**ASANSÖRLERDEKİ TALEP YOĞUNLUĞUNUN MULTITHREAD İLE KONTROLÜ PROJESİ**

Arda Talu

Kocaeli Üniversitesi Bilgisayar Mühendisliği Bölümü

170202037 [ardatalu.98@gmail.com](mailto:ardatalu.98@gmail.com)

**Özet**

Projenin amacı bir AVM’deki asansörlere gelen isteklerdeki yoğunluğu, multithread kullanarak diğer asansörlerle birlikte azaltmaktır.

Threadlerin ms bazında çalışma aralıklarında AVM’ye giriş çıkış kontrol edilir yoğunluğa göre asansörler aktif pasif konuma geçirilir.

**1.Giriş**

Programlama Dili: Proje Java dili kullanılarak gerçekleştirilmiştir.

Kullanılan Araçlar: Proje NetBeans 8.2 sürümü ve Notepad++ araçlarından faydalanılarak yapılmıştır.

**2.Algoritma**

Program başlatıldığında kullanıcıdan seçim yapması istenir.

1 ve 2 arasındaki sayılardan seçim yaptığı sayıya göre program çalıştırılır.

1 seçilirse asansörlerin o anki durumu(kat bilgisi beklemede olup olmadığı)gösterilir.

2 seçilirse çağrılacak kat bilgisi ve hedef kat bilgisi alınır.

Uygulama çalıştırılır ve asansörün kat kat hareketi thread süresine göre adımlarla ekrana yansıtılır.

Hedef kata ulaşıldığında bir sonraki seçim sorulur.

**3.Sözde Kod**

1. Başla
2. YazLab102.java dosyası açılır
3. import java.util.Scanner
4. private static Kontrol kontrol
5. private static Thread kontrolThread;menü.setVisible(true)
6. public static void main
7. Scanner input = new Scanner(System.in)
8. kontrol = Kontrol.getInstance()
9. kontrolThread = new Thread(kontrol)
10. kontrolThread.start()
11. int secim
12. iken(true)
13. Yaz("Seçiminizi giriniz: \n 1. Asansör Durum \n 2. Asansör Çağır")
14. Oku(secim)
15. Eğer(secim == 1)
16. İçin(int i=0;i<5;i++)
17. Asansorler asansor = Kontrol.getInstance().getAsansorList().get(i)
18. System.out.println("Asansör - " + asansor.getId() + " | Anlık kat - " + asansor.getAnlikKat() + " | Durum - " + asansor.getAsansorDurum())
19. Eğer(secim == 1)
20. Yaz("Çağrıldığı katı giriniz (0 - 4 arası): ")
21. Oku istekKat
22. Yaz("Hedef katı giriniz (0 - 4 arası): ")
23. Oku hedefKat
24. AsansorIstek asansorIstek = new AsansorIstek(istekKat, hedefKat)
25. Asansorler asansorler = asansorIstek.submitRequest()
26. GirisCikis.java dosyası açılır
27. enum AsansorDurum {
28. UP,
29. DOWN,
30. STATIONARY,
31. MAINTAINANCE,
32. active}
33. class AsansorIstek
34. private int istekKat
35. private int hedefKat
36. public AsansorIstek(int istekKat, int hedefKat)
37. this.istekKat = istekKat
38. this.hedefKat = hedefKat
39. public int getIstekKat()
40. return istekKat
41. public int getHedefKat()
42. return hedefKat
43. public Asansorler submitRequest()
44. return Kontrol.getInstance().selectElevator(this)
45. public class GirisCikis
46. Asansorler.java dosyası açılır
47. import java.util.Iterator
48. import java.util.LinkedHashMap
49. import java.util.Map
50. import java.util.NavigableSet
51. public class Asansorler implements Runnable
52. private boolean operating
53. private int id
54. private AsansorDurum asansorDurum
55. private int anlikKat
56. private NavigableSet<Integer> floorStops
57. public Map<AsansorDurum, NavigableSet<Integer>> floorStopsMap
58. public Asansorler(int id)
59. this.id = id
60. setOperating(true)
61. public int getId()
62. return id
63. public AsansorDurum getAsansorDurum()
64. return asansorDurum
65. public int getAnlikKat()
66. return anlikKat
67. public void setAsansorDurum(AsansorDurum asansorDurum) {
68. this.asansorDurum = asansorDurum
69. public boolean isOperating()
70. return this.operating
71. public void setOperating(boolean state)
72. this.operating = state
73. Eğer(!state)
74. setAsansorDurum(AsansorDurum.MAINTAINANCE)
75. this.floorStops.clear()
76. Değilse
77. setAsansorDurum(AsansorDurum.STATIONARY)
78. this.floorStopsMap = new LinkedHashMap<AsansorDurum, NavigableSet<Integer>>()
79. Kontrol.updateAsansorList(this)
80. setAnlıkKat(0) public void setAnlıkKat(int AnlıkKat)
81. this.anlikKat = AnlıkKat
82. public void move(){
83. synchronized (Kontrol.getInstance())
84. Iterator<AsansorDurum> iter = floorStopsMap.keySet().iterator()
85. İken(iter.hasNext())
86. asansorDurum = iter.next()
87. floorStops = floorStopsMap.get(asansorDurum)
88. iter.remove()
89. Integer currFlr = null
90. Integer nextFlr = null
91. İken (!floorStops.isEmpty())
92. Eğer (asansorDurum.equals(AsansorDurum.UP))
93. currFlr = floorStops.pollFirst()
94. nextFlr = floorStops.higher(currFlr);
95. try
96. Thread.sleep(500)
97. catch (InterruptedException e)
98. e.printStackTrace()
99. Değilse eğer (asansorDurum.equals(AsansorDurum.DOWN))
100. currFlr = floorStops.pollLast()
101. nextFlr = floorStops.lower(currFlr)
102. Değilse
103. setAnlıkKat(currFlr)
104. Eğer (nextFlr != null) {
105. generateIntermediateFloors(currFlr, nextFlr)
106. Değilse
107. setAsansorDurum(AsansorDurum.STATIONARY)
108. Kontrol.updateAsansorList(this)
109. Yaz("Asansor ID " + this.id + " | Anlık kat - " + getAnlikKat() + " | Sonraki hareket - " + getAsansorDurum())
110. try
111. Thread.sleep(1000)
112. catch (InterruptedException e)
113. e.printStackTrace()
114. try
115. Kontrol.getInstance().wait()
116. catch (InterruptedException e)
117. e.printStackTrace()
118. private void generateIntermediateFloors(int initial, int target)
119. Eğer(initial==target)
120. return
121. Eğer(Math.abs(initial-target) == 1)
122. return
123. int n = 1
124. Eğer(target-initial<0)
125. n = -1
126. İken(initial!=target)
127. initial += n
128. Eğer(!floorStops.contains(initial))
129. floorStops.add(initial)
130. public void run()
131. İken(true)
132. Eğer(isOperating())
133. move()
134. try {
135. Thread.sleep(200)
136. catch (InterruptedException e)
137. e.printStackTrace()
138. else
139. break
140. Kontrol.java dosyası açılır
141. import java.util.\*
142. import java.util.concurrent.ConcurrentSkipListSet
143. public class Kontrol implements Runnable
144. private boolean stopController
145. private static Map<Integer, Asansorler> upMovingMap = new HashMap<Integer, Asansorler>()
146. private static Map<Integer, Asansorler> downMovingMap = new HashMap<Integer, Asansorler>()
147. private static List<Asansorler> asansorList = new ArrayList<Asansorler>(16)
148. private static final Kontrol instance = new Kontrol()
149. private Kontrol()
150. Eğer(instance != null)
151. throw new IllegalStateException("Already instantiated")
152. setStopController(false)
153. initializeElevators()
154. public static Kontrol getInstance()
155. return instance
156. public synchronized Asansorler selectElevator(AsansorIstek asansorIstek)
157. Asansorler asansor = null
158. AsansorDurum asansorDurum=getRequestedElevatorDirection(asansorIstek)
159. int istekKat = asansorIstek.getIstekKat()
160. int hedefKat = asansorIstek.getHedefKat()
161. asansor=findElevator(asansorDurum, istekKat, hedefKat)
162. notifyAll()
163. return asansör
164. private static void initializeElevators(){
165. İçin(int i=0; i<5; i++)
166. Asansorler asansor = new Asansorler(i)
167. Thread t = new Thread(asansor)
168. t.start()
169. private static AsansorDurum getRequestedElevatorDirection(AsansorIstek asansorIstek)
170. AsansorDurum asansorDurum = null
171. int istekKat = asansorIstek.getIstekKat()
172. int hedefKat = asansorIstek.getHedefKat()
173. Eğer(hedefKat>=0 && hedefKat<5 && istekKat>=0 && istekKat<5)
174. Eğer(hedefKat - istekKat > 0)
175. asansorDurum = AsansorDurum.UP
176. Değilse
177. asansorDurum = AsansorDurum.DOWN
178. return asansorDurum
179. asansorList.add(asansor)
180. private static Asansorler findElevator(AsansorDurum asansorDurum, int istekKat, int hedefKat)
181. Asansorler asansor = null
182. TreeMap<Integer, Integer> sortedKeyMap = new TreeMap<Integer, Integer>()
183. Eğer(asansorDurum.equals(AsansorDurum.UP))
184. İçin(Map.Entry<Integer, Asansorler> elvMap : upMovingMap.entrySet())
185. Asansorler elv = elvMap.getValue();
186. Integer distance = istekKat - elv.getAnlikKat()
187. Eğer(distance < 0 && elv.getAsansorDurum().equals(AsansorDurum.UP))
188. continue
189. Değilse
190. sortedKeyMap.put(Math.abs(distance), elv.getId())
191. Integer selectedElevatorId = sortedKeyMap.firstEntry().getValue()
192. asansor = upMovingMap.get(selectedElevatorId)
193. Değilse eğer(asansorDurum.equals(AsansorDurum.DOWN))
194. İçin(Map.Entry<Integer, Asansorler> elvMap : downMovingMap.entrySet())
195. Asansorler elv = elvMap.getValue()
196. Integer distance = elv.getAnlikKat() - istekKat
197. Eğer(distance < 0 && elv.getAsansorDurum().equals(AsansorDurum.DOWN))
198. continue
199. Değilse
200. sortedKeyMap.put(Math.abs(distance), elv.getId())
201. Integer selectedElevatorId = sortedKeyMap.firstEntry().getValue()
202. asansor = downMovingMap.get(selectedElevatorId)
203. AsansorIstek newRequest = new AsansorIstek(asansor.getAnlikKat(), istekKat)
204. AsansorDurum elevatorDirection = getRequestedElevatorDirection(newRequest)
205. AsansorIstek newRequest2 = new AsansorIstek(istekKat, hedefKat)
206. AsansorDurum elevatorDirection2 = getRequestedElevatorDirection(newRequest2)
207. NavigableSet<Integer> floorSet = asansor.floorStopsMap.get(elevatorDirection)
208. Eğer (floorSet == null)
209. floorSet = new ConcurrentSkipListSet<Integer>()
210. floorSet.add(asansor.getAnlikKat())
211. floorSet.add(istekKat);
212. asansor.floorStopsMap.put(elevatorDirection, floorSet)
213. NavigableSet<Integer> floorSet2 = asansor.floorStopsMap.get(elevatorDirection2)
214. Eğer (floorSet2 == null)
215. floorSet2 = new ConcurrentSkipListSet<Integer>()
216. floorSet2.add(istekKat)
217. floorSet2.add(hedefKat)
218. asansor.floorStopsMap.put(elevatorDirection2, floorSet2)
219. return asansör
220. public static synchronized void updateAsansorList(Asansorler asansor)
221. Eğer(asansor.getAsansorDurum().equals(AsansorDurum.UP))
222. upMovingMap.put(asansor.getId(), asansor)
223. downMovingMap.remove(asansor.getId())
224. Değilse Eğer (asansor.getAsansorDurum().equals(AsansorDurum.DOWN))
225. downMovingMap.put(asansor.getId(), asansor)
226. upMovingMap.remove(asansor.getId())
227. Değilse Eğer (asansor.getAsansorDurum().equals(AsansorDurum.STATIONARY))
228. upMovingMap.put(asansor.getId(), asansor)
229. downMovingMap.put(asansor.getId(),asansor)
230. Değilse Eğer (asansor.getAsansorDurum().equals(AsansorDurum.MAINTAINANCE)){
231. upMovingMap.remove(asansor.getId())
232. downMovingMap.remove(asansor.getId())
233. public void run()
234. stopController = false
235. İken(true)
236. try
237. Thread.sleep(200)
238. Eğer(stopController)
239. break
240. catch (InterruptedException e)
241. Yaz(e.getStackTrace())
242. public void setStopController(boolean stop){
243. this.stopController = stop
244. public synchronized List<Asansorler> getAsansorList()
245. return asansorList;
246. public boolean isStopController()
247. return stopController
248. Bitiş

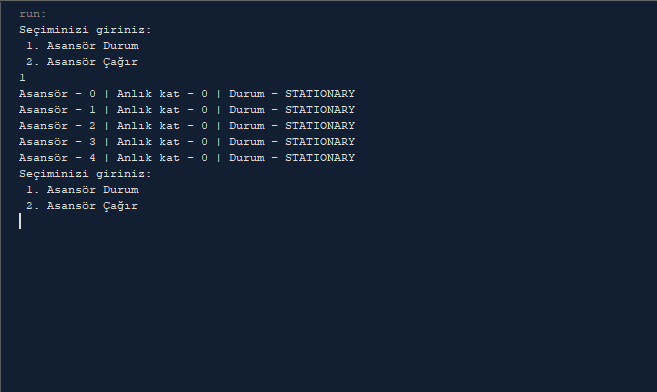
**4.Sonuç**

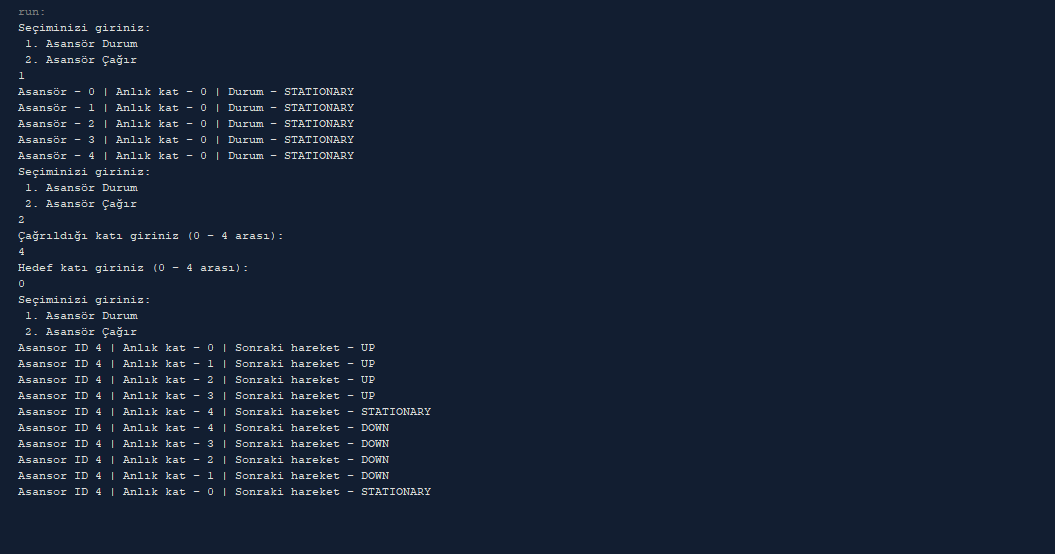
Kullanıcıya yaptığı seçimlere göre asansör bilgisi ve asansör hareketi komutlarını gerçekleyebilen bir program yazılmıştır.

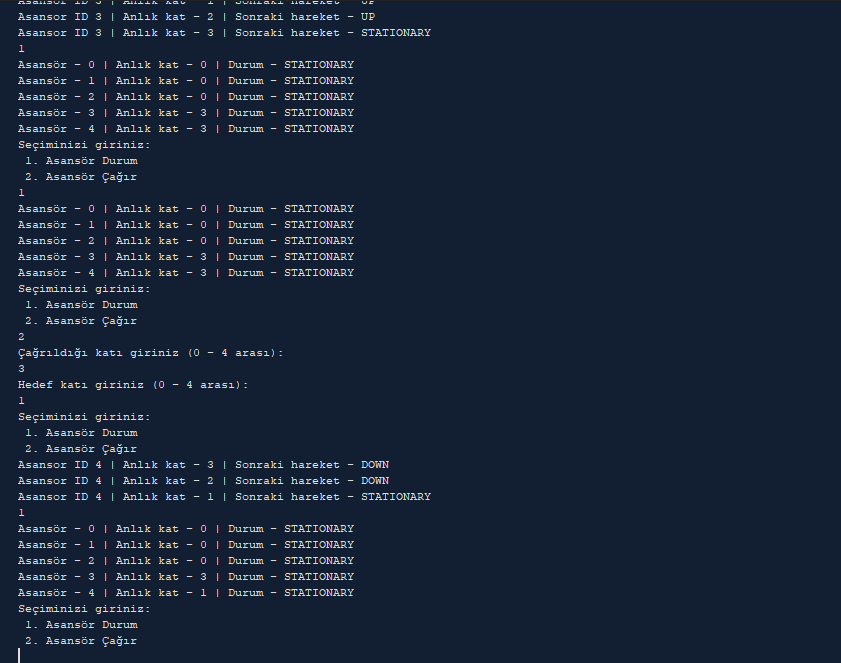
**5. Karşılaşılan Problemler**

Program yazılırken destek alınan kaynakların da karmaşıklığının etkisiyle bazı fonksiyonların uygulanması başarılamamıştır.

**6.Ekran Görüntüleri**







**7. Kaynakça**

[1]<https://www.geeksforgeeks.org/multithreading-in-java/>

[2]<https://www.javatpoint.com/multithreading-in-java>

[3] <https://www.geeksforgeeks.org/java-thread-priority-multithreading/?ref=lbp>

[4]<https://www.tutorialspoint.com/java/java_multithreading.htm>

[5]<https://github.com/pharshal/MultithreadedElevator/tree/master/src>

[6]<https://github.com/shirleyqt/ElevatorSimulation>