טרנספורמציות לשיפור איכות קוד קיים – Advanced Code Refactoring

קוד קורס: 0368-3316-01

שנת הלימודים תשע"ה, סמסטר א', אוניברסיטת תל אביב

היקף הקורס: 3 שעות

דרישת קדם: תוכנה 1 (קורס 0368-2157-01).

מרצה: ד"ר רן אטינגר. שעות קבלה: אחרי השיעור, בתיאום מראש.

תיאור הקורס:

בימים בהם מרווחי הזמן בין שחרורי גרסאות תוכנה מתקצרים, ובימים בהם מקובל לפתח משפחות מוצרים, במגוון גרסאות, מפתחי התוכנה נדרשים לעדכן את קוד המקור באופן תדיר. היכולת לעדכן את מבנה הקוד מבלי לשנות את הפונקציונליות שלו, ובמילים אחרות היכולת לבצע "ריפקטורינג", מהווה נדבך משמעותי ביכולתנו לתכנת ביעילות ובמהירות מבלי לפגוע באיכות הפנימית והחיצונית של התוצרים – כך שנוכל להמשיך ולפתחם לאורך זמן, ללא צורך "להתחיל מחדש" לעיתים קרובות מידי. לפיכך, טכניקות ריפקטורינג מאפשרות שמירה על עיצוב נכון וטוב של מערכות תוכנה, בזמן שבלעדיהן כל כך מקובל וכמעט בלתי נמנע שקוד שעובר עדכונים ושינויים רבים מאבד מאיכות העיצוב המקורי.

בקורס זה נלמד להכיר טכניקות ריפקטורינג מגוונות ואת דרכי השימוש בהן לצורך שיפור איכות קוד קיים. נתרכז בטכניקה לחילוץ פרוצדורה חדשה (Extract Method) מקטעי קוד קיימים. נלמד למשל כיצד להשתמש בפעולה זו לצורך ביטול כפילויות על ידי מיזוג עותקים שונים של קוד ששוכפל והשתנה. טרנספורמציית קוד זו מאתגרת במיוחד במקרים בהם השינויים בוצעו באופן שאינו אחיד בכל העותקים.

פעולות ריפקטורינג כ-Extract Method ניתנות לביצוע ידני על ידי המתכנת, או בעזרת כלים אוטומטיים. נלמד בקורס מגוון אלגוריתמים לביצוע פעולות אלה, גם במקרים מורכבים של חילוץ מספר בלוקים של קוד (שאינם רצופים) לכדי פרוצדורה חדשה אחת – ללא פגיעה בפונקציונליות של הקוד המקורי. אוטומציה שכזו דורשת יכולות ניתוח קוד מתקדמות, אותן נסקור בהרחבה בקורס.

נושאי הקורס העיקריים:

- מבוא ל-ריפקטורינג ושימושיו, כולל סקירה של מגוון טכניקות מהספרות, כגון Extract Method, Split Loop, Replace Temp with Query, Separate Query from Modifier, Compose Method, Form Template Method, Convert Procedural Design to Objects, ו-Separate Domain from Presentation. (דוגמאות הקוד כתובות בשפת ג'אווה.)
- שיטות לייצוג קוד עבור ניתוח אוטומטי של תוכנית באופן סטטי (כלומר מבלי להריץ או לדעת את ערכי הקלט של התכנית), תוך התייחסות לנושאים כ-control flow ו-dataflow, ותוך שימוש במבני נתונים כגון control flow graph וסוגים שונים של dependence graphs.
- שיטות לפירוק והרכבה של תכניות, כגון program slicing, fine slicing, ו-sliding.
- מגוון אלגוריתמים לטרנספורמציות הזזת קוד, לצורך הפיכת מספר בלוקים של פקודות לכדי בלוק אחד (רצוף), כהכנה לחילוצו לפרוצדורה חדשה.
- מגוון שימושים בטרנספורמציות ההזזה והחילוץ למטרת שיפור איכות הקוד, למשל עבור ביטול כפילויות (clone elimination) ושיפור מדדי איכות כ-cohesion.

הרכב ציון הקורס: 70% מבחן סופי (בו כל חומר עזר שאינו אלקטרוני מותר לשימוש), 30% תרגילי בית.

מטלות: כשלושה תרגילי בית תיאורטיים, הניתנים להגשה ביחידים או בזוגות. התרגילים (כמו המבחן הסופי) יידרשו וייבדקו היכרות מעמיקה עם האלגוריתמים הנלמדים. **אין** תרגילי תכנות. **אין** דרישה להוכחת נכונות האלגוריתמים או הוכחת תכונות תיאורטיות אחרות.

ספרות חובה: **אין**. כל החומר הדרוש להצלחה בתרגילים ובמבחן הסופי נלמד בכיתה.

ספרות מומלצת: האלגוריתמים הנלמדים בקורס מתוארים במספר מאמרים מהספרות האקדמית ועבודות מאסטר/דוקטורט. בנוסף, את תיאור טכניקות הריפקטורינג הנלמדות ודוגמאות הפעלה ניתן למצוא בספרים הבאים (שלושתם בהוצאת Addison-Wesley):

- *Refactoring: Improving the design of existing code*. Martin Fowler (1999).
- *Refactoring to Patterns*. Joshua Kerievsky (2004).
- *Refactoring Workbook*. William C. Wake (2003).