATION ENVIRONMENT---------------");              // Input simulation length              int simmin = -1;              while (simmin == -1) {                  try {                      System.out.print("Input simulation length (min): ");                      String input = scanner.nextLine();                      simmin = checkvaild(input, false); // the input value > 0                  } catch (WrongDataTypeException | NegativeZeroException | OverflowException e) {                      System.out.println(e.getMessage()); // show the exception                  }              }              // Input number of counters              int counter = -1;              while (counter == -1) {                  try {                      System.out.print("Input number of counter: ");                      String input = scanner.nextLine();                      counter = checkvaild(input, false); // the input value > 0                  } catch (WrongDataTypeException | NegativeZeroException | OverflowException e) {                      System.out.println(e.getMessage()); // show the exception                  }              }              System.out.println("\n--------------START SIMULATION --------------");              LinkedQueue waitingQueue = new LinkedQueue(); // create a new waiting queue              int[] teller = new int[counter]; // create a array of teller              int customersServed = 0;              int sum = 0;              int max = 0;              int min = 0;              double totalQueueLength = 0; // Total number of customers in the queue              int maxQueueLength = 0;  // Maximum queue length during the simulation              for (int i = 1; i <= simmin; i++) {                  System.out.println("At the beginning of iteration " + i + " ...");                  // Input serving time for a new customer                  int servingTime = -1;                  while (servingTime == -1) {                      try {                          System.out.print("Input serving time for a new customer: ");                          String input = scanner.nextLine();                          servingTime = checkvaild(input, true); // the input value >= 0                      } catch (WrongDataTypeException | NegativeZeroException | OverflowException e) {                          System.out.println(e.getMessage()); // show the exception                      }                  }                  if (servingTime > 0) {                      sum += servingTime;                      if (max == 0 && min == 0) {//make sure the max, min is 0                          max = servingTime;                          min = servingTime;                      }                      if (servingTime > max) {//check the serving time >  max,                          max = servingTime;//if yes, than update                      }                      if (servingTime < min) {//check the serving time <  min,                          min = servingTime;//if yes, than update                      }                      waitingQueue.enqueue(servingTime);//enqueue the serving time to the waiting queue                  }                  System.out.println("After " + i + " minute ##");                  int currentQueueLength = waitingQueue.size(); // Get the current queue length                  totalQueueLength += currentQueueLength; // Add to the total queue length                  if (currentQueueLength > maxQueueLength) { // Check if it's the maximum queue length                      maxQueueLength = currentQueueLength;                  }                    //for loop the counter                  for (int j = 0; j < counter; j++) {                      //check the queue is not empty and teller is free                      if (!waitingQueue.isEmpty() && teller[j] <= i) {                          int serviceTime = (int) waitingQueue.dequeue();                          teller[j] = i + serviceTime;                          customersServed++;                      }                      System.out.print("Teller\_" + (j + 1) + " [" + teller[j] + "] ");                  }                  System.out.println("Waiting Queue: " + waitingQueue.toString());              }              System.out.println("\n--------------END SIMULATION --------------");              System.out.println("Total minutes simulated: " + simmin + " minutes");              System.out.println("Number of tellers: " + counter);              System.out.println("Number of customers served: " + customersServed + " customers");              System.out.println("\n--------------DATA ANALYSIS --------------");              if (customersServed > 0) {                  System.out.println("Average serving time: " + (sum / customersServed) + " minutes");              } else {                  //make sure the serving time is not 0                  System.out.println("Average serving time: N/A (no customers served)");              }              System.out.println("Maximum serving time: " + max + " minutes");              System.out.println("Minimum serving time: " + min + " minutes");              System.out.println("Total waiting time: " + sum + " minutes");              System.out.println("Average number of customers waiting in the queue: " + (totalQueueLength / simmin));              System.out.println("Maximum queue length during the simulation: " + maxQueueLength );          } catch (QueueFullException e) {              System.out.println(e.getMessage());          } catch (QueueEmptyException e) {              System.out.println(e.getMessage());          } catch (Exception e) {              System.out.println("An unexpected error occurred: " + e.getMessage());          }      }      public static int checkvaild(String input, boolean canzero) throws WrongDataTypeException, NegativeZeroException, OverflowException {          try {              // Check if the input is a valid integer              long number = Long.parseLong(input); // Use long to detect overflow exception              if (number > Integer.MAX\_VALUE) {                  throw new OverflowException("Input number is too large. Please enter a smaller value.");              }              int intNumber = (int) number; // change the long to int              if (canzero) { // Check if the input can be 0 or not                  if (intNumber >= 0) {                      return intNumber; // Valid input                  } else {                      throw new NegativeZeroException("Input number must be >= 0. Please try again.");                  }              } else {                  if (intNumber > 0) {                      return intNumber; // Valid input                  } else {                      throw new NegativeZeroException("Input number must be > 0. Please try again.");                  }              }          } catch (NumberFormatException e) {              // If the input is not a number              throw new WrongDataTypeException("Wrong data type. Please enter a positive integer.");          }      }  } |

This is the program test case:

The normal test case 1 (No exception):

|  |
| --- |
| Input:  10  2  5  5  0  5  4  3  0  0  0  0 |
|  |

Input invalid test case 2 input negative number:

|  |
| --- |
| Input:  5  -1 // the counter can not input the negative number  2  5  5  5  5  5 |
|  |

Input invalid test case 3 input 0:

|  |
| --- |
| Input:  0// the simulation length should not be zero  5  2  3  6  2  0  0 |
|  |

Input invalid test case 4 input wrong data type:

|  |
| --- |
| Input:  10  2  Abc // Wrong data type  -1082143905809238402234234234234 // Wrong data type  +4309534905803944534545 // Wrong data type  [!@#$%^&\*()\_+{}[]<,.>/?`~/](mailto:!@#$%^&*()_+{}[]<,.>/?`~/) // Wrong data type 12312 1231 // Wrong data type  0.10  0  5  5  5  5  5  5  5  5  5 |
|  |

Input invalid test case 5 input wrong data type:

|  |
| --- |
| Input:  2147483648 //the value to large  5  2147483648 //the value to large  2  2147483648 //the value to large  2  2  2  2  2 |
|  |