

פרוייקט מסכם – קורס מורים מובילים תשס"ח

סניף בנק - BankBranch

הקדמה

הפרוייקט שלפניכם הוא פרויקט מסכם ליחידה. ניתן לתת אותו לכיתות לאחר לימוד נושא הרשימה ואף מאוחר יותר.

ההבדל בין משימה נלווית לבין פרויקט מסכם – משמעותי:

1. המשימה הנלווית משמשת כתרגיל מתמשך המתלווה לרוב פרקי הלימוד ומאפשר לתלמיד תרגול החומר החדש בכל פרק תוך הבנת השיפור והכלים החדשים שנלמדו בו. כאשר התלמיד יכול לבחור טיפוס אוסף חדש לרשימה כיתתית או עבור ספר הטלפונים אותם מימש קודם לכן, ולראות שבכך שיפר את יעילות המימוש או את קלות התכנות – הוא מבין באופן מתפתח את הידע החדש שרכש בפרקים.

מטרת הפרוייקט המסכם שונה. פרויקט זה מעלה בעיה שלמה אם כי לא גדולה, ומציע לתלמידים לבחור את כלי הייצוג והמימוש עבורה מתוך האוסף הקיים בארגז הכלים שלו שנבנה במהלך לימוד כל היחידה. הדיונים סביב הפרוייקט נוגעים בנקודות רבות שנלמדו לכל אורך היחידה ומאפשרים בחינה מסכמת של כל החומר הנלמד.

2. הפרוייקט מאפשר להבין ולו במעט את הרעיון של "עיצוב תוכנה" כאשר התלמיד מתבקש לעצב מערכת תוכנה (קטנה אמנם), משלב ניתוח הדרישות והמטרות תוך התחשבות בצורך התחזוקה והפעילות השוטפת על המערכת. הפעילות על הפרוייקט יכולה אולי לתת לתלמיד המחשה מצומצמת של מערכת השיקולים המלווים פיתוח מערכת תוכנה.

גורם הזמן העומד לרשות המורה – יקבע לאיזה עומק והרחבה ינצל המורה את הפרוייקט המסכם. הפרוייקט מודולרי בבסיסו וניתן להרחיבו או לצמצמו לרמת תרגיל מסכם פשוט למדי.

הפרוייקט שלפניכם מכיל חמש משימות מודרנות לתלמידים. כמו כן משולבות בו הצעות, הנחיות ודיונים שהם בבחינת המדריך למורה. בחרו את השימוש שלכם בפרוייקט בכל פעם מחדש, בהתאמה לכל כיתה, על פי ההנחיות השונות והדפיסו דפי משימה מתאימים.

פרוייקט זה פותח על ידי קבוצת המורים המובילים, שהשתתפה בשנת הלימודים תשס"ח – 2007/8 בקורס מורים מובילים "עיצוב תוכנה מבוסס עצמים" שארגן מרכז המורים הארצי למדעי המחשב והנחה צוות הפיתוח של היחידה.

קבוצה זו ביצעה, דנה, שיפרה ועיצבה את צורתו הסופית של הפרוייקט כפי שהיא מוצגת לפניכם. תודתנו לכל המשתתפים על השקעתם ועזרתם.

מה3נה

חלק א

ברצוננו להגדיר מחלקה בשם BankBranch המגדירה סניף בנק. אנו נתמקד רק בחלק מפעילות הסניף והוא: **ניהול חשבונות בנק עבור לקוחות.**

המטרה:

הגדרת הממשק של המחלקה BankBranch.

דיונים ראשוניים:

1. מהם הנתונים שיש לשמור בסניף בנק?
2. מיהן היישויות המשתתפות בפעילות סניף שכזה?
3. האם אנו זקוקים למחלקות שיגדירו יישויות נוספות או שהיישויות יכולות להיות מזוהות רק על פי נתונים בודדים? (אדם, לקוח, חשבון בנק...)
4. או בכיוון מעט שונה: איך מזוהה לקוח (על פי שמו? על פי מספר תעודת זהות?) הצגת האופציה להגדיר לקוח שמזוהה על פי מספר תעודת זהות ומחזיק את כל חשבונותיו (שבהם לא מופיע מספר ת.ז. שלו).
5. משימה: הגדרת הנתונים הנשמרים עבור כל יישות והפעולות שמוגדרות על נתונים אלו. רצוי בעזרת UML.
6. איך נשמרים חשבונות הבנק גם לאור שיקולי יעילות: האם יש עניין לרכז עבור כל לקוח את חשבונותיו או שאפשר לשמור את כל החשבונות "בתפזורת" כך שכל חשבון יכיל את פרטי בעליו? האם יש הבדל מבחינת יעילויות (של חיפוש וטיפול בחשבונות) לאופן בו נשמרים החשבונות?
7. אם הוחלט לייצג "חשבון בנק" ובו פעולות כמו משיכה והפקדה, יש לדון מה משמעות הכפילות הנוצרת בין פעולות אלו של חשבון לפעולות של סניף הבנק (רצוי למנוע כפילות שידועה למשתמש החיצוני וכך למנוע את המצב שנוצר ברשימה – שבה פעולות החוליה נשארו חשופות). האם משמעות העניין היא שאין לכלול פעולות שכאלה בטיפוס "חשבון בנק"?
8. מספר חשבון: האם נקבע על ידי הטיפוס "חשבון בנק" (אם בכלל הוחלט להגדיר אחד כזה) או על ידי סניף הבנק? משמעות התכונה הזו והבנה שלמרות שהיא תכונה של החשבון היא צריכה להיקבע על ידי הסניף!
9. מהן המגבלות הקיימות לגבי הפעילות בסניף (האם יש הגבלה על מספר החשבונות שמנהל סניף? האם יש הגבלה על מספר החשבונות שמנהל לקוח? מה המשמעות של התשובות לשאלות אלו לגבי בחירה במבני נתונים או בטיפוסי נתונים מתאימים לצורך ייצוג היישויות ומימושן?)

מסקנות חשובות:

1. אין דרך מידול מוחלטת ויחידה כל זמן שהסיפור והמטרה לא מוגדרים באופן סגור (מאד יכול להיות שגם כשהסיפור סגור היטב המידול אינו יחידאי). המטרה שהוצגה בחלק א אינה סגורה ולכן יתכנו כיווני מידול שונים.
2. אין הגבלה על מספר הלקוחות החברים בסניף הבנק.
3. אין הגבלה על מספר החשבונות שיכול לקוח לפתוח.
4. מסקנת שתי הנקודות הקודמות היא שיש צורך להשתמש בטיפוסי אוסף דינמיים ובלתי מוגבלים (רשימות) לצורך הייצוגים. ניתן לשקול שימוש במערך ולהבין שזהו שימוש מסורבל ויקר.
5. אם יש הבחנה בין אדם Person, ללקוח בנק Customer – מה המשמעות של הבדל זה (ניתן לחשוב על אדם כעל תעודת זהות: אוסף המזהה את האדם במקומות שונים ולצרכים שונים והוא הבסיס לרישומו לכל צורך פרטי. על המשמעות של לקוח בנק ניתן לחשוב כצורך פרטי שבסיסו זיהוי האדם בעזרת תעודת הזהות שלו והמשכו בתוספת תכונות המשמעותיות ליישום המבוקש).
6. המשמעות של סגירת חשבון: האם הוא נמחק בפועל או "רק" נסגר. משמעות הדבר לגבי אופן הייצוג של חשבון. המשמעות של החלטה זו תבחן בחלק ד שם נעסוק ביכולת לברר טעויות ואי בהירויות לגבי פעולות שנעשו בעבר על פי דו"ח תנועות של החשבון. בשלב זה **אין צורך לעסוק בכך** אם הנקודה לא הועלתה על ידי הכיתה. הצורך והיכולת לחזור ולפתוח מידול שנעשה בעבר ולשפרו לאור צרכים המתבררים בהמשך המשימה הם חלק ממהותו של פרויקט תכנותי שלם ויבהירו את רעיון התחזוקה של מערכת. אין להקדים את המאוחר. במידה ולא עלה נושא זה בכיתה יש להבליעו או למחקו כליל מתוך הממשק (לבטל כל התייחסות לחשבון "פעילי"), עד חלק ד של הפרויקט.

אפשרויות:

- צורת השימוש במשימה זו תלויה ברמת התלמידים, מטרות מקומיות, זמן מוקצב ועוד. על פי כל אלו יחליט המורה אם להתחיל ממידול ראשוני חופשי לחלוטין או מהגדרות מדויקות.**
- ניתן לחלק את הממשק המצורף ולחסוך את הדיון הראשוני. גם במקרה שכזה רצוי להתייחס לפחות לחלק מהנקודות לדיון כדי לוודא שהתלמידים מבינים שנעשתה כאן בחירה מושכלת בין אפשרויות שונות ולמה היא התקבלה כמו שהתקבלה.
 - יש לעבור יחד עם התלמידים על הפעולות שבממשק ולהבין אותן לאשורן (כולל משמעות הפרמטרים של הפעולות וערכי ההחזרה השונים).
 - ניתן לחלק ממשק חלקי ובו חסרים קטעים משמעותיים לדיון – ולמקד את הדיון רק סביב חלקים חסרים אלו. (ניתן למחוק את ההתייחסויות לנושא ה"פעילות" של החשבון – עד חלק ד של הפרויקט).

חלק ב

המטרה:

כתיבת המחלקה BankBranch, על פי ממשק שהתגבש בחלק א או על פי הממשק המצורף.

המחלקה BankBranch

מגדירה סניף בנק שמנהל אוסף חשבונות בנק. בכל חשבון בנק נשמרים הנתונים: מספר-חשבון (המזהה את החשבון באופן ייחודי), תעודת הזהות של בעל החשבון ויתרת הכסף בחשבון. לכל החשבונות יש מסגרת אשראי קבועה אחת בגובה 5000 שקלים*.

BankBranch()	הפעולה בונה סניף בנק חדש
int openAccount (String customerID)	הפעולה פותחת חשבון בנק חדש בסניף, שמספרו נקבע על ידי המחלקה בצורה אוטומטית-סידרתית החל מהמספר 1000, עבור הלקוח שמספר ת.ז. שלו הוא customerID. יתרת הכסף בחשבון היא 0 שקלים. הפעולה מחזירה את מספר החשבון שנפתח
void closeAccount (int accountNum)	הפעולה סוגרת את חשבון הבנק שמספרו נתון. הנחה: קיים חשבון פעיל שמספרו accountNum
void deposit (int accountNum, double amount)	הפעולה מפקידה amount שקלים לחשבון הבנק שמספרו accountNum. הנחות: קיים חשבון פעיל שמספרו accountNum. amount גדול מאפס
boolean withdrawal (int accountNum, double amount)	הפעולה מושכת amount שקלים מהחשבון שמספרו accountNum. הפעולה תחזיר true אם המשיכה לא גרמה לחריגה מהמסגרת*, אחרת תחזיר false והמשיכה לא תתבצע. הנחות: קיים חשבון פעיל שמספרו accountNum. Amount גדול מאפס
double getBalance (int accountNum)	הפעולה מחזירה את היתרה ב-accountNum. הנחה: קיים חשבון שמספרו accountNum
String getOwnerID (int accountNum)	הפעולה מחזירה את מספר ת.ז. של בעל החשבון שמספרו accountNum. הנחה: קיים חשבון שמספרו accountNum
boolean accountExists (int accountNum)	הפעולה מחזירה true אם accountNum קיים בסניף הבנק, אחרת תחזיר false

String toString()	<p>הפעולה מחזירה מחרוזת המתארת את סניף הבנק :</p> <pre><accNum1>, <ownerID1>, <balance1>, <active/not active> <accNum2>, <ownerID2>, <balance2>, <active/not active> :</pre>
-------------------	--

מה עליכם לעשות?

1. כתבו את המחלקה BankBranch ותעדו אותה. ניתן להשתמש במחלקות עזר נוספות אך אז יש לכתוב אותן במלואן כולל תיעוד.
2. כתבו תוכנית בדיקה TestBankBranch.java.

הערות:

1. יש לוודא שהנושא של מסגרת אשראי וחריגה מהמסגרת מובנים ומוכרים לתלמידים. אחרת יקשה עליהם לממש את הפעולות כראוי*.
2. הממשק המוצע אינו מכיל את הפעולה :

List<Integer> getAllAccountsNums (String customerID)

1. המחזירה את אוסף כל מספרי החשבונות שמחזיק לקוח מסוים. בכוונה איננו כוללים פעולה זו בממשק כדי לאפשר את הדיון על הצורך בקיום פעולה שכזו בחלק ג. אם תלמידיכם העלו את הצורך בהגדרת הפעולה אין למנוע אותם מלהגדירה ולממשה. הדיון בחלק ג יונחה בהתאמה.
2. הגדרת התכונות מתרגלת שימוש בסוגי איברים שונים: מסגרת האשראי היא תכונת מחלקה ולכן היא סטטית. כמו כן עליה להיות מוגדרת כ-**final**. מספר החשבון גם הוא תכונת מחלקה סטטית.
3. האוסף המדובר הוא אוסף ספציפי, כלומר הוא אינו אוסף גנרי שיכול לשמש לאחסון של איברים שונים אלא רק חשבונות בנק.
4. שיקולים תכנותיים או אפשרות לדיון בכיתה: האם יש הצדקה להגדיר פעולת חיפוש פנימית-פרטית לצורך כל הפעולות המתבססות על חיפושים? מה החיסכון והתועלת בהגדרת פעולה שכזו אם הם קיימים בכלל?

אפשרויות:

ניתן לחלק לתלמידים תוכנית בדיקה מוכנה ולחסוך בזמן.

חלק ג

המטרה:

בדיקת שלמותו של הממשק הקיים לאור צרכים של המערכת הבנקאית.

מה עליכם לעשות?

הוסיפו לתוכנית הבדיקה את הפעולה הבאה:

<pre>static void printNegBalance (BankBranch bankBr)</pre>	<p>הפעולה תדפיס את כל החשבונות בסניף bankBr שיתרתם שלילית. כל חשבון יודפס בשורה נפרדת בה יופיעו הנתונים:</p> <p>מספר החשבון, מספר ת.ז. של בעל החשבון ויתרת הכסף בחשבון</p>
--	--

דיונים מתקדמים:

1. על פי דרישת השאלה - הפעולה המבוקשת צריכה להיות מוגדרת במחלקת הבדיקה, כלומר היא פעולה חיצונית למחלקה סניף-בנק. אם במחלקה BankBranch לא מוגדרת הפעולה:
`List<Integer> getAllAccountsNums (String customerID)`
אין דרך לממש את הפעולה `printNegBalance(...)`. פתחו את הדיון על כך עם תלמידים.
2. בחנו עם התלמידים את האפשרות וההיגיון שבהוספת הפעולה `printNegBalance(...)` לממשק המחלקה "סניף בנק".
3. יש להחליט עם הכיתה על קו פעולה מוסכם ולפיו יש לממש את הפעולה (לא בהכרח סטטית וחיצונית).

חלק ד

המטרה:

הרחבת הפרויקט וקירובו לעולם הממשי.

כדי לדמות חשבון בנק ממשי ברצוננו לשמור את ההיסטוריה של פעולות ההפקדה והמשיכה שנעשו בכל חשבון כך שניתן יהיה להפיק דו"ח תנועות עבור כל חשבון המתנהל בבנק.

דיונים מקדימים:

1. מהם השינויים והתוספות שיש לבצע בייצוג המחלקות הקיימות ומימושן כדי שניתן יהיה להפיק את דו"חות התנועות. האם יש צורך להגדיר יישויות חדשות לצורך המשימה הנוכחית?
2. בשלב זה יכול להתעורר צורך לחזור להגדרת פעולת הממשק האחראית על סגירת חשבון של לקוח ולשנותה. אם חשבון לא פעיל - נמחק בפועל מרשימת החשבונות, לא ניתן לקבל עבורו דו"ח תנועות ואם כך לא ניתן יהיה לשחזר מהלכים שנעשו בו בעבר. אם חשבון שנסגר רק סומן כ-"לא פעיל", ניתן להציג דו"חות מתאימים לגביו אך יש לכך מחיר. סריקת רשימת החשבונות של לקוח יעילה פחות אם שמורים בה גם חשבונות לא פעילים, שכן לצורך פעולות שונות יש לחזור ולבדוק בזמן סריקה, לגבי כל חשבון אם הוא פעיל או לא. עם הוספת התכונה המתארת את מצב החשבון – יכול להתעורר הצורך להוסיף לממשק של חשבון בנק/סניף בנק, פעולה המחזירה את מצבו של חשבון שמספרו נתון.
3. ניתן לשקול (ולשנות ייצוגים ומימושים בהתאמה) – שמירה נפרדת של רשימת החשבונות הפעילים והחשבונות הבלתי פעילים (=סגורים).

מה עליכם לעשות?

בצעו את השינויים הנדרשים במחלקה BankBranch כך שבעת זימון הפעולה toString() תוחזר מחרוזת המתארת את פרטי כל החשבונות בסניף הבנק (כפי שהיה קודם), ובנוסף, עבור כל חשבון, תפורט כל היסטורית פעולות המשיכה וההפקדה (דו"ח תנועות), שבוצעו בו מאז שנפתח ועד לרגע זימון פעולה זו. סדר היסטורית הפעולות יופיע על פי סדר ביצוען.

הערות:

כדי להציג היסטוריה של פעולה יש לשמור עבור כל פעולה שלושה נתונים:

- התאריך בו התבצעה
 - סוג הפעולה (משיכה או הפקדה)
 - הסכום שעבורו נעשתה הפעולה
- הדו"ח יציג את היתרה בחשבון בתום כל פעולה המופיעה בו.

אפשרויות:

אוסף התנועות יכול להישמר כמחרוזת משורשרת. לצורך הצגה בלבד זוהי אפשרות קבילה אך אם צפוי צורך לבצע חיפושים והשוואות על הנתונים באוסף – הרי ששמירה שכזו אינה סבירה.

חלק ה

הערות והצעות מתקדמות:

1. התחושה החזקה הראשונית בעת העיסוק בסניף הבנק היא של עבודה עם רשימות שונות, אך ניתן לחשוב על ייצוג חלקים מן המשימה בעזרת מפה כאשר מספר תעודת זהות ומספר חשבון יכולים להיות מפתחות. השימוש במפה סביר שלא יקרה בשנת ההוראה הראשונה של החומר אך כדאי לשקול את השימוש בו בהמשך.
2. נבחן לדוגמה את השימוש במפה לצורך ייצוג אוסף החשבונות בסניף. ייצוג זה סביר מכמה סיבות עיקריות:
 - א. אין מגבלה על כמות החשבונות שיכול אדם לפתוח
 - ב. אין כפילות של חשבונות בעלי אותו מספר.
 - ג. אין חשיבות לסדר שמירת החשבונות באוסף.
3. הוספת הפעולה המתבקשת: `moneyTransfer(...)` האחראית על העברת כסף מחשבון לחשבון. היכן תוסף הפעולה?
4. הרחבת הדרישה לגבי התנועות הנשמרות עבור כל חשבון. ברצוננו להוסיף את היכולת לבטל פעולה אחרונה שהתבצעה בחשבון. הרחבה זו מאפשרת באופן "טבעי" שימוש במחסנית במסגרת הפרויקט.
5. ניתן להגדיר מספר סניפי בנק ולבצע פעולות על סניפי הבנק: למצוא איזה סניף הכי רווחי? (על פי סכום הכספים בכל היתרות או/וגם על פי מספר החשבונות שנפתחו בסניף).
6. ניתן להוסיף אילוץ נוסף המגדיר סדר של אוסף החשבונות. לדוגמה, ניתן לדרוש שהפעולה `toString()` תחזיר את כל החשבונות מסודרים בסדר עולה על פי ערך היתרה בהם. במצב זה על התלמיד להחליט אם הוא שומר את האוסף ממוין בכל שלב ושלב או שהוא מבצע מיון רק כאשר `toString()` מזומנת. ניתן לקיים דיון ביעילות שתי האפשרויות.