

# OOP

תרגיל שני: מחלקות פנמיות, איטרטורים ו - gui בסיסי.  
הגשה: יום שלישי ה - 5.4.05, עד סגירת בנין רוס.

## 1. (15%)

השתמשו במבני הנתונים LinkedList ו - ArrayList שכתבתם בתרגיל הקודם והוסיפו להם איטרטורים. הממשק של האיטרטור מוגדר ע"י הקובץ Iterator.java שבתיקיית התרגיל. עליכם לממש איטרטור רגיל לרשימה המשורשרת, איטרטור רגיל עבור המערך הגמיש ואיטרטור נוסף עבור המערך הגמיש שעובר על אבריו בסדר הפוך. הקבצים Q1.java ו - Q1.out מגדירים את ההתנהגות הנדרשת.  
אם מסיבה כלשהי אינכם מעוניינים להשתמש במבני הנתונים שכתבתם בתרגיל הקודם, ניתן להשתמש בפתרון לתרגיל 1, המפורסם באתר. שמו לב עם זאת שפתרון זה איננו בטוח לחלוטין (נכתב באישון לילה ולא נבדק כראוי), והאחריות לתקינות הקוד שאתם מגישים, עדיין עליכם. מותר גם להשתמש בפתרונות של חברים. בכל מקרה שאתם משתמשים בקוד שלא אתם כתבתם יש לציין זאת בתיעוד הקובץ (אין צורך לתעד קוד זה).

## 2. (35%)

בשאלה זה אתם מתבקשים לממש מבנה נתונים של גרף לא מכוון. כל קדקוד בגרף יכול אובייקט כללי (מטיפוס Object). הגרף יאפשר הכנסת קדקוד, יצירת צלע בין שני קדקודים קיימים, ומספר שירותים נוספים. ההתייחסות לקדקוד קיים היא ע"י שימוש במספרו הסידורי (ביחס לסדר הכנסת הקדקודים לגרף).  
בנוסף, הגרף יספק שני איטרטורים: האחד שעובר על הקדקודים לפי מספרים הסידורי, והשני שמבצע סריקת עומק - Depth First Search (או DFS).  
סריקת DFS דומה לחיפוש יציאה ממבוך: כל פעם בוחרים את המעבר הראשון האפשרי וממשיכים בו. אם נתקעים בדרך ללא מוצא, חוזרים למקום האחרון בו הייתה אופציה שלא ניסינו, וממשיכים בה. סריקה בצורה כזו מבטיחה שנבדוק כל מקום במבוך (טענה הדורשת הוכחה, כמובן). קיימות מספר דרכים לממש סריקת עומק. אנו נביא כאן אלגוריתם אחד, שעשוי להתאים לכם:

אתחול: הכנס את קדקוד תחילת הסריקה למחסנית (push)  
לולאה:

הוצא מהמחסנית קדקוד (pop), עם כבר ביקרת בו, הוצא שוב. כך עד שיהיה בידך קדקוד חדש.

הכנס את כל שכניו של הקדקוד החדש למחסנית (push)

הסריקה נגמרת כאשר המחסנית ריקה.

מדובר כמובן רק ב - pseudo-code והרבה פרטי מימוש נשארים פתוחים ואתם חופשיים לשנות וליעל. נכונות האלגוריתם, דורשת הוכחה אבל אינטואיציה אפשר לקבל ע"י מספר סימולציות. הקבצים Q2.java ו - Q2.out מגדירים את ההתנהגות הנדרשת.

המימוש הפנימי של הגרף יהיה ע"י מערך גמיש של רשימות משורשרות של שכנים.

## 3. (50%)

בשאלה זו תתבקשו לממש אפליקציה המאפשרת עבודה עם גרף דרך ממשק גרפי. הקוד שלכם אמור לאפשר יצירה וויזואלית של גרף, שינוי שלו, והפעלת איטרטורים עליו. הקובץ Q3.jar מדגים איך התוכנית שלכם אמורה להתנהג. בכדי להפעיל אותו יש להוריד את הקובץ לתיקיה ריקה, ואז לבצע את הפעולות הבאות:

```
$ jar xvf Q3.jar  
$ java Gui
```

(ה - '\$' מסמן תחילת שורת shell)

בכדי להוסיף קדקוד חדש יש ללחוץ על העכבר במקום פנוי. בכדי לבחור קדקוד יש ללחוץ עליו ואז הוא משנה את צבעו לאדום. בכדי להוסיף צלע יש לבחור קודקוד ואז ללחוץ על קדקוד אחר. בשביל להזיז קודקוד יש לבחור אותו ואז ללחוץ על מקום פנוי (שימו לב שהקשרים שלו משתנים בהתאם). הכפתורים למטה מאפשרים ציור מחודש של הגרף, ניקוי הגרף ומעבר סידרתי על הקדקודים לפי סדר DFS או לפי סדר הוספתם. הכפתורים Button1 ו- Button2 הם לשימושכם האישי ואתם יכולים לממש בהם מה שתמצאו.

איך מתחילים?

הממשק הגרפי נכתב עבורכם ונמצא בקובץ הארכיב Gui.jar. הממשק יודע לנהל חלון גרפי פשוט ולקרוא לפונקציות המתאימות בתגובה למאורעות שונים כמו לחיצה על כפתור או לחיצה על העכבר. הפונקציות להם קורא הממשק נמצאות באובייקט מסוג App המייצג את האפליקציה שלכם. בניגוד לתוכניות שכתבתם עד כה, בשאלה זו, לא תכתבו פונקציות main() מכיוון שפונקציה זו נמצאת כבר ב- Gui. התפקיד שלכם הוא לממש את הפונקציות באובייקט ה- App המגיבות למאורעות השונים. בתיקית התרגיל נמצא הקובץ App.java שיכול לשמש כשלד לתוכנית שלכם.

בכדי להתחיל לעבוד, כדאי להוריד מהאתר את הקבצים Gui.jar ו- App.java, לשים אותם בתיקייה ריקה, לפתוח, לקמפל ולהריץ:

```
$ jar xvf Gui.jar
$ javac *.java
$ java Gui
```

נסו ללחוץ על Button1 ו- Button2 ולאחר מכן לקרוא את הקוד של App.java.

בשלב הראשון אנו ממליצים לממש פונקציה המציירת קו ישר בין שתי נקודות. בכדי לבדוק את התקינות שלה בכל המצבים אפשר לצייר ציור הדומה לציור שמתקבל בלחיצה על Button1 ב- Q3.jar.

בכדי לממש את האפליקציה מומלץ (כרגיל) לנסות לפרק את המטלה לפונקציות פרטיות רבות, שעושות מטלות פשוטות וברורות, ושיש להם שם מתאים.

#### בונוס :

ישנם אלגוריתמים רבים על גרפים שאפשר לממש, ולזכות בזכותם בנקודות נוספות. גודל הבונוס יהיה פרופורציוני למורכבות האלגוריתם. דוגמאות לאלגוריתם אפשריים:

- בבחירת שני קדקודים, סמן את המסלול הקצר ביותר ביניהם.
- צבע רכיבי קשירות שונים בצבעים שונים.
- אינדיקציה האם הגרף מכיל מעגלים.

חשוב לציין בקובץ ה- readme שלכם, כל בונוס שאתם ממשים. יש להסביר מה האלגוריתם שמימשתם ואיך המשתמש יכל להפעיל אותו. יש להשתמש רק בכפתורים Button1 ו- Button2 לצורך הבונוסים, ולא בכפתורים האחרים.

**הערה:** גם בתרגיל זה אין חובה לטפל בבעיות המתגלות בזמן ריצה ודורשות זריקת Exception. כמו כן, אסור להשתמש במבני נתונים של הספרייה הסטנדרטית.

#### הגשה:

יש להגיש את כל קבצי ה- java ו- class שסופקו על ידינו בכדי שאחרי קומפילציה כזו:

```
$ javac *.java
```

יהיה אפשר להריץ את התוכניות באופן הבא:

```
$ java Q1
$ java Q2
$ java Gui
```

ולקבל פלט השווה ל- Q1.out, Q2.out והתנהגות דומה ל- Q3.jar.

**בהצלחה !**