1 //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
 2 //  
 3 // Name: Derek Gallardo  
 4 // Date: July 26th, 2022  
 5 // Version: 1.0.0  
 6 // Programming Language: Java  
 7 // Java Version: 17  
 8 // Description: This program creates two stacks and a queue to keep track of cars using a parking lot.  
 9 // Each car should be assigned a random decal and only those who have a specific special decal will be permitted to park.  
 10 // Randomly assign a decal and check the decal before attempting to park the car.  
 11 // \*if the user does not have the correct parking decal t park in the lot, the driver should be told nicely that they do not  
 12 // have the correct permit t park in this lot.  
 13 // The parking lot is last-in, first-out stack.  
 14 // The program must first check to see if the car has the correct decal.  
 15 // If it does have the correct permit, the car may be added to the lost as long as there is enough room.  
 16 // To park a car, the car's license tag is added to the parking lot stack.  
 17 // The parking can hold 15 car at time.  
 18 // If the parking lot is full, a waiting care can be added to a waiting queue.  
 19 // When a car leaves the lot, the next car in the queue will be able to park in the parking lot before allowing a new car to be entered.  
 20 // \*\* The parking lot should populate with 10 car when the program launches \*\*  
 21 // One stack representing the parking lot, One stack representing the street.   
 22 // A queue will be used to store the cars waiting for a place to park if the lot is full.  
 23 // \*Use integers a license plate numbers.  
 24 // \*License plates should have 5 digits  
 25 // Positive numbers add a car, negative numbers remove a car, zero stps the simulation.  
 26 // Park - positive number -> Retrieve - Negative Number in front of license number  
 27 // The program should tell me if I have the correct decal and if there is room to park my car.  
 28 // If the stack representing the parking lot is full, my car should be added to the waiting queue.  
 29 // If I remove my car, the next car in the queue should be parked before a new car can be entered unless the queue is empty.  
 30 // The status of the parking lot and the waiting queue should be printed to the console after each operation is complete.  
 31 // The status includes the cars that are parked in the lot as well as the cars that are currently waiting in the queue to park.   
 32 ///   
 33 // The expected output should contain the list of cars currently parked in the parking lost and the list of cars currently  
 34 // waiting to park after each car is entered. The prgraom should also tell the user if they have a correct decal. If they do,  
 35 // The program should tell the driver if their car will be parked in the lot or if it must wait in the waiting queue for a space  
 36 // to become available. If they do not, the program should politely tell the user they do not have the correct decal to park in this lot.  
 37 //   
 38 // Utilize a stack for the parking lot  
 39 // Utilize a stack for the street  
 40 // Utilize a queue to store the cars waiting to park.  
 41 // Generate a random decal type.  
 42 // Correct decal check  
 43 // Limit the number of cars in the lot to max of 15  
 44 // Populate the parking lot with 10 cars to start  
 45 // Output that includes the cars parked in the lot and any  
 46 // cars waiting in the queue to park after each operation.  
 47 // Easy to understand user interface that allows the user to park a car or retrieve a parked car.  
 48 //  
 49 //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
 50   
 51 import java.util.Random;  
 52 import java.util.Scanner;  
 53   
 54 public class Main {  
 55   
 56 public static void main(String[] args)  
 57 {  
 58 Stack parking\_lot = new Stack();  
 59 Stack street = new Stack();  
 60 Queue waiting\_list = new Queue();  
 61   
 62 // Declare some variables  
 63 Scanner scan = new Scanner(System.in);  
 64 Random rand = new Random();  
 65   
 66 // First, populate ten cars  
 67   
 68 while(parking\_lot.size() < 10)  
 69 {  
 70 int license = 5000 + rand.nextInt(5000) % 5000;  
 71 int decal = 5000 + rand.nextInt(5000) % 5000;  
 72 if(decal % 10 < 5)  
 73 {  
 74 Car c = new Car(license, decal);  
 75 parking\_lot.push(c);  
 76 }  
 77 }  
 78   
 79 System.out.println("Parking lot : ");  
 80 System.out.println(parking\_lot);  
 81   
 82 // Start the simulation  
 83   
 84 String input;  
 85 int choice = 1;  
 86   
 87 while(choice != 0)  
 88 {  
 89 System.out.println("Enter choice (0 to stop):");  
 90 input = scan.nextLine();  
 91 choice = Integer.parseInt(input);  
 92   
 93 if(choice == 0)  
 94 System.out.println("Thank you!");  
 95 else if(choice > 0)  
 96 {  
 97 // Assign a random decal  
 98 int decal = 5000 + rand.nextInt(10000) % 5000;  
 99 System.out.println("Decal assigned = " + decal);  
100 if(decal % 10 < 5)  
101 System.out.println("You don't have the permission to park here.");  
102 else  
103 {  
104 Car c = new Car(choice, decal);  
105 if(parking\_lot.isFull())  
106 waiting\_list.enqueue(c);  
107 else  
108 parking\_lot.push(c);  
109 }  
110 }  
111 else  
112 {  
113 // Car trying to exit the system.  
114 int license = -choice;  
115 while(parking\_lot.top().get\_license() != license)  
116 {  
117 Car c = parking\_lot.pop();  
118 street.push(c);  
119 // System.out.println(c.get\_license() + " pushed to the street");  
120 }  
121   
122 if(parking\_lot.isEmpty())  
123 System.out.println("The car does not exist in the parking lot.");  
124 else  
125 parking\_lot.pop();  
126   
127 while(!street.isEmpty())  
128 {  
129 Car c = street.pop();  
130 parking\_lot.push(c);  
131 }  
132   
133 // See if a car can be moved from the queue to the parking lot  
134 if(!waiting\_list.isEmpty() && parking\_lot.size() < 15)  
135 {  
136 Car c = waiting\_list.dequeue();  
137 parking\_lot.push(c);  
138 System.out.println("Car " + c.get\_license() + " dequeued from the Q");  
139 }  
140 }  
141   
142 System.out.println("Parking lot : ");  
143 System.out.println(parking\_lot);  
144 }  
145   
146 System.out.println("Thank you!");  
147   
148 }  
149   
150 }  
151 1   
 2 public class Car  
 3 {  
 4 int license;  
 5 int decal;  
 6   
 7 Car(int \_license, int \_decal)  
 8 {  
 9 license = \_license;  
10 decal = \_decal;  
11 }  
12   
13 // Accessors  
14 public int get\_license()  
15 {  
16 return license;  
17 }  
18   
19 public int get\_decal()  
20 {  
21 return decal;  
22 }  
23   
24 public String toString()  
25 {  
26 return "Car license: " + license + "\tDecal: " + decal;  
27 }  
28 }  
29 1   
 2 public class Queue  
 3 {  
 4 Car[] contents;  
 5 int rear;  
 6   
 7 public Queue()  
 8 {  
 9 contents = new Car[15];  
10 rear = 0; // the index of the location where the next Car will be inserted.  
11 }  
12   
13 public boolean isEmpty()  
14 {  
15 return rear == 0;  
16 }  
17   
18 public void enqueue(Car car)  
19 {  
20 if(rear == 15)  
21 return;  
22 contents[rear++] = car;  
23 }  
24   
25 public Car dequeue()  
26 {  
27 if(rear == 0)  
28 return null;  
29   
30 Car c = contents[0];  
31   
32 for(int i = 0; i < rear - 1; i++)  
33 contents[i] = contents[i + 1];  
34 --rear;  
35   
36 return c;  
37 }  
38   
39 public String toString()  
40 {  
41 String result = "";  
42 for(int i = 0; i < rear; i++)  
43 result = result + (contents[i]) + "\n";  
44 return result;  
45 }  
46 } 1   
 2 public class Stack  
 3 {  
 4 Car[] contents;  
 5 int top;  
 6   
 7 public Stack()  
 8 {  
 9 contents = new Car[15];  
10 top = 0; // the index of the location where the next Car will be inserted.  
11 }  
12   
13 public int size()  
14 {  
15 return top;  
16 }  
17   
18 public boolean isEmpty()  
19 {  
20 return top == 0;  
21 }  
22   
23 public boolean isFull()  
24 {  
25 return top == 15;  
26 }  
27   
28 public Car top()  
29 {  
30 if(isEmpty())  
31 return null;  
32 System.out.println("Top returned " + contents[top - 1].get\_license());  
33 return contents[top - 1];  
34 }  
35   
36 public void push(Car car)  
37 {  
38 contents[top++] = car;  
39 }  
40   
41 public Car pop()  
42 {  
43 if(top == 0)  
44 return null;  
45   
46 Car c = contents[top - 1];  
47 --top;  
48 return c;  
49 }  
50   
51 public String toString()  
52 {  
53 String result = "";  
54 for(int i = top - 1; i >= 0; i--)  
55 result = result + (contents[i]) + "\n";  
56   
57 return result;  
58 }  
59 }