Ujian Akhir Semester

Algoritma dan Struktur Data 2

Tahun Akademik 2017/2018

1. Program java sederhana dengan menggunakan metode Linked list Stack dan Queue
2. **package** pertemuan3;
4. **import** java.util.Scanner;
6. **public** **class** LinkedList {
8. LinkedListNode head;
9. LinkedListNode tail;
11. LinkedList() {
12. **this**.head = **null**;
13. **this**.tail = **null**;
14. }
16. **void** print() {
17. LinkedListNode current = **this**.head;
18. **while** (current != **null**) {
19. System.out.print(current.data + " ");
20. current = current.next;
21. }
22. System.out.println("");
23. }
25. **void** push(LinkedListNode new\_node) {
26. **if** (**this**.head == **null** && **this**.tail == **null**) {
27. **this**.head = new\_node;
28. **this**.tail = new\_node;
29. } **else** {
30. tail.next = new\_node;
31. new\_node.prev = tail;
32. **this**.tail = new\_node;
34. }
35. }
37. **void** insert(LinkedListNode new\_node) {
38. **if** (**this**.head == **null**) {
39. **this**.head = new\_node;
40. **this**.tail = new\_node;
41. } **else** **if** (new\_node.data <= **this**.head.data) {
42. **this**.head.set\_prev(new\_node);
43. **this**.head = new\_node;
45. } **else** **if** (new\_node.data >= **this**.tail.data) {
46. **this**.tail.set\_next(new\_node);
47. **this**.tail = new\_node;
49. } **else** {
50. LinkedListNode position = head;
51. **while** (position.data < new\_node.data) {
52. position = position.next;
54. }
55. LinkedListNode previous\_position = position.prev;
56. new\_node.set\_prev(previous\_position);
57. new\_node.set\_next(position);
59. }
60. }
62. LinkedListNode find\_node\_by\_data(**int** data) {
63. LinkedListNode current = **this**.head;
64. **while** (current != **null**) {
65. **if** (current.data == data) {
66. **return** current;
67. }
68. current = current.next;
69. }
70. **return** **null**;
71. }
73. LinkedListNode lastNode(LinkedListNode node) {
74. **while** (node.next != **null**) {
75. node = node.next;
76. }
77. //        System.out.println("last : " + node.data);
78. **return** node;
79. }
81. **void** delete(LinkedListNode deleted) {
82. **if** (deleted != **null** && **this**.head != **null**) {
83. **if** (**this**.head == **this**.tail && deleted == **this**.head) {
84. **this**.head = **null**;
85. **this**.tail = **null**;
86. } **else** **if** (deleted == **this**.head) {
87. LinkedListNode new\_head = **this**.head.next;
88. **this**.head.set\_next(**null**);
89. new\_head.set\_prev(**null**);
90. **this**.head = new\_head;
91. } **else** **if** (deleted == **this**.tail) {
92. LinkedListNode new\_tail = **this**.tail.prev;
93. **this**.tail.set\_prev(**null**);
94. new\_tail.set\_next(**null**);
95. **this**.tail = new\_tail;
96. } **else** {
97. LinkedListNode deleted\_prev = deleted.prev;
98. LinkedListNode deleted\_next = deleted.next;
99. deleted.set\_prev(**null**);
100. deleted.set\_next(**null**);
101. deleted\_prev.set\_next(deleted\_next);
102. }
103. }
105. }
107. //queue
108. //queue first in first out
109. LinkedListNode qpop() {
110. LinkedListNode taken = **null**;
111. **if** (**this**.head == **null** && **this**.tail == **null**) {
112. taken = **null**;
113. } **else** **if** (**this**.head == **this**.tail) {
114. taken = head;
115. **this**.head = **null**;
116. **this**.tail = **null**;
117. } **else** {
118. taken = head;
119. head = head.next;
121. }
122. **return** taken;
123. }
125. // stack = last in first out
126. LinkedListNode spop() {
127. LinkedListNode taken = **null**;
128. **if** (**this**.head == **null** && **this**.tail == **null**) {
129. taken = **null**;
130. } **else** **if** (**this**.head == **this**.tail) {
131. taken = tail;
132. **this**.head = **null**;
133. **this**.tail = **null**;
134. } **else** {
135. taken = tail;
136. **this**.tail.prev.set\_next(**null**);
137. //tail.prev.next = null;
138. **this**.tail = tail.prev;
139. }
140. **return** taken;
141. }
143. //quick sort
144. //mengecek nilai pertama dan terakhir
145. **public** **void** quickSort(LinkedListNode node) {
146. LinkedListNode last = lastNode(node);
148. \_quickSort(node, last);
149. }
151. **void** \_quickSort(LinkedListNode l, LinkedListNode h) {
152. **if** (h != **null** && l != h && l != h.next) {
153. LinkedListNode temp = partition(l, h);
154. \_quickSort(l, temp.prev);
155. \_quickSort(temp.next, h);
156. }
157. }
159. LinkedListNode partition(LinkedListNode l, LinkedListNode h) {
160. **int** x = h.data;
161. LinkedListNode i = l.prev;
163. **for** (LinkedListNode j = l; j != h; j = j.next) {
164. **if** (j.data <= x) {
165. i = (i == **null**) ? l : i.next;
166. **int** temp = i.data;
167. i.data = j.data;
168. j.data = temp;
169. }
170. }
171. i = (i == **null**) ? l : i.next;
172. **int** temp = i.data;
173. i.data = h.data;
174. h.data = temp;
175. **return** i;
176. }
178. **public** **static** **void** main(String[] args) {
179. Scanner sc = **new** Scanner(System.in);
180. LinkedList a = **new** LinkedList();
181. **int** data = 0, bdata = 0, pilih;
182. **char** ulang, ulangpr;
183. ulang = 'y';
184. ulangpr = 'y';
186. **do** {
187. System.out.println("=================== MENU ===================");
188. System.out.println("1. Masukkan Antrian ");
189. System.out.println("2. Tampilkan Antrian ");
190. System.out.println("3. Keluarkan Antrian dengan Stack ");
191. System.out.println("4. Keluarkan Antrian dengan Queue");
192. System.out.println("============================================");
193. **do** {
194. System.out.print("Pilih proses\t: ");
195. pilih = sc.nextInt();
196. } **while** (pilih < 1 || pilih > 4);
197. **switch** (pilih) {
198. **case** 1:
199. **do** {
200. System.out.print("Masukkan banyak data\t: ");
201. bdata = sc.nextInt();
202. sc.nextLine();
203. System.out.println("-------------------------------");
204. **for** (**int** i = 1; i <= bdata; i++) {
205. System.out.print("Masukkan Data ke " + i + "\t: ");
206. data = sc.nextInt();
207. a.push(**new** LinkedListNode(data));
208. }
209. **do** {
210. System.out.print("Tambah antrian lagi? (Y / T)\t");
211. ulangpr = sc.next().charAt(0);
212. } **while** (ulangpr != 't' && ulangpr != 'y');
213. } **while** (ulangpr == 'y');
214. **do** {
215. System.out.print("Kembali ke menu awal ? (Y / T)\t");
216. ulang = sc.next().charAt(0);
217. } **while** (ulang != 'y' && ulang != 't');
219. **break**;
221. **case** 2:
222. System.out.println("============= Data Antrian =============");
223. a.print();
224. **do** {
225. System.out.print("Kembali ke menu awal ? (Y / T)");
226. ulang = sc.next().charAt(0);
227. } **while** (ulang != 'y' && ulang != 't');
229. **break**;
231. **case** 3:
232. **if** (a.head != **null** && a.tail != **null**) {
233. System.out.println("Antrian yang dihapus adalah : " + a.spop().data);
234. } **else** {
235. System.out.println("Tidak dapat menghapus data. Antrian Kosong !!");
236. }
237. a.print();
238. **do** {
239. System.out.print("Kembali ke menu awal ? (Y / T)");
240. ulang = sc.next().charAt(0);
241. } **while** (ulang != 'y' && ulang != 't');
243. **break**;
245. **case** 4:
246. **if** (a.head != **null** && a.tail != **null**) {
247. System.out.println("Antrian yang dihapus adalah : " + a.qpop().data);
248. } **else** {
249. System.out.println("Tidak dapat menghapus data. Antrian Kosong !!");
250. }
251. a.print();
252. **do** {
253. System.out.print("Kembali ke menu awal ? (Y / T)");
254. ulang = sc.next().charAt(0);
255. } **while** (ulang != 'y' && ulang != 't');
256. **break**;
257. }
258. } **while** (ulang == 'y' || ulang == 'Y');
259. }
260. }
262. **package** pertemuan3;
264. **public** **class** LinkedListNode {
266. LinkedListNode next;
267. LinkedListNode prev;
268. **int** data;
270. LinkedListNode(**int** new\_data) {
271. data = new\_data;
272. **this**.next = **null**;
273. **this**.prev = **null**;
274. }
276. **void** set\_prev(LinkedListNode other) {
277. **this**.prev = other;
278. **if** (other != **null**) {
279. other.next = **this**;
280. }
281. }
283. **void** set\_next(LinkedListNode other) {
284. **this**.next = other;
285. **if** (other != **null**) {
286. other.prev = **this**;
287. }
288. }
290. }
291. Pembahasan program
     1. Jelaskan tujuan utama dari program

Tujuan dari program ini adalah untuk memahami letak perbedaan antara metode stack dengan metode queue, dengan membuat sebuah antrian dimana data antrian dimasukkan dengan metode linkedlist kemudian user harus memasukan (push) data, lalu memilih dikeluarkan (pop) secara stack atau queue.

* 1. Jelaskan apa saja input dan tipe data dari inputan program tersebut

Pada saat program berjalan, user akan diminta untuk menginputkan proses apa yang akan dilakukan.

* + 1. Menu 1, digunakan untuk menginputkan banyak data yang akan dimasukkan kedalam antrian dan juga mengisikan data sebanyak banyak data yang diinputkan.
    2. Menu 2 digunakan oleh user untuk menampilkan antrian.
    3. Pada menu 3 : yaitu pilihan ”Keluarkan Antrian dengan Stack” maka data yang telah diinputkan pada menu (1) akan dikeluarkan dengan metode stack.

Contoh : 1, 2, 3

Stack : 1, 2

* + 1. Pada menu 4 : yaitu pilihan “Keluarkan Antrian dengan Queue” maka data yang telah diinputkan pada menu (1) akan dikeluarkan dengan metode queue.

Contoh : 1, 2

Queue : 2

* 1. Jelaskan implementasi dan teori yang anda pilih pada program
     + - Linked List

Penerapan linkedlist dalam program ini dapat dilihat seperti source code dibawah ini.

1. LinkedListNode next;
2. LinkedListNode prev;
3. **int** data;
5. LinkedListNode(**int** new\_data) {
6. data = new\_data;
7. **this**.next = **null**;
8. **this**.prev = **null**;
9. }
11. **void** set\_prev(LinkedListNode other) {
12. **this**.prev = other;
13. **if** (other != **null**) {
14. other.next = **this**;
15. }
16. }
18. **void** set\_next(LinkedListNode other) {
19. **this**.next = other;
20. **if** (other != **null**) {
21. other.prev = **this**;
22. }

Memasukkan data kedalam linkedlist

1. **void** push(LinkedListNode new\_node) {
2. **if** (**this**.head == **null** && **this**.tail == **null**) {
3. **this**.head = new\_node;
4. **this**.tail = new\_node;
5. } **else** {
6. tail.next = new\_node;
7. new\_node.prev = tail;
8. **this**.tail = new\_node;
10. }
11. }

Mengeluarkan data dari linkedlist dengan metode queue

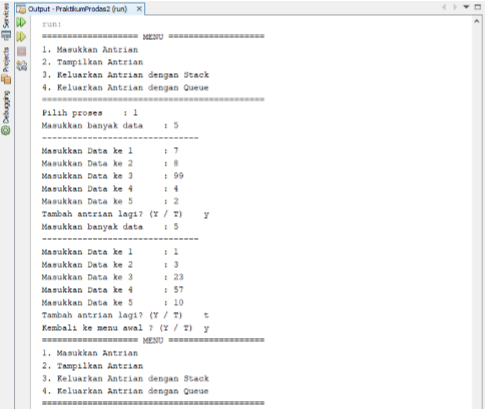
1. LinkedListNode qpop() {
2. LinkedListNode taken = **null**;
3. **if** (**this**.head == **null** && **this**.tail == **null**) {
4. taken = **null**;
5. } **else** **if** (**this**.head == **this**.tail) {
6. taken = head;
7. **this**.head = **null**;
8. **this**.tail = **null**;
9. } **else** {
10. taken = head;
11. head = head.next;
12. }
13. **return** taken;
14. }

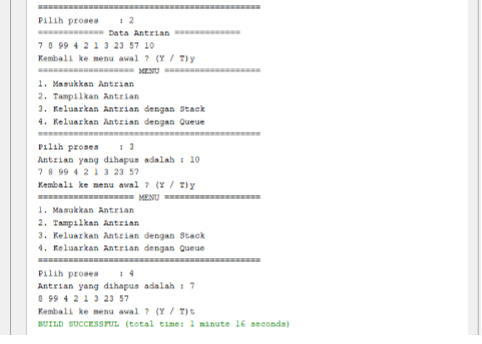
Mengeluarkan data dari linkedlist dengan metode stack

1. LinkedListNode spop() {
2. LinkedListNode taken = **null**;
3. **if** (**this**.head == **null** && **this**.tail == **null**) {
4. taken = **null**;
5. } **else** **if** (**this**.head == **this**.tail) {
6. taken = tail;
7. **this**.head = **null**;
8. **this**.tail = **null**;
9. } **else** {
10. taken = tail;
11. **this**.tail.prev.set\_next(**null**);
12. //tail.prev.next = null;
13. **this**.tail = tail.prev;
14. }
15. **return** taken;
16. }
    * + - Queue sebenarnya adalah sebuah konsep “first in first out”, yaitu data yang masuk terlebih dahulu dan dikeluarkan paling awal sedangkan,
        - Stack adalah sebuah konsep “last in first out”, yaitu data yang masuk terakhir akan dikeluarkan paling awal begitu pun sebaliknya dan
        - Linkedlist dalam program ini adalah sebuah wadah yang digunakan untuk menampung metode queue dan stack.
    1. Gambarkan flowchart dari system / aplikasi anda!



* 1. Apa bentuk output utama dari program?





Tentunya output utama dari sebuah program memiliki keterkaitan yang erat dengan inputan pada program itu sendiri. Selain itu output dan struktur jalannya program telah diperjelas melalui melalui flowchart yang ada. Program ini memperlihatkan bagaimana proses pengambilan atau pengeluaran data dilakukan dengan menggunakan metode stack ataupun queue.

Pada tampilan awal terdapat output atau interface yang menampilkan empat pilihan yang tersedia. Menu tersebut menampilkan hal apa saja yang dapat kita lakukan didalam program sederhana ini.

* 1. Menu pertama berfungsi untuk memasukan data kedalam antrian, data yang diisikan bergantung pada jumlah data yang diinputkan oleh user.
  2. Menu yang kedua berfungsi untuk menampilkan isi dari antrian. Akan ditampilkan apakah didalam antrian telah terdapat data ataupun tidak.
  3. Menu ketiga ialah proses pengeluaran data atau pemanggilan nomor antrian dengan menggunakan metode stack. Program akan menampilkan tampilan “Antrian kosong” apabila tidak terdapat data yang dapat di pop. Dan sebaliknya apabila terdapat data maka program akan mengeluarkan data yang paling terakhir untuk dikeluarkan, sesuai dengan konsep stack sendiri yaitu *last-in-first-out.*
  4. Sama hal nya dengan menu nomor ketiga, menu nomor empat juga akan memeriksa terlebih dahulu isi dari dalam antrian, yang apabila terdapat data maka program akan mengeluarkan data yang pertama masuk namun apabila tidak terdapat data maka program akan menampilkan tampilan “Antrian kosong”. Menu ke empat meggunakan metode queue, yang memiliki konsep *first-in-first-out.*