אוניברסיטת בן גוריון

הפקולטה למדעי המחשב

מבוא לפירוק והרכבה של תכניות – פרויקט סיום

*מטרת ונושא הפרוייקט: הדגמת אלגוריתמי code-motion וrefactoring שנלמדו במהלך הסמסטר.*

מרצה הקורס: ד"ר רן אטינגר

מגישים: טל יצחק (204260533)

מאי נגלר (200935963)

כלל הקוד בפרויקט נכתב בJava, וזמין ב-GitHub:

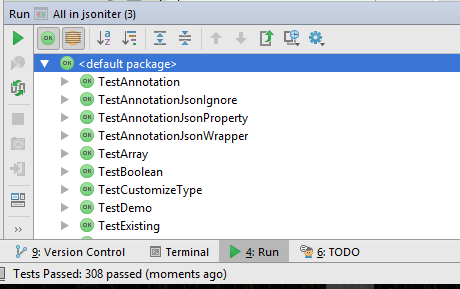
github.com/talitz/Introduction-to-Code-Motion-Refactoring-Project

סמסטר ב', תשע"ז, 2017

רקע

הפרויקט מממש Parser גמיש ומהיר ל-JSON, הזמין בשפות Java ו-Go. הפרויקט נלקח מהrepository הזמין לצפייה בקישור הבא: <https://github.com/json-iterator/java>.

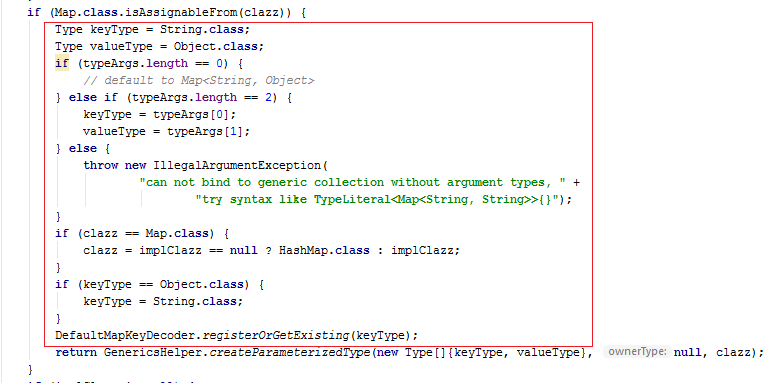
הפרויקט נכתב עם סביבת טסטים ב- JUnit, להלן המחשה של מעבר כלל הטסטים:



בפרויקט נציג דוגמאות מקטעי קוד נבחרים ונריץ את האלגוריתמים שלמדנו בכיתה לrefactoring & code motion.

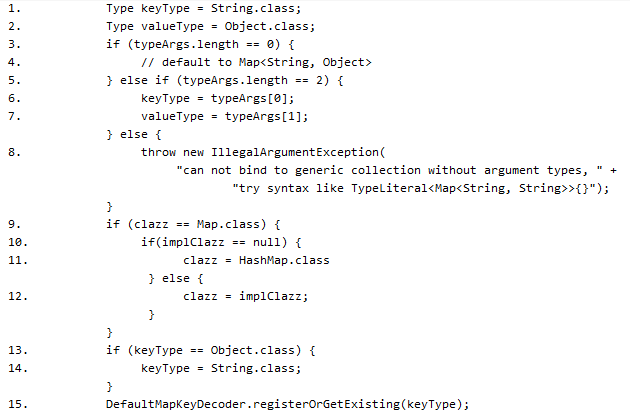
דוגמא 1: PDG-Based Sliding

נתבונן בחתיכת הקוד הבאה:

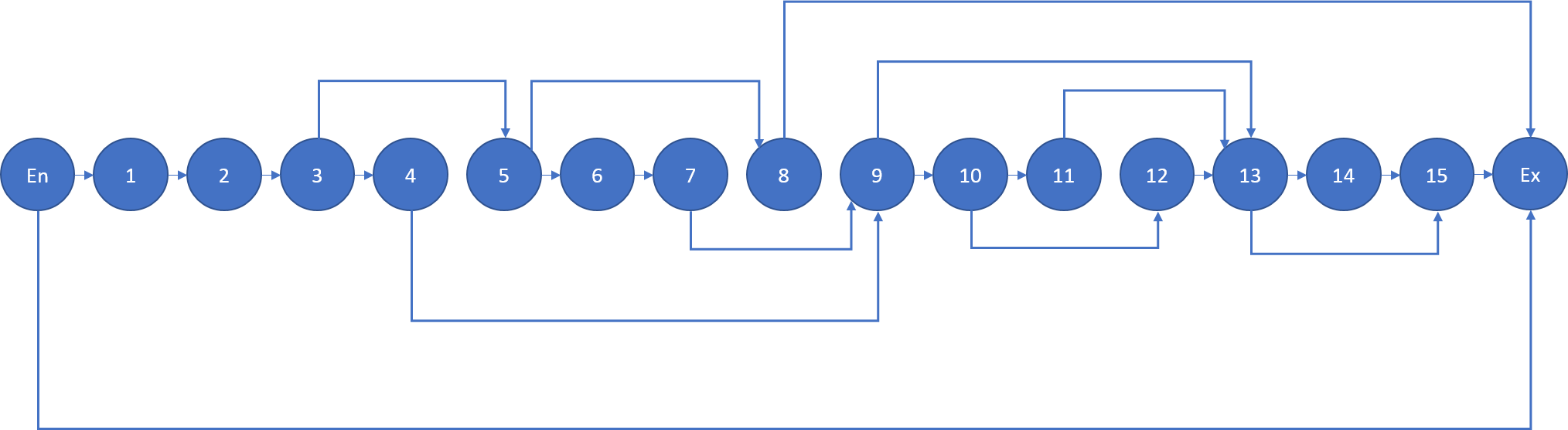


נרצה לבצע PDG-Base Sliding כאשר קבוצת המשתנים שנבחר תהיה .

נמספר את כלל השורות בקוד:



נבנה את גרפי ה- CFG ו- PDG:



כבסיס לסליידינג על קטע הקוד המורכב מהפקודות 1-15 שלעיל, נקבל את הייצוג הבא: קבוצת הקודקודים N היא:

וקבוצת הקשתות E הכוללת תלויות מסוג control (הקבוצה Ec) ותלויות data (הקבוצה Ed) מסוג flow או anti כדלהלן:

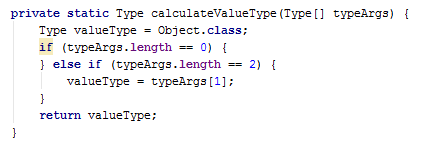
|  |  |  |
| --- | --- | --- |
| Vars | Kind | Edge |
| - | Control | (Entry,1) |
| - | Control | (Entry,2) |
| - | Control | (Entry,3) |
| - | Control | (3,4) |
| - | Control | (3,5) |
| - | Control | (3,9) |
| - | Control | (3,13) |
| - | Control | (3,15) |
| - | Control | (3,16) |
| - | Control | (5,6) |
| - | Control | (5,7) |
| - | Control | (5,8) |
| - | Control | (5,13) |
| - | Control | (5,15) |
| - | Control | (5,16) |
| - | Control | (9,10) |
| - | Control | (10,11) |
| - | Control | (10,12) |
| - | Control | (13,14) |
| {clazz} | Flow | (Entry,9) |
| {implClazz} | Flow | (Entry,10) |
| {implClazz} | Flow | (Entry,12) |
| {keyType} | Flow | (1,13) |
| {keyType} | Flow | (1,15) |
| {keyType} | Flow | (6,13) |
| {keyType} | Flow | (6,15) |
| {clazz} | Anti | (9,11) |
| {clazz} | Anti | (9,12) |
| {keyType} | Anti | (13,14) |
| {keyType} | Flow | (14,15) |
| {keyType} | Flow | (1,Exit) |
| {keyType} | Flow | (6,Exit) |
| {keyType} | Flow | (14,Exit) |
| {valueType} | Flow | (2,Exit) |
| {valueType} | Flow | (7,Exit) |
| {clazz} | Flow | (11,Exit) |
| {clazz} | Flow | (12,Exit) |
| {IllegalArgumentException} | Flow | (8,Exit) |

כעת נפעיל סליידינג עבור .

תוצאת הסליידינג:

1. הקוד המחולץ (הסלייס, תת הקבוצה Nv) יהיה:
2. הקוד המשלים את הקו-סלייס יהיה (Ncov):

נבצע Extract Method לslice:



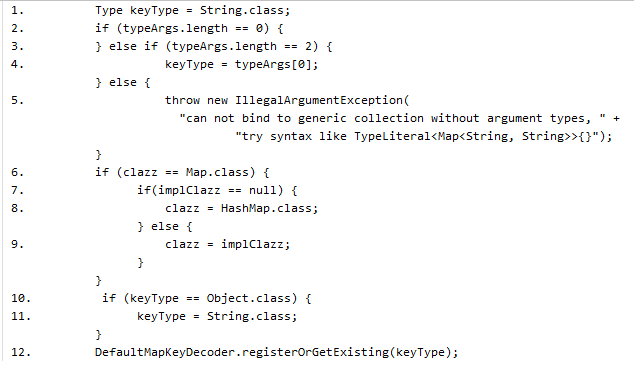
נתבונן כעת בקוד, רגע לפני ביצוע Extract Method ל-co-slice:



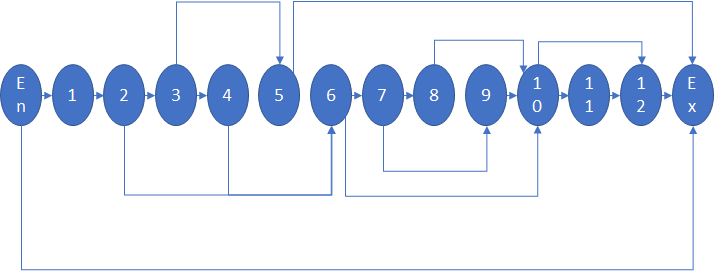
נשים לב שמתבצע חישוב בco-slice עבור המשתנים clazz ו-keyType, כלומר כדי לבצע Extract Method ל-co-slice, נצטרך לאחסן את הנתונים החוזרים מהפונקציה באובייקט שיכיל 2 שדות עבור ערכים אלו (זה מה שמוצע ע"י IntelliJ כשמבצעים Extract Method במצב שכזה).

אולם, אנו מזהים כאן פוטנציאל להריץ שוב את אלגוריתם ה-PDG Based Sliding, על קבוצת המשתנים .

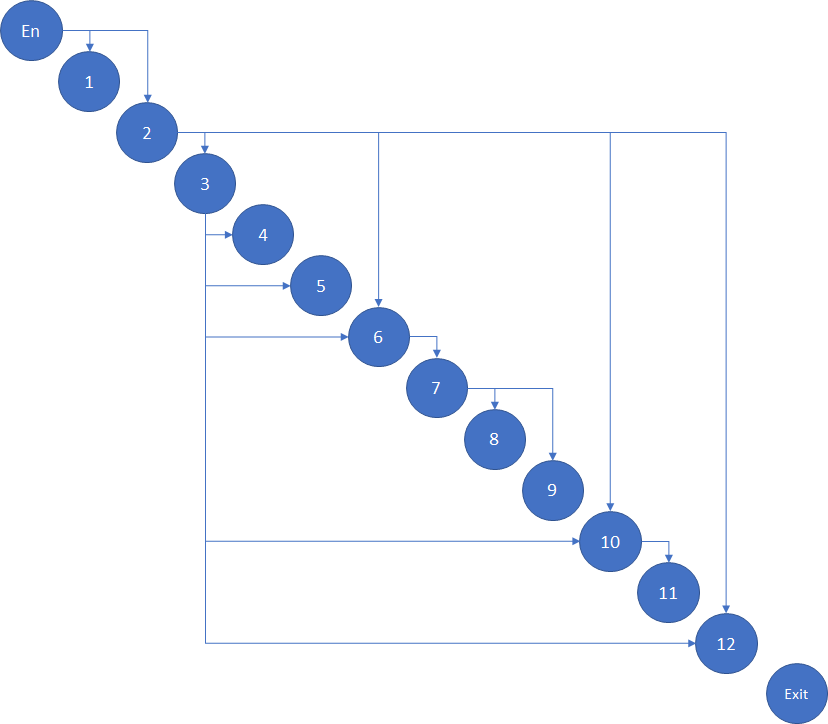
נמספר את שורות הקוד מחדש (עבור קטע ספציפי זה):



נבנה את גרף ה-CFG:



גרף ה- PDG (רק קשתות Control, את קשתות הData נפרד בטבלה לצורכי נוחות ותמונה ברורה של הגרף):



כבסיס לסליידינג על קטע הקוד המורכב מהפקודות 1-12 שלעיל, נקבל את הייצוג הבא: קבוצת הקודקודים N היא:

וקבוצת הקשתות E הכוללת תלויות מסוג control (הקבוצה Ec) ותלויות data (הקבוצה Ed) מסוג flow או anti כדלהלן:

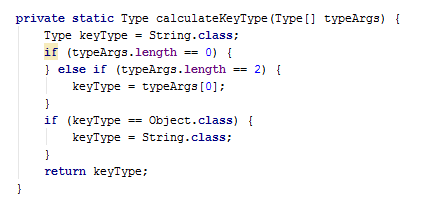
|  |  |  |
| --- | --- | --- |
| Vars | Kind | Edge |
| - | Control | (Entry,1) |
| - | Control | (Entry,2) |
| - | Control | (2,3) |
| - | Control | (2,6) |
| - | Control | (2,10) |
| - | Control | (2,12) |
| - | Control | (3,4) |
| - | Control | (3,5) |
| - | Control | (3,6) |
| - | Control | (3,10) |
| - | Control | (3,12) |
| - | Control | (6,7) |
| - | Control | (7,8) |
| - | Control | (7,9) |
| - | Control | (10,11) |
| {clazz} | Flow | (Entry,6) |
| {implClazz} | Flow | (Entry,7) |
| {implClazz} | Flow | (Entry,9) |
| {keyType} | Flow | (1,10) |
| {keyType} | Flow | (1,12) |
| {keyType} | Flow | (1,Exit) |
| {keyType} | Flow | (4,10) |
| {keyType} | Flow | (4,12) |
| {keyType} | Flow | (4,Exit) |
| {clazz} | Anti | (6,8) |
| {clazz} | Anti | (6,9) |
| {IllegalArgumentException} | Flow | (5,Exit) |
| {clazz} | Flow | (8,Exit) |
| {clazz} | Flow | (9,Exit) |
| {keyType} | Anti | (10,11) |
| {keyType} | Flow | (11,12) |
| {keyType} | Flow | (11,Exit) |

כעת נפעיל סליידינג עבור .

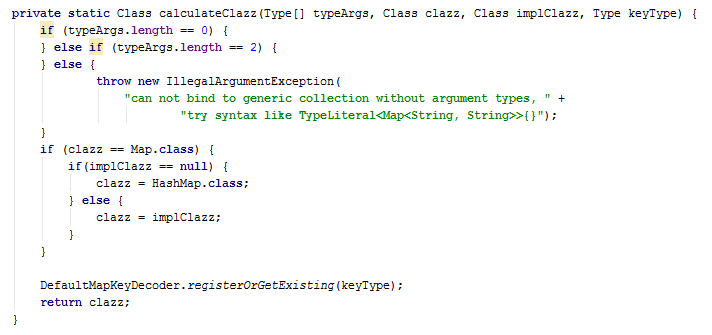
תוצאת הסליידינג:

1. הקוד המחולץ (הסלייס, תת הקבוצה Nv) יהיה:
2. הקוד המשלים את הקו-סלייס יהיה (Ncov):

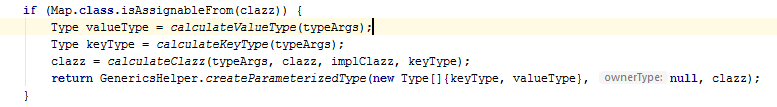
כעת נוכל לבצע Extract Method לslice:



ולco-slice:

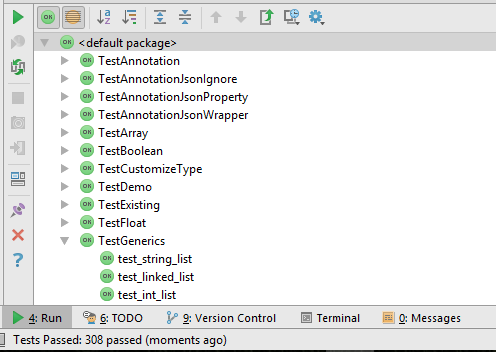


ובסה"כ נקבל את הקוד הבא:



נוודא כעת שכלל הטסטים עדיין עוברים (קליק ימני על הפרוייקט –

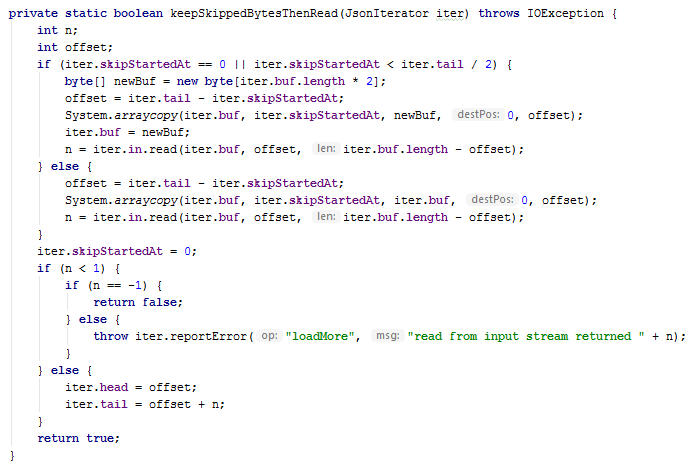
run 'All Tests'):



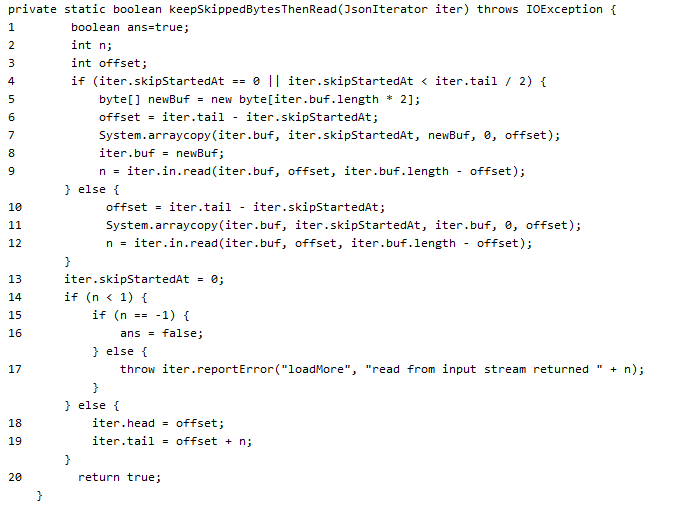
כנדרש ☺

דוגמא 2: Slide-Based Bucketing

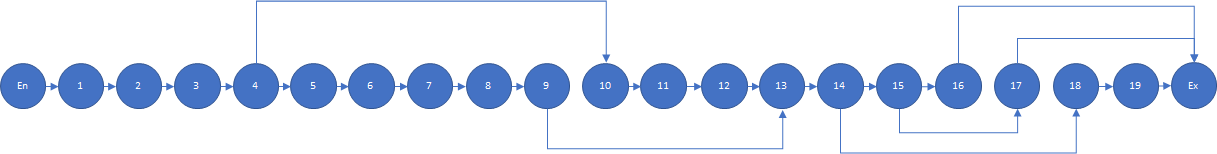
נתבונן בקטע הקוד הבא מהמחלקה IterImplForStreaming:



נמספר את שורות הקוד ונבצע עיבוד מקדים:



נבנה את גרף ה-CFG המתאים:



בלה..