הצעה לממ"ן התכנותי

נגה קליין 318295144

הקדמה

בחרתי להגיש תרגיל תכנותי הממש את סימולציה של מודל "החומה הסינית".

מודל זה מכיל subjects, objects, data set, CI. במודל זה אם משתמש ניגש לעצם מסוים במחלקת קונפליקט מסוימת, אזי אסור לו לגשת לאף עצם נוסף באותה מחלקת קונפליקט, כלומר נבנית "חומה" סביב מחלקת הקונפליקט.

בתרגיל תכנותי זה ארצה לממש את המדיניות המוזכרת לעיל, כפי שנתבקשנו בממ"ן 1 תרגיל 10, באמצעות מודל HRU, כלומר הכללה של מטריצה הגישה.

תשובתי לתרגיל 10 לפיה אממש את המודל:

- *נרצה לממש את מדיניות ה"קיר הסיני" באמצעות מודל* HRU. *נניח מספר קטן של מחלקות קונפליקט.*

א. טבלת הגישה A *תכיל:*

a. *שורות – כמספר הנושאים, עצמים,* dataset *ומחלקות הקונפליקט*

b. *עמודות – כמספר הנושאים, עצמים,* datasets *ומחלקות קונפליקט.*

עבור O *עצם ב* A[O, O] *ייכתב ה* dataset *אליו* O *שייך.*

עבור D, dataset *ב* A[D, D] *תכתב מחלקת הקונפליקט אליה* D *שייך.*

ובנוסף עבור נושא S *ומחלקת קונפליקט* C:

ב A[S, O/D/C] *ההרשאות של* S *ביחס ל* O/D/C *בהתאמה.*

הרעיון הכללי הוא שנדע לקשר כל עצם למחלקת הקונפליקט המתאימה לו ובכך לדעת האם פעולות הקריאה והכתיבה אפשריות עבורו.

נסמן $d_1, \dots, d_n$ *ה* datasets *האפשריים ו* $c_1, \dots, c_m$ *מחלקות הקונפליקט.*

ב.

command get_read_access(S, O)
  if Read ∈ A[S, O]
    $\vee \Big(D \in A[O, O] \wedge \big(\text{Read} \in A[S, D] \vee (C \in A[D, D] \wedge \text{Read} \notin A[S, C])\big)\Big)$
  than enter Read to A[S, D], A[S, C], A[S, O]
end

במילים: S *יוכל לקרוא מ* O *אם ישנה ההרשאה המתאימה, או אחד מהשניים מתקיים:*

S *קרא בעבר מתוך ה* dataset *אליו* O *משתייך.*

S *לא קרא בעבר מתוך מחלקת הקונפליקט אליו* O *משתייך.*

החל משלב זה נסמן את המקומות המתאימים בהרשאת הקריאה, וכעת S *לא יוכל לקרוא מ*dataset *אחיו במחלקת הקונפליקט הנתונה וכן יוכל לקרוא מאובייקטים הנמצאים באותו ה* dataset.

command get_write_access(S, O)
  if Read ∈ A[S, O]
    and $D \in A[O, O] \wedge C \in A[D, D] \wedge \forall C' \neq C : \text{Read} \notin A[S, C']$
    than enter Write to A[S, O]
end

במילים: S *יוכל לכתוב ל* O *אם שני התנאים הבאים מתקיימים:*

הרשאת הכתיבה מיוחסת ל S *בעבור האובייקט* O, *מן ההכרח שזה יתקיים אילו יתקיימו התנאים שצוינו לעיל.*

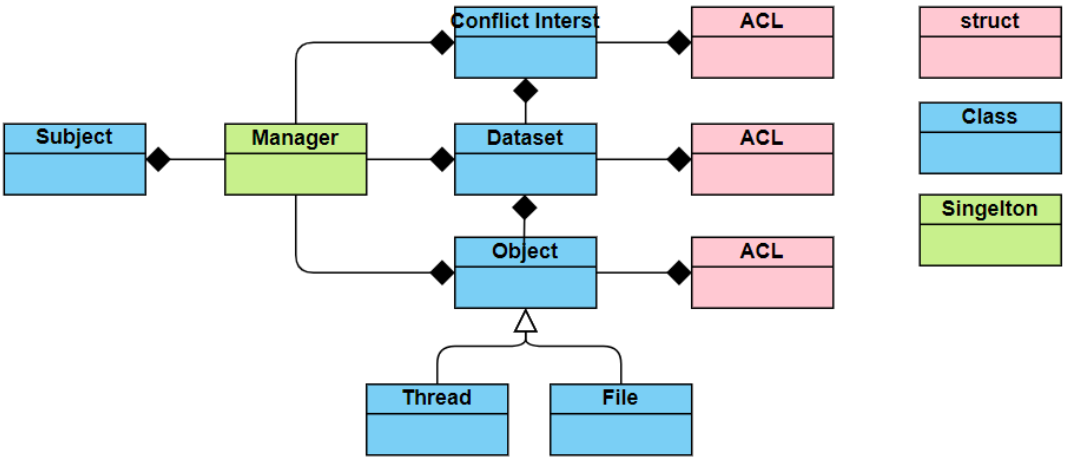
לכל מחלקת קונפליקט השונה מזו ש O *משתייך אליה,* S *לא יכול לקרוא ממנה.*

את מטריצת הגישה אבחר לממשל בצורת Access Control List.
כלומר כל אובייקט במערכת יחזיק ברשימת הרשאות בצמוד לו המגדירה איזה נושא או מערכת מותרים בגישה אל האובייקט וכן לביצוע פעולות עליו.
במערכת זו ה ACL הינה מבנה נתונים מסוג מפה, הממפה שמות של נושאים לרשימת (list) ההרשאות המותרות להם. ניתן להכניס הרשאות ונושאים לרשימה ובאותו האופן למחוק אותם.

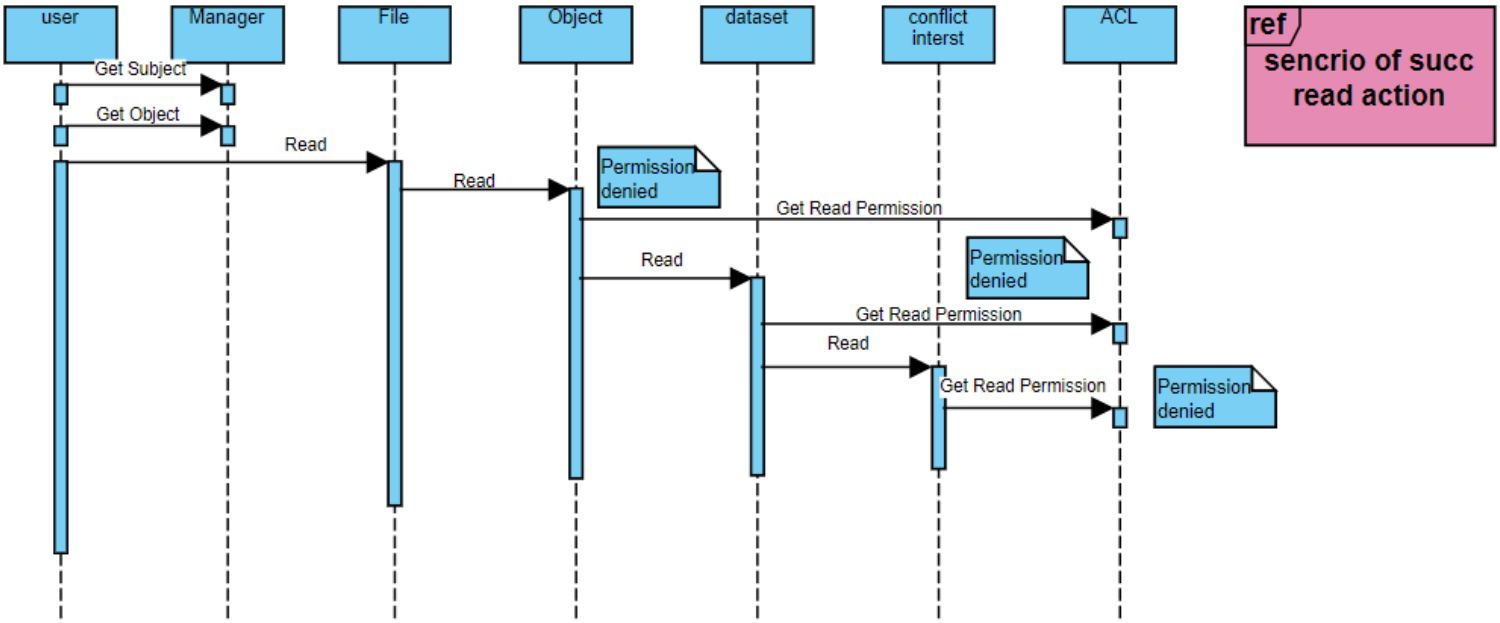
מימוש

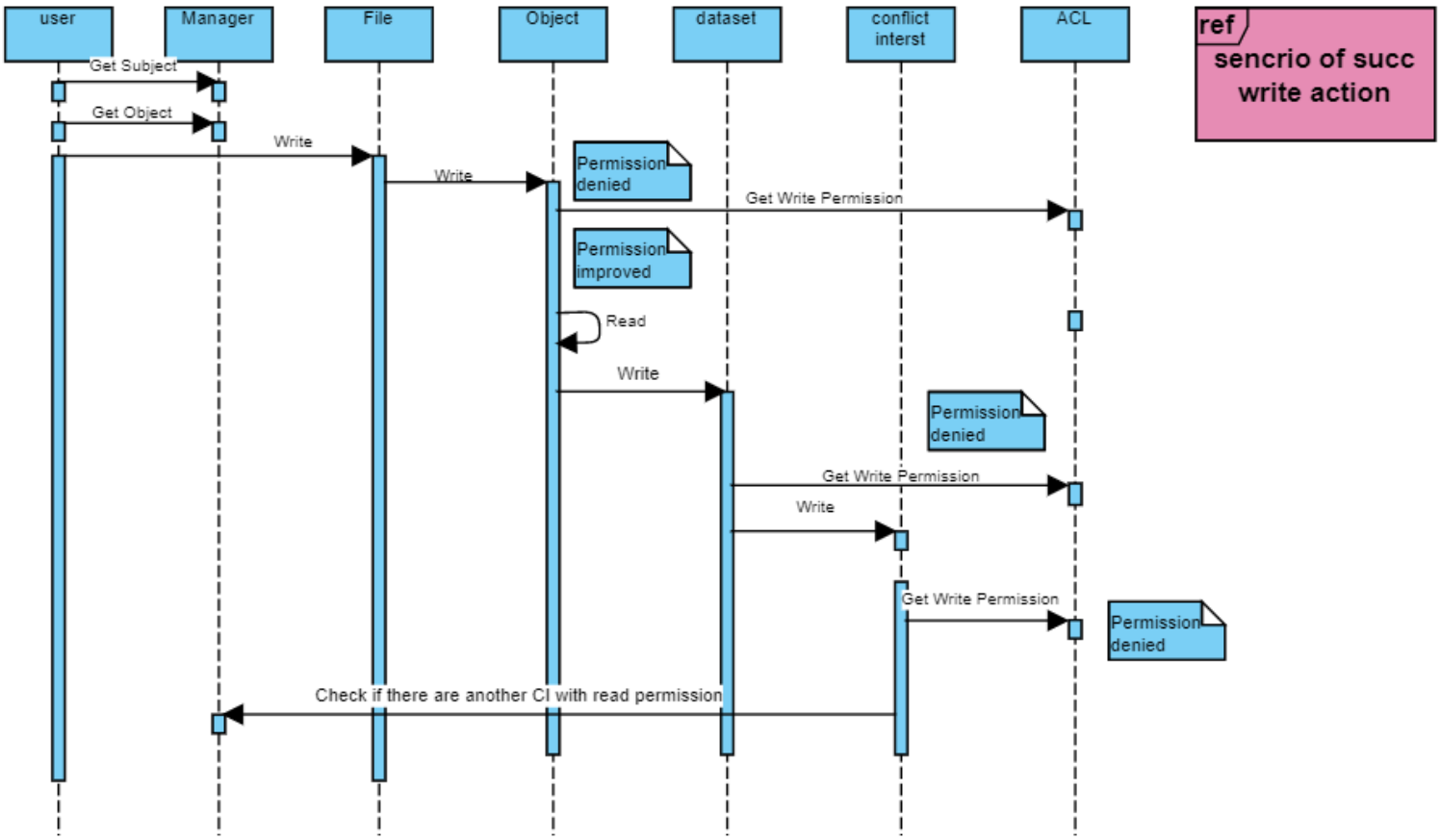
התרגיל ימומש בשפת c++ מעל VS.

מימוש הפרויקט יחולק באופן הבא:



1. מימוש תבנית עיצוב dependency injection – מחלקה Manager אשר מקשרת בין הנושאים (subjects) לאובייקטים (objects) ומצמצמת את התלות ושכפול הקוד.
המחלקה ממומשת כ singleton במערכת, מחזיקה בכל הרשומות בה: subjects, objects, ds, ci ומנהלת אותם ממבט על - מאפשרת הוספה של כל אחד מהם למערכת, מחיקתם ונתינת הרשאות במידת הצורך.
2. Object – מחלקה ממנה ירשו כל האובייקטים בפרויקט, לנוחות אשתמש ב thread, File. כל אובייקט יכיל קישור ל data set יחיד אליו הוא משתייך ואת רשימת ההרשאות שלו.
אובייקט ה File יחזיק במערך תווים בגודל 128 אשר יסמלץ את המידע הנכתב לתוך קובץ.
אובייקט ה Thread יחזיק במערך תווי בגודל 256 אשר יסמלץ את הזיכרון השייך לאותו התהליך.
3. Data Set – מחלקה המכילה קישור למחלקת הקונפליקט אליה משתייכת ואת רשימת ההרשאות שלה. מקושרת ל CI ספציפי.
4. Conflict Interest – מחלקה המכילה את רשימה ההרשאות על מחלקת הקונפליקט.
5. Subject – האובייקטים במערכת.
6. קובץ הרצה main המאפשר למשתמש להשתמש בפקודות ידועות מראש כגון יצירת משתמש ואובייקט, לבקש גישה וכו'...
7. פרויקט Unit Tests הבודק את כלל המערכת.

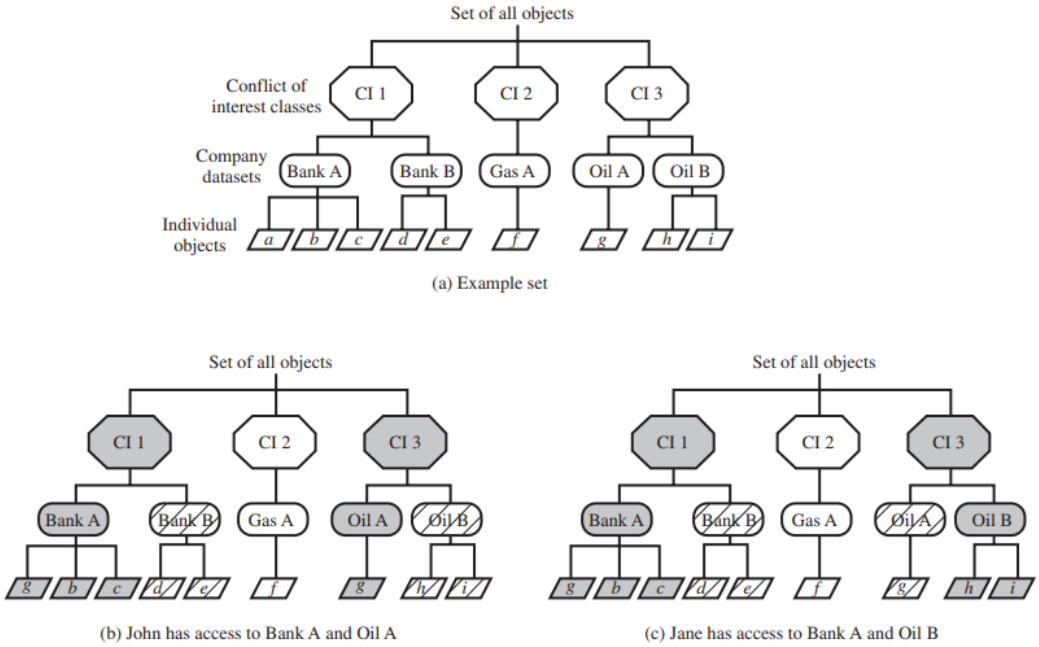

user
Manager
File
Object
dataset
conflict interst
ACL
ref
sencrio of succ read action
Get Subject
Get Object
Read
Read
Permission denied
Get Read Permission
Read
Permission denied
Get Read Permission
Read
Get Read Permission
Permission denied


user
Manager
File
Object
dataset
conflict interst
ACL
ref
sencrio of succ write action
Get Subject
Get Object
Write
Write
Permission denied
Get Write Permission
Permission improved
Read
Write
Permission denied
Get Write Permission
Write
Get Write Permission
Permission denied
Check if there are another CI with read permission

המערכת תפעל באופן הבא:

1. המשתמש יכניס למערך את מספר מסד הנתונים בו ירצה להשתמש.
מסדי הנתונים הינם קבצי json המכילים רשומות "obj%num" באשר המספרים רצים מ 1 ועד לכמות האובייקטים הרצויים במערכת.
כל אובייקט יכיל שם, את סוגו (תהליך או קובץ), את ה dataset , conflict interest שהוא משתייך אליהם, מיהו ה owner שלו ורשימה של נושאים עבור כל הרשאה: כתיבה, קריאה, הפעלה.
2. באופן איטרטיבי המשתמש יוכל להזדהות כאחד מהנושאים במערכת (אם לא קיים נקבל שגיאה ונתבקש לנסות שנית).
3. בכל איטרציה המשתמש יכניס את הפעולה שירצה לבצע: קריאה, כתיבה, הוספת נושא, הוספת אובייקט.
נשים לב כי לא ניתן להוסיף הרשאות באופן ישיר, אלא רק בעת ביצוע פעולות הקריאה, כתיבה (אם ההרשאה התקבלה, היא תתווסף באופן אוטומטי).
4. על מנת לבצע קריאה או כתיבה מקובץ יש להכניס את מספר הבתים הרצויים לפעולה ואת המיקום היחסי בתוך הנתונים.
5. המערכת מממשת את מודל החומה הסינית, על כן simple security rule, *-property rule מתקיימים בה:
   a. ניתן לקרוא מקובץ אם שייך ל DS שנקרא בעבר או שייך ל CI שלא נקרא עדיין – על כן כל DS ו CI יחזיקו גם כן ברשימת הרשאות המעידה האם בוצעה בעבר קריאה מהם.
   בעת הבקשה לקריאה מקובץ, יבדק אם ה DS המתאים מחזיק בהרשאה זו – אם כן, הקריאה מתאפשרת. אם לא נבדק כי ה CI המתאים איננו מחזיק בהרשאה זו.
   b. ניתן לכתוב לקובץ אם אפשר לקרוא ממנו וגם האובייקטים שהמשתמש יכול לקרוא מהם שייכים לאותו DS בלבד, או במילים אחרות ה CI המתאים הוא היחיד שניתן לקרוא ממנו.
   על כן נשתמש במנגנון ה dependency injection – המחזיק בכל הרשומות במערכת, והוא יבדוק האם קיים CI שונה המחזיק בהרשאת הקריאה. אם כן הפעולה תכשל, אחרת תצליח.

הסימולציה אותה בחרתי להציג הינה המקרה המוצג בספר הקורס SB



(a) Example set

(b) John has access to Bank A and Oil A

(c) Jane has access to Bank A and Oil B

במקרה זה ישנם שני משתמשים John, Jane המנסים שניהם לקרוא ולכתוב ממאגר כלשהו.
ניתן לראות כיצד החוקים של מודל זה באים לידי ביטוי בדוגמה זו, ובעיקר כיצד משמש אותנו חוק * ,
למניעת זרימה לא חוקית של מידע, אשר תגרום לניגוד אינטרסים.
אילו ג'ון היה יכול לקרוא את המידע מקובץ g (CI3), אותו ג'יין איננה יכולה לקרוא, ולכתוב אותו לקובץ a
(1CI) אותו ג'יין אכן יכולה לקרוא היינו מקבלים מצב בו המידע מועבר באופן לא חוקי ורצוי.
על כן החלוקה לפי קבוצות אינטרסים המשפיעה על פעולת הקריאה פותרת את המצב -
ג'ון כעת איננו יכול לכתוב מידע אילו הוא יכול לקרוא מ2 קבוצות שונות ולכן בהכרח לא יעביר באופן לא חוקי
מידע לג'יין.