**תיעוד רטוב 2**

**אפיון התוכנית ומבנה כללי**

בתרגיל זה מימשנו מערכת של בנק וכספומטים, המאחסנת מידע לגבי חשבונות שונים ומאפשרת לבצע עליהם פעולות מוגדרות. בנוסף, מערכת הבנק מבצעת פעולות מתמשכות על החשבונות. כל הפעולות שמתרחשות בבנק מתבצעות תוך ניסיון למקביליות מקסימלית.

**דיאגרמת בלוקים**

Bank

bankMap

bankAccount

Exceptions

macroBank

(החיצים מסמנים מעבר של מידע).

זוהי דיאגרמה המתארת איך המודולים "מדברים" אחד עם השני. הנתונים המועברים בין המודולים כוללים למשל דרישה לשדות פנימיים של חשבון (כמו האם הוא קפוא, מהי הסיסמא וכו') ולעיתים גם פעולות (כך למשל- bankMap עוטף את הפעולות שמבצע החשבון, כמו למשל getBalance).

דיאגרמה נוספת תתאר את אופן התנהלות הבנק:

ATM #n

Bank

Account #1

Account #n

ATM #1

…

…

כאן החצים מסמנים את ההיררכיה: הבנק יוצר כספומטים, שבתורם יוצרים חשבונות. לבנק עדיין יש אפשרות שליטה בחשבונות.

**מחלקות ומבני נתונים**

1. bankAccount- חשבון בנק:

זוהי מחלקה שמייצגת כל אחד מחשבונות הבנק של המערכת.

היא כוללת את השדות:

1. מזהה החשבון.
2. סיסמת החשבון.
3. מאזן נוכחי.
4. מאפיינים נוספים הקשורים לסנכרון, אותם נגדיר ועליהם נפרט בהמשך.
5. bankMap- מפת חשבונות בנקים:

זוהי מחלקה שמייצגת למעשה את הבנק שלנו. מבנה הנתונים אותו בחרנו הוא מפה, שכן באופן טבעי הוא נראה לנו מתאים ביותר- מקשר בין מספר חשבון שהוא המזהה לבין האובייקט עצמו שהוא חשבון בנק. המחלקה הזו כוללת שדה אחד בלבד שנקרא innerMap ומחזיק בתוכו מפה כפי שתיארנו קודם.

1. Exceptions- חריגות:

אלו הן מחלקות שמשמשות אותנו כדי לזהות טעויות ובעיות בעת השימוש בבנק ולטפל בהן.

1. ATM- כספומט:

זוהי מחלקה שמייצגת כל אחד מהכספומטים של הבנק.

היא כוללת את השדות:

1. מזהה הכספומט.
2. רשימת הפעולות של הכספומט לבצע.

**הסבר על הקבצים השונים של הפרויקט**

Bank.cpp-

זהו קובץ המעטפת, הקובץ הראשי אשר מריץ את כל הבנק. הוא קולט את הפקודה להפעלת הבנק, מבצע בדיקות תקינות ויוצר את הכספומטים, כמו גם את התהליכים שרצים ומאפשרים לנו לקחת עמלה מכל החשבונות ולהדפיס את מצב הבנק. קובץ זה אחראי גם על סנכרון, כפי שיפורט בהמשך.

bankMap.h ו- bankMap.cpp-

זהו "שלב ביניים" בגישה בין הכספומטים לבין החשבונות עצמם. כאן ממומשות פונקציות מעטפת לכל הפעולות הנדרשות בתרגיל, כמו גם פעולות נוספות שנדרשות.

bankAccount.h ו- bankAccount.cpp-

אלו הם קבצים שמגדירים את חשבון הבנק וממשים את כל הפעולות הנדרשות כמו גם פעולות נוספות. כאן מבוצעת סינכרוניזציה ברמת החשבון, כפי שיפורט בהמשך.

Expceptions.h -

קובץ עזר שמתאר חריגות מסוגים שונים שיכולות לקרות במהלך ריצת התוכנית. כפי שנהוג ב- CPP, קובץ זה מאגד את החריגות שיכולות להיזרק.

MacroBank.h -

קובץ עזר המגדיר מאקרואים לפעולות שאנחנו מבצעים פעמים רבות (כמו הדפסה לקובץ לוג וכו') על מנת להגביר את הקריאות של הקוד.

**הסבר על סינכרוניזציה בתרגיל שלנו**

נתאר את האופן בו מימשנו את המקביליות והסינכרון בין חוטים בתוכנית.

bankAccount

בקבצים אלה רצינו לממש מקביליות בעת פניה לאותו חשבון, אך יחד עם זאת להגן על החשבון מפני פעולה של חוטים במקביל בחלקים קריטיים.

לשם כך הגדרנו מנעולים לקריאה וכתיבה של קטע קריטי ולקריאה וכתיבה של מצב "קפוא". בכל אחת מהפונקציות שרלוונטיות להתעסקות מפורשת עם מצב חשבון או עם מצב קיפאון, מימשנו את הפתרון לבעיית קוראים כותבים שהוצגה בכיתה. כך אפשרנו מקביליות בעת קריאה, אך יחד עם זאת הגנו על החשבון שלנו מפני טעויות.

עבור העברה- השתמשנו בשיטת קוראים כותבים גם כן. את 2 החשבונות המעורבים נועלים לשינוי וקריאת היתרה, ונועלים לשינוי סטטוס הקיפאון. משחררים את המנעולים רק לאחר השינוי הרצוי בשני החשבונות (או כישלון מסיבה כלשהי).

קיימים 2 מנעולים לכל חשבון בנק בשיטת קוראים כותבים. מנעול יתרת החשבון ומנעול קיפאון החשבון.

קריאת סטטוס הקיפאון בחשבון מבוצע ע"י הפונקציות (תוך שימוש בקוראים כותבים):

* isAccountFrozen()- האם החשבון קפוא.
* withrawMoney(int withrawSum)- משיכת כסף מהחשבון. הססטוס נקרא על מנת לא לבצע משיכה כאשר החשבון קפוא. הפונקציה תמשיך "לקרוא" את סטטוס קיפאון החשבון עד לסיום פעולת שינוי היתרה על מנת שלא ישתנה הסטטוס במהלך הפעולה.
* depositMoney(int depositSum)- הפקדת כסף להחשבון. הססטוס נקרא על מנת לא לבצע הפקדה כאשר החשבון קפוא. הפונקציה תמשיך "לקרוא" את סטטוס קיפאון החשבון עד לסיום פעולת שינוי היתרה על מנת שלא ישתנה הסטטוס במהלך הפעולה.

שינוי סטטוס הקיפאון בחשבון מבוצע ע"י הפונקציות (תוך שימוש בקוראים כותבים):

* freeze()
* unFreeze()

קריאת יתרת החשבון מבוצע ע"י הפונקציות (תוך שימוש בקוראים כותבים):

* getBalance()
* getBalanceNoSleep()- בשונה מהקודמת, מבוצע ללא שינה (בשביל פונקציות של הבנק).
* printAccount()- יתרת הבנק נשמרת על מנת להדפיס פרטים על החשבון.

שינוי יתרת החשבון מבוצע ע"י הפונקציות (תוך שימוש בקוראים כותבים):

* withrawMoney(int withrawSum)
* withrawMoneyForCommission(int percentage)- בשונה מהקודמת, מבוצע ללא שינה וללא בדיקת סטטוס קיפאון (בשביל לקיחת עמלה מחשבון).
* depositMoney(int depositSum)

פעולת נעילה ושחרור עבור העברה בין חשבונות (תוך שימוש בקוראים כותבים):

* lockForTransfer ()- זוהי למעשה פונקציה בה משתמשים בעת העברה בין חשבונות על מנת לנעול את האפשרות לבדוק יתרה או לשנות אותה, ולבדוק את סטטוס הקיפאון עד סוף ההעברה. בפקודה העברה ינעלו 2 החשבונות הרלוונטיים במהלך ביצוע כל ההעברה.

Bank

בקובץ זה רצינו לממש מקביליות של הכספומטים (אשר מיוצגים על ידי חוטים), יחד עם החוט שמייצג הדפסה והחוט שמייצג לקיחת עמלה. ידענו כי בעת התעסקות עם אותו חשבון אנחנו מוגנים- כפי שתיארנו קודם.

יחד עם זאת, עדיין נדרשנו להגן על כתיבה במקביל לקובץ הלוג, על האפשרות לפתיחת חשבון במקביל על ידי שני חוטים שונים ועל מאזן הבנק.

לשם כך הגדרנו מנעול לקובץ הלוג, מנעול למפת החשבונות כולה ומנעול למאזן הבנק.

מנעול קובץ הלוג ננעל כאשר חוט מסוים כותב לקובץ הלוג ומשתחרר כאשר הוא מסיים.

מנעול המפה ננעל אך ורק בעת פתיחת חשבון ובדיקה האם חשבון קיים במפה (במהלך פעולות ATM), כדי למנוע את המצב (יחסית נדיר) של פתיחת אותו חשבון על ידי שני כספומטים שונים באותו זמן. לא נעלנו אותו בשום מקרה אחר כדי למנוע פגיעה במקביליות.

מנעול מאזן הבנק ננעל לאחר שביצענו פעולת לקיחת עמלה ואנחנו רוצים לעדכן את המאזן הכולל של הבנק.

**הערות ספציפיות למקרים מסוימים אשר לא הוגדרו באופן מוחלט בתרגיל**

* הנחנו שאין צורך לחכות שנייה עבור פעולות הקפאה והפשרה (לא מצויין אחרת).
* אם חשבון קפוא מקבלת פקודת הקפאה, או חשבון לא מוקפא מקבלת פקודה הפשרה, הפעולה מצליחה.
* ניתן לברר יתרה של חשבון קפוא.
* לא ניתן לבצע העברה בין חשבונות אם אחד מהם קפוא.
* הבנק יכול לקחת עמלה מחשבונות קפואים.
* בעת פקודה העברה בין חשבונות לא ניתן לבצע אף פעולה אחרת על החשבונות המעורבים (ולהפך).
* בעת הדפסת שורות הסטטוס ע"י הבנק, כל חשבון הוא "עצמאי". כלומר כל חשבון מודפס באופן בלתי תלוי בשאר החשבונות ומצב המנעולים שלהם.
* sleep(1) מבוצע בתוך הפונקציות של bankAccount.cpp עבור משיכה, בירור יתרה, הפקדה.
* עבור פתיחת חשבון sleep(1) מבוצע ב-Bank.cpp בתוך נעילת המנעול של המפה.
* עבור העברה בין חשבונות sleep(1) מבוצע ב-bankMap.cpp לאחר נעילת 2 החשבונות הרלוונטיים.
* כאשר קורים דברים שאינם חוקיים כגון קלט לא תקין של קבצים, חריגה מגודל int וכו' התוכנית מפסיקה מיידית.