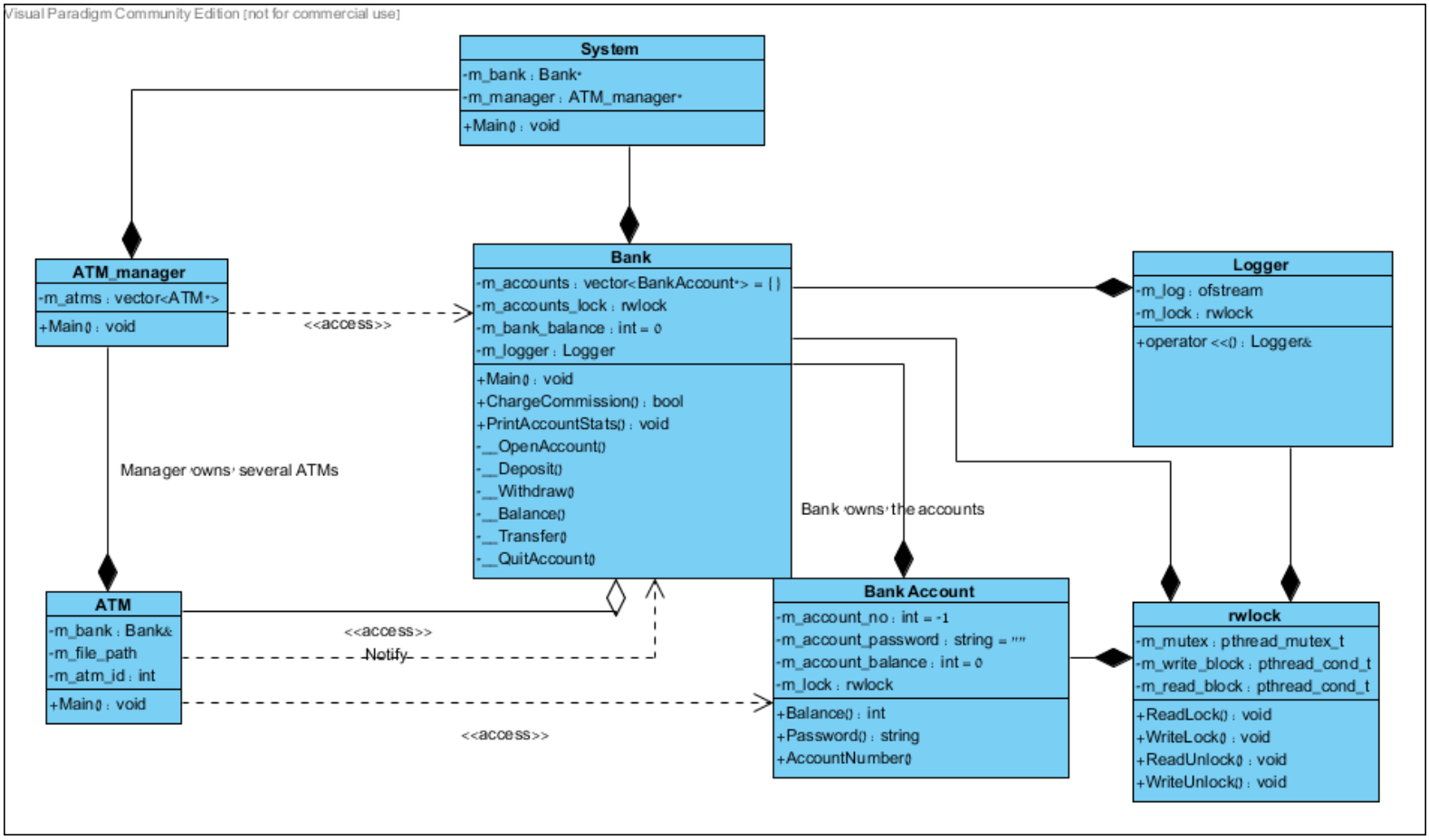
Operating Systems Wet 2

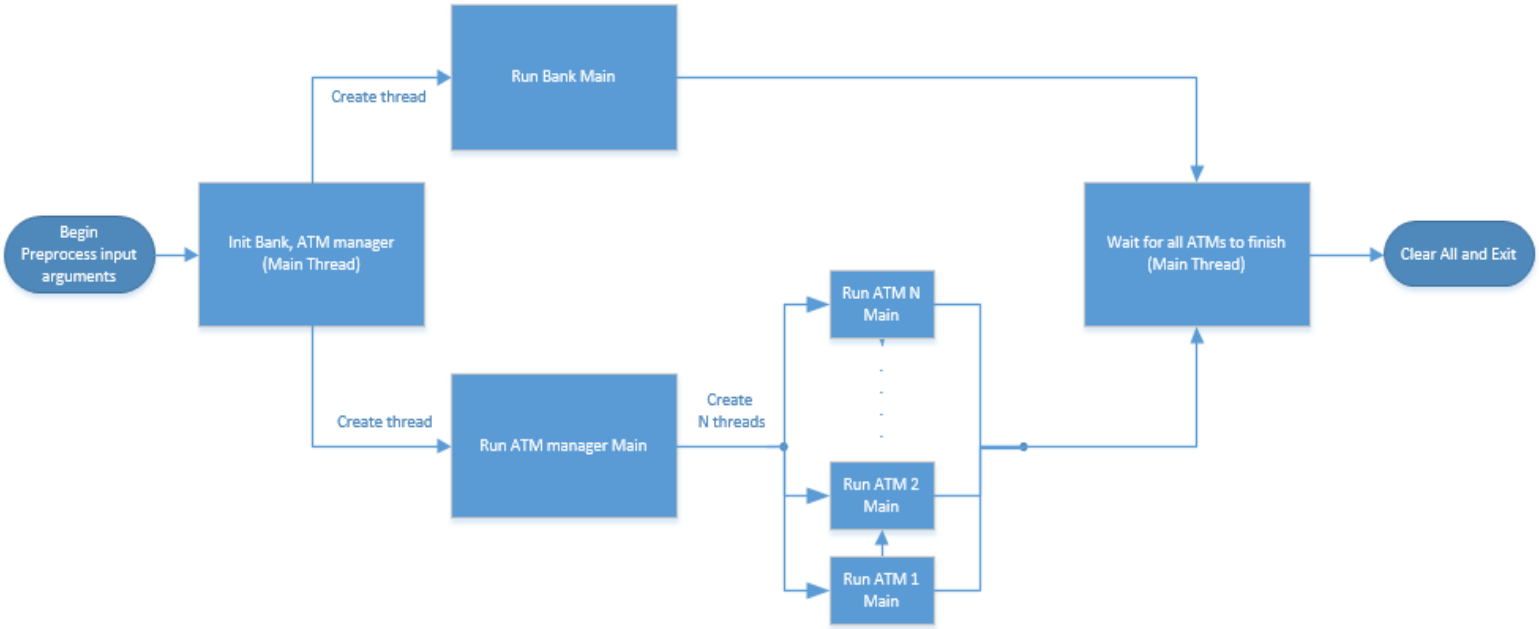
מגישים:

- רון לוי: 201121613
- דרור קבלי 311177109

היררכיית בלוקים של המערכת הממומשת:

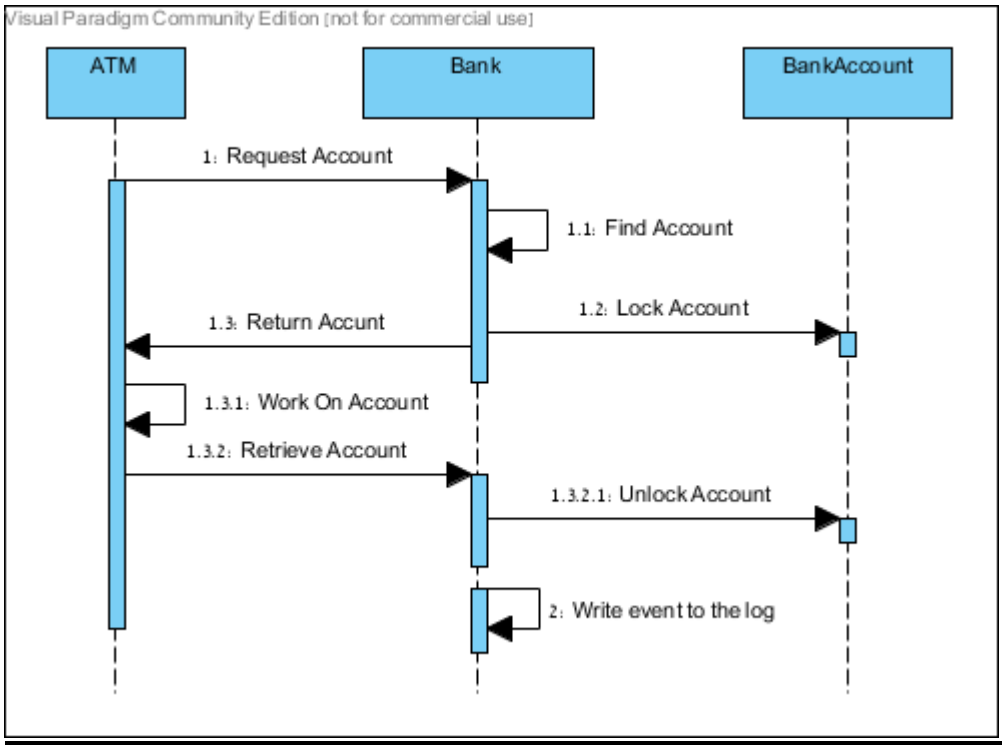


תהליך ריצת התכנית:



כל בלוק אשר לא כתוב בו במפורש שרץ ב-Main Thread, רץ כ-Thread נפרד משלו.

תהליך בקשת פעולה יחידה של כספומט מול הבנק:



ריצת ה-Thread הראשי:

תפקידי ה-Thread הראשי:

1. פענוח הארגומנטים שמועברים לתכנית מהטרמינל
2. אתחול מנהל הכספומטים (שמאתחל N כספומטים שונים) ואתחול הבנק.
3. יצירת Threads N חדשים עבור N הכספומטים הקיימים במערכת + יצירת Thread לריצת התכנית הראשית שמנהלת את אובייקט הבנק.
4. המתנה לכל ה-Threads שנוצרו במערכת, שיסיימו (יבצעו Join).
5. ניקוי הזיכרון שנתפס על ידי התכנית
6. יציאה

ביצוע ה-Threads שנוצרו:

ישנם חוטים נוספים במערכת שנוצרו המתחלקים לשני תפקידים:

1. תפעול 2 פעולות הבנק המרכזיות הרצות בשני threads נפרדים
   a. משיכת עמלות
   b. הדפסת מצב הבנק ל-stdout
2. תפעול N כספומטים שונים ב-N threads נפרדים

מנגנון קוראים-כותבים:

מנגנון הקוראים-כותבים מומש במחלקת ה-rwlock.

המחלקה כוללת:

1. מנעול pthread_mutex_t
2. משתנה תנאי המאפשר נעילה משותפת הנקרא read_block, מסוג pthread_cond_t
3. משתנה תנאי המאפשר נעילה אקסקלוסיבית הנקרא write_block, מסוג pthread_cond_t

מנגנון המנעול מעדיף כותבים על פני קוראים מהסיבה הפשוטה שיש הרבה יותר קוראים במערכת הניגשים למבנה הנתונים של הבנק מבלי לשנותו (הכספומטים) מאשר תהליכונים הדורשים נעילה אקסקלוסיבית (הדפסה למסך דורשת "snapshot" של כל מצבי החשבונות ולכן אסור לתת לאף כספומט גישה לשנות חשבון כלשהו תוך כדי הדפסה, גביית עמלות דורשת גישה לשינוי החשבונות מבלי שאף כספומט אחר "ייגע" בתוכן שלהם) ולכן יש צורך למנוע מצב של "הרעבה" של התהליכונים המשנים את מבנה הנתונים ו/או הדורשים נעילה אקסקלוסיבית כי הם תהליכים המשפיעים על הבנק עצמו ומספקים מידע על נתוני הבנק באופן תדיר.

מנגנון המנעול מוחזק במקומות הבאים:

1. בכל אחד מחשבונות הבנק על מנת לאפשר סינכרון קריאה-כתיבה בין threads שונים שניגשים אליו (כספומטים/בנק)
2. בצמוד למבנה הנתונים השומר על רשימה החשבונות של הבנק. מנעול זה מגן כאמור מקריאה/כתיבה בו זמנית של תהליכי הבנק והכספומטים ובמקרים בהם מוסיפים/מוחקים חשבון בבנק.
כל פעולת כספומט שאינה הוספה/מחיקת חשבון דורשת גישת קריאה למבנה הנתונים (נעילה משותפת), כל פעולת כספומטים שהיא הוספה/מחיקת חשבון דורשת גישת כתיבה למבנה הנתונים מכיוון שהן משנות אותו.
מקרים נוספים בהם נדרשת גישת קריאה (נעילה אקסקלוסיבית) הוא כאמור בפעולות שהבנק מבצע בעצמו, כלומר הדפסת מצב הבנק למסך (כולל מצב החשבונות) + גביית עמלות, מנימוקים שהוסברו כנ"ל.
3. באובייקט ה-Log אליו רושם הבנק את כל פעולותיו. מכיוון ש-N כספומטים שונים מבצעים פעולות שעל הבנק לדווח עליהן, יש דרישה להחזיק מנעול שייגן על ה-Log מפני קריאות לא מסונכרנות שעלולות ליצור פלט לא ברור.
4. במונה המשותף בבנק הסופר כמה כספומטים סיימו את פעולתם. מונה זה משונה (גדל ב-1) כל פעם שכספומט מסיים פעולה, וההודעה על כך נשלחת לבנק מהכספומט עצמו, כלומר יש מצב שכמה חוטים יסיימו פעולה בו זמנית ולכן יש צורך להחזיק גם כאן מנעול שיאפשר כתיבה נכונה למונה.

המחלקות השונות במערכת (הסבר היררכיית המלבנים):

1. **מחלקת ה-System:** מאתחלת את מנהל הכספומטים (שיוצר N כספומטים) ומאתחלת את הבנק.
בפונקציית ה-Main של ה-System יאותחלו וירוצו 2 Threads חדשים שיריצו את ה-Main של הבנק ומנהל הכספומטים.
מחזיק פוינטר לאובייקט בנק ופוינטר לאובייקט מנהל הכספומטים.

2. **מחלקת מנהל הכספומטים (ATM_manager):** מאתחלת N כספומטים שונים.
בפונקציית ה-Main של ATM_manager יאותחלו וירוצו N Threads חדשים שבכל אחד מהם ירוץ ה-Main של כספומט יחיד (כלומר, N Threads שונים ל-N כספומטים שונים).
מחזיק vector של פוינטרים לאובייקט ATM המהווים את כל הכספומטים הפועלים במערכת.

3. **מחלקת הכספומט (ATM):** קוראת פקודות מקובץ נתון ומבקשת מהבנק לבצע פעולות בהתאם.

הכספומט מבצע את פקודותיו דרך פונקציית ה-Main שלו ושם גם שולח את הבקשות שלו לבנק. לאחר כל פעולה הכספומט ישן 100 מילי שניות.
המחלקה מחזיקה את תוכן הקובץ כולו (*) ממנו היא קוראת את הפקודות, רפרנס לאובייקט הבנק אליו היא שולחת את הבקשות לביצוע ומספר ייחודי המאפשר זיהוי של הכספומט.

(*)הערה: מכיוון ש-getline גורם לבעיות סינכרוניזציה עם כתיבה ל-cout (ע"י הבנק), ה-ATM קורא את הקובץ כולו ומבצע שבירה ל-tokens של כל שורות הקובץ עוד בבנאי של המחלקה, לפני שנוצרים החוטים של הבנק (כלומר, באתחול שמבצע ATM_manager)

4. **<u>מחלקת הבנק (Bank):</u>** מרכזת את כל המידע הדרוש על החשבונות הקיימים במערכת. בנוסף, מחלקת הבנק גובה עמלות מהחשבונות בהם היא מחזיקה ומדפיסה את מצב הבנק ל-stdout (**).

המחלקה מריצה שני Threads (בתוך ריצת Main של המחלקה). הראשון מבצע גבייה של עמלות מחשבונות הבנק, השני מדפיס את מצב הבנק ל-stdout.
גביית עמלות תקרה כל 3 שניות, הדפסת מצב הבנק תקרה כל חצי שנייה.

המחלקה מחזיקה:

a. vector (גישה אקראית לאיברים) של פוינטרים ל-BankAccount
b. מנעול הצמוד לווקטור הנ"ל
c. Logger (Thread-safe) אליו כותב הבנק את כל האירועים הקורים בו
d. ייתרת הבנק
e. מונה (Thread-safe) שסופר את כמות הכספומטים שסיימו את פעולתם

(**) הערה: הפונקצייה אשר מבצעת את הדפסת מצב הבנק ל-stdout היא הפונקצייה היחידה במערכת שכותבת ל-stdout.

5. **<u>מחלקת חשבון בנק (BankAccount):</u>** מייצגת חשבון בנק יחיד.
המחלקה מאפשרת משיכה, הפקדה ובירור יתרה. כל אחת מהפעולות הן Thread-safe הודות למנעול rwlock שמחזיקה המחלקה.

המחלקה מחזיקה:

a. יתרת החשבון
b. מספר החשבון
c. סיסמת החשבון
d. מנעול rwlock שמאפשר סינכרוניזציה בין threads שונים והופך את המחלקה ל-Thread-safe

6. **<u>מחלקת Logger:</u>** מחלקת קובץ המאפשר כתיבה מסונכרנת (Thread-safe) בין חוטים. מופע של המחלקה מוחזק על ידי הבנק, וכותב לקובץ log.txt בצורה בטוחה.

המחלקה מחזיקה:

a. אובייקט ofstream – כותב לקובץ log.txt
b. מנעול rwlock שמאפשר כתיבה מסונכרנת בין threads שונים והופך את הכתיבה לקובץ ל-Thread-safe.

7. **<u>מחלקת rwlock:</u>** ראה "מנגנון קוראים כותבים".

8. **<u>מחלקת thread_safe_counter:</u>** מחלקה פנימית של הבנק. סופר את כמות ה-ATM שסיימו ביצוע. המחלקה הינה Thread-safe הודות לשימוש במנעול rwlock שמגן על המונה הפנימי. המונה מאותחל ל-0, את הגבול ניתן לקבוע על ידי שינוי

   את המונה אפשר לשנות על ידי מימוש של אופרטור++, בדיקת התנאי שאומר שכל הכספומטים שסיימו על ידי HasReachedTop.

   המחלקה כוללת:
   a. מונה – סופר את כמות הכספומטים שסיימו את פעולתם.
   b. גבול עליון – כמות הכספומטים במערכת. בעזרתו יקבע הבנק אם כל הכספומטים סיימו פעולתם.
   c. מנעול rwlock – מאפשר שינוי מסונכרן בין הכספומטים המסיימים בו זמנית. הופך את המחלקה ל-Thread-safe.