הסבר על כל class

Mergable heap:

היא המחלקה הראשית בפעולה שלי שבה אני:

קורא את המידע מהקובץ טקסט, שואל את המשתמש שאלות לגבי העדפת הממשק.

השתמשתי במחלקה StringBuilder של java בשביל לקרוא את הטקסט מהקובץ.

אני שואל את המשתמש איך הוא רוצה לגשת לרשימות בצורה ממויינות או לא ממויינת ולפי מה שהוא עונה אני שולח את המידע שמאוכסן ב StringBuilder לאחת מהפעולות הבאות

יצרתי 2 פעולות stringToListUnsorted ו stringToListSorted  
חשוב לציין שפעולות אלו יוצרות רשימה מקושרת מהקובץ טקסט על ידי הפעולה insert כלומר אם אני ב stringToListUnsorted אני שולח איבר איבר ל insert של unsorted ולהפך.

אחר כך אני קורא ל- getFunc שהיא שואלת את המשתמש איזו פעולה הוא רוצה ליישם על הרשימות:  
MakeHeap, Insert, Minimum, ExtractMin, Union.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Node:

במחלקה זו הגדרתי את כל התכונות שאני רוצה שיהיה ל node ול- linked list  
אני לא רואה צורך לפרט פה כי הפעולות לא מסובכות ויש פירוט בקוד.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Sorted:

makeHeap: יוצרת ערימה ריקה חדשה ומחזירה אותה.

Insert: מקבלת רשימה וערך שצריך להוסיף אליה, יוצרת מצביע לראש הרשימה ותחילה עוברת על מקרי הקצה:

1. האם הרשימה ריקה אם כן פשוט הכנס את האיבר לרשימה הגדר אותו להיות הראש והזנב שלה וצא.
2. האם האיבר שאני רוצה להכניס קטן מראש הרשימה (שכן הרשימה ממויינת) אם כן הגדר את המצביע של האיבר שאני מכניס לראש הרשימה הישנה והגדר את האיבר שאני מכניס כראש הרשימה
3. מקרה זה מטופל בסוף הקוד וקובע שאם האיבר שאנו רוצים להוסיף גדול מהאיבר האחרון ברשימה אז צריך להגדיר את האיבר שאנו רוצים להוסיף כ tail וכמובן כאיבר האחרון ברשימה

אחרת פשוט יש לחפש את המקום של האיבר שאנו רוצים להוסיף בצורה ליניארית ולהוסיפו לרשימה למקום המתאים.

Minimum:  
מחזירה את האיבר הראשון ברשימה  
ExtractMin:

מחזירה את האיבר הראשון ברשימה וגם מנתקת אותו מהרשימה

Union:  
מקבלת 2 רשימות ומאחדת את שתי הרשימות לתוך הרשימה הראשונה תוך כדי שמירה על כך שהרשימה הראשונה תיהיה ממויינת בסדר עולה.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Unsorted:

makeHeap: יוצרת ערימה ריקה חדשה ומחזירה אותה.

Insert: מקבלת רשימה וערך שצריך להוסיף אליה ומכניסה את האיבר שאנו רוצים להוסיף לראש הרשימה כלומר הtail נשאר קבוע והראש משתנה.

Minimum:  
מחזירה את האיבר הכי קטן הרשימה לא ממויינת עוברים על הרשימה הלא ממויינת בצורה ליניארית.  
ExtractMin:

משתמשת ב Minimum בשביל לדעת איזה ערך להוציא ואז עוברת על הרשימה ברגע שנתקלת בערך שהמינימלי מנתקת אותו מהרשימה.

Union:  
מקבלת 2 רשימות ומאחרת את הרשימה השניה אל הראשונה באופן הבא הגדרה של של המצביע של הזנב של הרשימה הראשונה לראש של הרשימה השנייה.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Input:  
קובץ טקסט בעל 2 שורות כל שורה מורכבת ממספרים ורווחים וכל שורה מייצגת רשימה

\*\*חשוב במידה ויש צורך לשנות את הרשימות נא לדאוג שהאיבר האחרון ברשימה יהיה מספר ולא רווח אחרת הקוד לא יצליח להפוך את הקלט לרשימה.

לפי ההסבר שעשיתי לעיל:  
ניתוח סיבוכיות הוא:

|  |  |  |
| --- | --- | --- |
|  | unSortedLists | SortedLists |
| makeHeap | O(1) | O(1) |
| Insert | O(1) | O(n) |
| Minimum | O(n) | O(1) |
| ExtractMin | O(n) | O(1) |
| Union | O(1) | O(n+m) \*\* |

\*\*אורך רשימה 1 הוא: n ואורך רשימה 2: הוא m.

צילומי מסך על ההרצות

A screenshot of a computer

Description automatically generated

A computer screen with white text

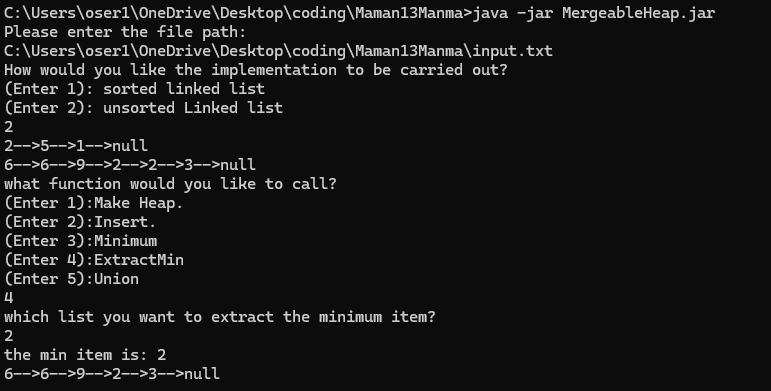
Description automatically generated

A computer screen shot of a black screen

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated