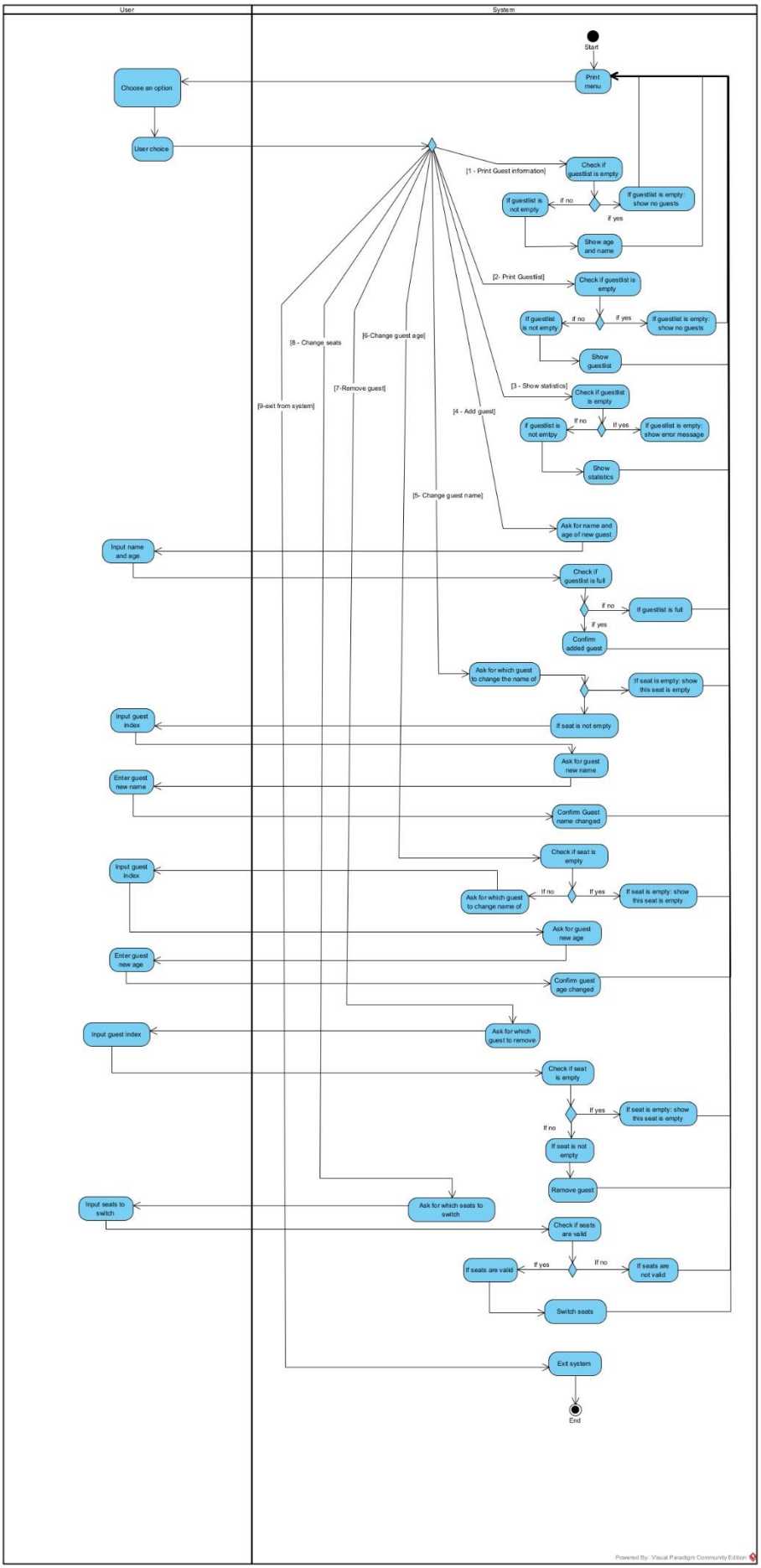
DA339A Rapport U1

Namn: Josef Zakaria Jehia

Dator-id: AQ1637

Datum för inlämning: 2024-10-13


User
System
Start
Print menu
Choose an option
User choice
[1 - Print Guest information]
Check if guestlist is empty
If guestlist is not empty
if no
If guestlist is empty: show no guests
if yes
Show age and name
[2- Print Guestlist]
Check if guestlist is empty
If guestlist is not empty
if no
if yes
If guestlist is empty: show no guests
Show guestlist
[3 - Show statistics]
Check if guestlist is empty
if guestlist is not emtpy
If no
If yes
If guestlist is empty: show error message
Show statistics
[4 - Add guest]
Ask for name and age of new guest
Input name and age
Check if guestlist is full
if no
If guestlist is full
if yes
Confirm added guest
[5- Change guest name]
Ask for which guest to change the name of
If seat is empty: show this seat is empty
Input guest index
If seat is not empty
Ask for guest new name
Enter guest new name
Confirm Guest name changed
[6-Change guest age]
Check if seat is empty
Input guest index
Ask for which guest to change name of
If no
If yes
If seat is empty: show this seat is empty
Ask for guest new age
Enter guest new age
Confirm guest age changed
[7-Remove guest]
Input guest index
Ask for which guest to remove
Check if seat is empty
If yes
If seat is empty: show this seat is empty
If no
If seat is not empty
Remove guest
[8 - Change seats
Input seats to switch
Ask for which seats to switch
Check if seats are valid
If seats are valid
If yes
If no
If seats are not valid
Switch seats
[9-exit from system]
Exit system
End
Powered By: Visual Paradigm Community Edition

Reflektioner och motiveringar

Reflektion över uppgiftens utmaningar

En del i arbetet som jag kände var knepigt att fixa var att hantera de olika funktionerna i gästlistan, särskilt när det gällde att lägga till, ändra och ta bort gäster. Min tanke var att försöka implementera metoder som inte gjorde att koden blev svårhanterlig, vilket jag kände var knepigt. Det var även en utmaning att se till så att alla felmeddelanden visades rätt när en användare skrev in ogiltiga inmatningar.

Att rita aktivitetsdiagrammet var också en utmaning i detta moment. Att få till en exakt rätt representation av alla menyval tog sin tid, jag justerade också diagrammet flera gånger så att det stämde med koden och UML-notationen.

Reflektion över uppgiftens implementation

Koden jag skrivit följer kraven som man ska följa och är uppdelad i metoder för att varje del ska vara så enkelt som möjligt att följa. Detta kan man se på statistiken för gästlistan, där jag lyckades få till rätt beräkning för antalet vuxna och barn genom att hantera åldersgränsen på 13 år. När jag exekverade koden dök det upp några felmeddelanden jag justerade för att få fram korrekt information till användaren, jag löste detta genom att lägga till fler kontroller i de metoder som hanterade inmatning.

Gästlistans storlek hanterade jag på ett dynamiskt sätt för att programmet fortfarande ska funktionera även om storleken på listan ändras. Att implementera byte av platser mellan gäster var en rolig utmaning men fungerade bra när jag väl hade löst logiken.

Motivering av lösning för att läsa input från användare

Jag valde att använda Scanner-klassen för att läsa in data från användaren på grund av att den gör det enkelt att hantera textbaserade inmatningar direkt från terminalen. Detta gör så att användaren direkt kan skriva in namn, ålder och andra val i programmet, vilket gör att programmet känns mer interaktivt. Ett annat sätt är att skapa separata metoder för varje form av inmatning, men detta hade gjort koden mer komplicerad och svårare att följa. Genom att ha inmatningen i main-loopen tycker jag att det lättare att hantera och förstå.

En fördel med denna lösning är att den är mycket flexibel på så sätt att användaren kan direkt påverka programmet utan att behöva starta om eller gå tillbaka till en specifik del av koden. Jag la till felhantering som gör att programmet direkt ger användaren ett meddelande om något inte är rätt, till exempel om de försöker välja ett index som inte finns.

En nackdel med detta är att main-loopen kan bli för lång och detta kan göra det svårare att följa om inmatningsvalen är för många. En annan möjlighet hade varit att dela upp inmatningarna i separata metoder för återanvändning, men jag valde i detta fallet att prioriterade enkelhet. Genom att hantera inmatningen direkt i huvudloopen blev koden mer lättläst och smidig.