

OOP

תרגיל ראשון - מבני נתונים גנריים ומיון דלי. מועד הגשה: יום שלישי ה- 15.3.05

1. (30%)

כתבו מבנה נתונים המייצג מערך גמיש של אובייקטים כלליים. מבנה הנתונים יאפשר גישה בזמן קבוע לאיבר כלשהו וכן הוספת איבר חדש לסוף המערך בזמן ממוצע השואף לקבוע. מבנה הנתונים יקרא ArrayList והוא ישתמש במערך רגיל על מנת לאכסן את האברים שלו. בהתחלה יחזיק ArrayList מערך בגודל אבר בודד. כאשר מוכנס אבר חדש, הוא מוכנס למערך אם יש בו מקום. אם המערך מלא, ייוצר מערך חדש, כפול בגודלו מהקודם, כל אברי המערך הישן יועתקו למערך החדש והאבר החדש יוכנס במקום הפנוי הבא במערך החדש. המערך הישן יהפוך ללא רלוונטי והמערך החדש יהיה זה שבו יחזיק ArrayList את הנתונים. הקובץ Q1.java מדגים שימוש במבנה הנתונים שאתם אמורים ליצור. הקובץ Q1.out הוא הפלט הנדרש מהרצה של Q1. קבצים אלה מגדירים את ההתנהגות הנדרשת מהקוד שלכם. שימו לב שהדפסת אובייקט מסוג ArrayList, מבהירה איזה תאים מלאים באינפורמציה ואיזה ריקים. הפונקציה size() מחזירה את מספר האלמנטים המאוכסנים במבנה הנתונים והפונקציה get מחזירה את האלמנט ה-i. הפעילו את Q1.java עם הקוד שלכם וודאו שהפלט זהה לקובץ Q1.out. את ההשוואה ניתן לבצע ע"י הפניית הפלט לקובץ חדש ואז השוואת הקבצים ע"י תוכנת diff.

2. (40%)

כתבו מבנה נתונים המייצג רשימה משורשרת חד כיוונית בשם LinkedList. הרשימה תאכסן אובייקטים כלליים ותאפשר את הפעולות הבאות: הכנסת אבר לתחילתה, הכנסת אבר לסופה, מחיקת האבר הראשון והחזרתו, המרת רשימה למחרוזת, מיון הרשימה ושירשור שתי רשימות. מיון הרשימה יעשה ע"י אלגוריתם מיון מיזוג (merge-sort). סדר המיון יקבע ע"י אובייקט השוואה שישלח לפונקציית המיון. עליכם ליצור ממשק כללי בשם Comperator המייצג את היכולת לקבוע יחס סדר. הממשק יכיל פונקציה המשווה בין שני אובייקטים וקובעת מי גדול ממי או שהם שווים בגודלם. בנוסף, עליכם ליצור שתי מחלקות הממשות ממשק זה. אחת בשם IntComperator, היודעת להשוות שני אובייקטים מסוג int ומגדירה עליהם את יחס הסדר הרגיל. והשנייה בשם intRevComperator, המשווה גם היא שני אובייקטים מסוג int אבל מגדירה עליהם יחס סדר הפוך. את הקוד של Comperator, intComperator ו- intRevComperator ניתן להוסיף לקובץ של LinkedList.java או למקם בקובץ נפרד. שרשור שתי רשימות, יגרום לרשימה הראשונה להכיל את השרשור, ולרשימה השנייה להיות ריקה. הקובץ Q2.java מדגים שימוש אפשרי ברשימה והקובץ Q2.out מכיל את הפלט שלו. בחינה של Q2.java ו- Q2.out תבהיר לכם מה צריך להיות הממשק של LinkedList ואיך צריכה להתנהג.

3. (30%)

מיון דלי הוא מיון פשוט העובד בצורה יעילה כאשר המספרים מתפלגים באופן אחיד בתחום. הרעיון הבסיסי הוא לחלק את תחום המספרים – מהמספר הקטן ביותר עד הגדול ביותר ל- n קטעים שווים הנקראים דליים. לדוגמה, אם המספר הקטן ביותר הוא 1, המספר הגדול ביותר הוא 10 ומספר הדליים הוא 3, אז הדלי הראשון הוא הקטע בין 1 ל- 4, השני הוא הקטע בין 4 ל- 7 והשלישי בין 7 ל- 10. לכל דלי יש מבנה נתונים המאכסן את כל המספרים בקטע שהוא מייצג. בהתחלה, האלגוריתם מחלק את כל המספרים לדליים המתאימים להם, לאחר מכן הוא ממין כל דלי בנפרד ולבסוף הוא משרשר את תחולת כל הדליים הממוינים לפי הסדר.

עליכם להשתמש במבני הנתונים שיצרתם בשאלות הקודמות על מנת לממש את האלגוריתם. צרו ArrayList שמחזיק רשימות משורשרות של אובייקטי Double. כל רשימה משורשרת תייצג דלי. חלוקת המספרים לדליים אמורה להתבצע ע"י חישוב אריתמטי של אינדקס הדלי. מיון כל דלי יעשה בעזרת מיון המיזוג שמימשתם בשאלה הקודמת.

את האלגוריתם יש לממש כפונקציה סטטית בשם bucketSort הנמצאת במחלקה בשם Algorithms. הפונקציה bucketSort אמורה לקבל מערך של מספרים למיון (מסוג double), את מספר הדליים ואובייקט השוואה לצורך מיון. הקבצים Q3.java ו- Q3.out מדגימים את השימוש בקוד שלכם. הפלט שלכם לא אמור להיות זהה ל- Q3.out מכיוון שהזמנים תלויים במחשב שעליו הורצה התכנית. שימו לב ש- Q3.java משתמש באובייקט השוואה בשם DoubleComperator. עליכם להוסיף מחלקה בשם זה המגדירה יחס סדר רגיל על אובייקטי Double.

אופן ההגשה : נא לפעול לפי הוראות הגשת התרגילים שבאתר.

הערות

- קראו את דרישות סגנון התכנות בקורס ופעלו לפיהם.
- בתרגיל זה אין להשתמש במבני נתונים מהספרייה הסטנדרטית של Java.
- בתרגיל זה אין חובה לטפל בבעיות המתגלות בזמן ריצה ודורשות זריקת Exception. מובן שטיפול כזה הוא רצוי, וכדאי להוסיפו בשביל לשפר את הקוד, אך לא יורדו נקודות אם הוא חסר.

בהצלחה!