1. Node
   1. שדות סטטיים:
      1. listFactory – המתודה createList מקבלת את הפרמטר parent ומחזירה את הרשימה הריקה החדשה.
   2. שדות:
      1. Key
      2. Mark – סימון בוליאני לcuts
      3. Next
      4. Prev
      5. Children – רשימה מקושרת עם הילדים של הצומת
      6. Siblings – הרשימה המקושרת שהצומת שייך לה (siblings == heap iff node is a root)
   3. בנאי:
      1. מקבל את הkey מהלקוח ושומר אותו בשדה key.
      2. מאתחל רשימה מקושרת של הילדים בעזרת המפעל (children.parent == this)
   4. מתודות get:
      1. getParent
      2. getKey
      3. getSize
      4. rank
      5. getMark
   5. מתודות set:
      1. setPrev(node) – מטפל בprev הנוכחי של הצומת (מנתק את הקשר), ומעדכן את המצביע של this.prev להיות node. מדובר במתודה private, בשביל לבצע השמה של prev צריך להשתמש בinsertPrev.
      2. setNext(node) – כנ"ל כמו setPrev()
      3. setMark – עדיין לא מושלם, צריך מצביע לערימה
   6. מתודות Has:
      1. hasPrev
      2. hasNext
      3. hasParent
   7. מתודות insert:
      1. insertPrev(node) – מוסיף node לפני הצומת ומעדכן את המצביעים של שני הצמתים הרלוונטיים
      2. insertNext(node)...
      3. insertChild(node) – מוסיף את node ראשון לרשימת הילדים
      4. plantNext(list) – שותל את הרשימה אחרי עצמי
      5. plantPrev(list) – שותל לפני עצמי.
   8. מתודות שליפה ומחיקה:
      1. eject() – מוציא את הצומת מתוך העץ יחד עם ילדיו ומחזיר את הצומת לאחר ההוצאה: siblings, next, prev == null.
      2. rejectChildren() – מנתק את הילדים הנוכחיים של הצומת ומגדיר לעצמי רשימת ילדים חדשה וריקה מהמפעל. מחזיר את רשימת הילדים שנותקה.
      3. plantUp() – מנתק את הילדים, מוחק את עצמי מהרשימה, שותל את הילדים לפני הצומת שהיתה הצומת שאחריי, מוריד את הגודל של רשימת האחים שלי ב-1, ואני נמחק ולא מוחזר.
2. LinkedList
   1. שדות סטטיים: אין
   2. שדות:
      1. Root
      2. Tail – האיבר האחרון
      3. Length
      4. Size
      5. MinNode
      6. Parent
   3. בנאי: אין
   4. מתודות get:
      1. getMin()
   5. מתודות set:
      1. setSize(size) - פרטית ולא לשימוש ישירות
      2. increaseSize(delta) – קוראת לsetSize וקוראת לparent.siblings.updateSize().
      3. decreaseSize(delta) – כמו increaseSize, קורא לincreaseSize אם delta שלילי.
   6. מתודות has:
      1. isEmpty
      2. hasParent – false iff the list is a list of heap roots.
   7. מתודות insert:
      1. insertFirst(node) – קוראת לinsertPrev של root, מעדכנת את root, מעדכנת את tail אם צריך, מעדכנת את אורך הרשימה, קוראת לincreaseSize ומעדכנת את המצביע לminNode אם צריך.
      2. Annex(list2) – מחברת את list2 לסוף הרשימה ,מעדכן מצביעים (merge), מעדכן את אורך וגודל הרשימה לפי אורך וגודל list2.
      3. plantBefore(list2, nodeAfter) – קורא לannex במידה וnodeAfter == null, ואם לא, שותל את list2 לפני nodeAfter, מעדכן גודל ואורך, מעדכן root במידה וצריך.
   8. מתודות שליפה ומחיקה:
      1. deleteNode(node) – משנה את root, tail אם צריך, מעדכנת את minNode אם צריך, קוראת לeject(), מעדכנת את גודל ואורך הרשימה, ומחזירה את הצומת.
      2. deleteKey(key) – עוברת על הרשימה, כשמצאה את הצומת הרלוונטית, קוראת לdeleteNode ומחזירה את הצומת.
      3. deleteMin = deleteNode(minNode). לא מחזירה כלום.
   9. פעולות נוספות:
      1. updateMin() – עבור מחיקות. עוברת על כל הרשימה וקובעת מצביע לצומת המינימלי
   10. איטרטורים:
       1. Iterarot() – עובר על צמתים
       2. keyIterator() – עובר על מפתחות