Professor Javaher

COSC 2436

Project 1: Design, Implementation, Problems and References

Program Design:

Overview: The program process ticket purchases for visitors, makes necessary queues for each ride and removes visitors from queues, simulate and complete rides. The program pauses execution for 1-5 seconds randomly to create a real simulation experience of the ride. There are 5 rides available for Visitors and they can choose the ride they want. Confirmation messages are shown after purchase of ticket with ticket type (child,adult or senior)and ride name, after being added to the queue of the ride, and at the start and end of ride. If the ride is not available, a message is shown "Ride not found! Try again." It is super easy for user to enter data and get results.

There are 2 classes in the program: Visitor() and ticketSystem().

Visitor():

- Interacts with user.
- All the input from user is taken here.
- It interacts with ticketSystem() class further to give desired output.
- It contains main method: input from user takes place here. Calling of other methods is here as well.
- After getting visitor's name, age and ride number all this is sent to Visitor.
- Visitor is accessible to all methods being called later (purchaseTicket, joinRideQueue, simulateRides)
- Visitor method takes in information and determines the TicketType based on age.
- Determines ridePreference based on Ride integer (comparing it back to the list displayed to user)
- All methods in ticketSystem is called in main method in order.

ticketSystem():

 Contains all methods to process ticket purchases, joining Ride queues and simulation of rides.

- It has Stack for all people who purchases tickets and Array list for each ride.
- Some confirmation messages are printed after the execution of each method.
- It has 4 methods.
- ticketSystem method is just adding a new linkedlist to each of the ride's queue. So they all can have their own queues.
- purchaseTicket method process purchasing of tickets by adding them to stack ticketBooth.
- joinRideQueue method adds visitors to queue of their Preferred ride.
- simulateRides method removes the visitor from queue and start the ride experience. Then, pause for 1-5 seconds randomly pretending like actual simulation of the ride and then gives the confirmation message that the visitor has completed the ride.

Together these two classes shows the desired output.

Implementation:

Initial idea:

	V 0.1		N N	all a al	9
	Visitor Class	<i>&&</i>	sistem (bu	Told str	
•	name	d			
•	Age	Adult	() Bull Deep	100	*
•	ticketType <	C serve.	Site Control		9
•	nde Preference		Similation	Ride	k.
		J.	Hendfonet	Stack	
west w	was Class visit	ri (comons	3		
0			(and		
		i ticket			
1stole	at an bullic in	sistilit ja			
listof a	eteropulic m		u n	coul to	Visito
118-91	Werin	out name =	u m	send to	Visitor
(5-2	Mila Human used	out nam =	" " —	"	Visitor
118-91	Mila Human used	out nam =	" " —	send to	Visitor
(5-2	sering sering shiput Re	out name = but age = dereference =	" " -)	"	
haises.	Shiput Re Rides = [Roller	out name = put age = dereference = coaster, re	$ \begin{array}{ccc} " & -3 \\ & -3 \\ & -3 \\ \end{array} $ risushed, B	"	
haises.	Shiput Re Rides = [Roller	out name = put age = dereference = coaster, re	$ \begin{array}{ccc} " & -3 \\ & -3 \\ & -3 \\ \end{array} $ risushed, B	"	
haises.	sering sering shiput Re	out name = put age = dereference = coaster, re	$ \begin{array}{ccc} " & -3 \\ & -3 \\ & -3 \\ \end{array} $ risushed, B	"	
دوندها	Rides = [Roller Duck form	out name = put age = dereference coaster, re en Water	" " → risuehed, B	" " " " " " " " " " " " " " " " " " "	ay,
دوندها	Rides = [Roller Duck form	out name = put age = dereference = coaster, Fe en Water	$ \begin{array}{ccc} " & -3 \\ & -3 \\ & -3 \\ \end{array} $ risushed, B	" " " " " " " " " " " " " " " " " " "	ay,
sed could	Rides = [Roller Deugh how tick	put vane = put age = dereference = coaster, Fer en , Water etType =	" " → risweheel, B slide] get from	Visitor	ay,
sed could	Rides = [Roller Deugh how tick	put nane = put age = dereference = coaster, Fer en Water etType ==	riswheel, B vide of from get from use Ticket (Visitor	as,
sed certical	Rides = [Roller Duck how	put nane = put age = dereference = coaster, Fer en Water etType ==	riswheel, B vide of from get from use Ticket (Visitor Suy	as,
Section of Section	Rides = [Roller Deugh how tick	put nane = put age = dereference = coaster, Fer en Water etType ==	" " → risweheel, B slide] get from	Visitor	as,
sed certical	Rides = [Roller Deugh how tick	put nane = put age = dereference = coaster, Fer en Water etType ==	riswheel, B vide of from get from use Ticket (Visitor Suy	as,

-	treket System Class
27.00	N 1 7 6 1
•	purchase licked ()
•	1084 Riche Queul)
•	purchase Tickell() jorn Rice Queul) Ride Similation
	Stack ticketbooth
	a Licket
ianu	Acoustic to author wide
	- Access to Ride Names needed
	Arraylist for every vide Vames needed - Access to Ride Names needed - Access to Ride Preference needed
, bu	Risson (Roller capter, Persionelised Gumbara
	- Adding people/visitors to specific asses Queus
	- Remove after completing mides Simulation with Randonly
	Simulation ruth Kandouls
MON	us . () burney biskister

After this, I had I made the rough program. Just a simple program to have main method and both classes watching the classes and objects lecture videos. Then I worked on the main

method first to create the interface I wanted to have. I got help from Array and Arraylist that I made while studying their presentations to help setup getting input from user and Array list for ride queues. Stacks and Queues video was helpful in designing last part of the program (simulation). It helped me with using random to simulate rides.

Problems:

- The main problem was writing my first program in java. I had to look at other videos
 of Dr. Javaher about creating first java program and how to actually create a project
 and class file (https://youtu.be/shaaiy-A-Wpl?feature=shared)
 (https://youtu.be/8HMaG6IKHcA?feature=shared)
- I had to google how to connect two classes, I was unsure how to access one class in other class. I googled an example to see how multiple classes work in java. I found that they were using this variable name (https://systechgroup.in/blog-java-program-using-two-classes/). I used it in my Visitor class and it was then accessible in other classes as visitor variable name.
- Another problem was a silly one. Once I was done with my code. It was actually
 printing it all in one line. Then I had to find out how to create new lines in java.

 (https://www.geeksforgeeks.org/java-program-to-print-a-new-line-in-string/)
- Another problem was with output as well. It was not asking me about my
 ridePreference and was just running the rest of the program without knowing which
 ride I want. At this time This was not a great issue, some random typo and errors
 made it do that. And when I added new lines it was fixed.
- One of the problems was Time. I had exams for other classes. I started working on this project way ahead of time but still some work was left to be done in the week of submission. Another was my Anxiety, I was so nervous and worried about each step in the program. I think this was just because this was the first project and I was overwhelmed. I will work on both of these for next project.

Other references:

https://youtu.be/hVD79qXNLsM?feature=shared

stacks and queue lecture. The last example has simulation code, I got most help for simulation part in my code from that.

https://youtu.be/fWtAntJ0dpM?feature=shared

helped me with classes and joining classes. How to create instances.