תרגיל בית מס' 5 (חובה). רשימות

- המועד האחרון להגשה ומספר הסטודנטים המקסימלי בקבוצה מופיע באתר הגשת העבודות.
- ההגשה דרך אתר הגשת העבודות בלבד.
- מותר להתייעץ וללמוד יחד עם החברים, אבל על כל קבוצת מגישים לכתוב ולהריץ את הקוד, ולערוך את התרגיל להגשה לבד. בשום פנים ואופן אין להעביר קבצי קוד או קבצי הגשה בין הקבוצות. העתקה אפילו של חלק מהתרגיל עלולה לפסול את התרגיל כולו. הגשת התרגיל מהווה הצהרה שהמגישים לא העתיקו מהקבצים של האחרים בתהליך הכנת התרגיל.
- חשוב לוודא שכל קבצי ההגשה עוברים קומפילציה ב-Eclipse. הפתרון שלא עובר קומפילציה לא יקבל ציון.
- יש לארוז את כל הקבצים של קוד המקור לקובץ Zip אחד ולהגיש דרך הגשת העבודות. **חוץ מהקבצים המפורטים בדרישות התרגיל אין לכלול בהגשה שום תיקיה או קובץ נוסף.**
- בתחילת כל קובץ עם קוד המקור אמור להופיע בלוק הערות עם מספר התרגיל, שמות ותעודות זהות של המגישים ושם הקובץ. דרישות לגבי סגנון הכתיבה של הקוד מופיעות בנספח בסוף תרגיל בית מס' 1. חשוב לעמוד בדרישות הסגנון כדי לקבל ציון מלא על התרגיל.
- עם שאלות ובירורים לגבי התרגיל נא לפנות דרך הפורום באתר הקורס. כך כל הסטודנטים של הכיתה יקבלו תועלת מהשאלות ומהתשובות. עם שאלות שלא קשורות לכל הכיתה נא לפנות דרך פניה למרצה באתר הקורס.

שאלה	קבצי ההגשה
1	1. `LargestInArray.java` – הקוד של התכנית 2. `LargestInArrayResults.txt` – פלט של 3 דוגמאות הרצה שונות
2	1. `Median.java` – הקוד של התכנית 2. `MedianResults.txt` – פלט של 3 דוגמאות הרצה שונות

3	1. Savings.java – הקוד של התכנית 2. SavingsResults.txt – פלט של 3 דוגמאות הרצה שונות
4	1. SchoolGradesList.java – הקוד של המחלקה SchoolGradesList 2. SchoolGradesListTester.java – הקוד של המחלקה SchoolGradesListTester 3. SchoolGradesListResults.txt – פלט של 3 דוגמאות ההרצה
5	1. BankAccount.java – הקוד של המחלקה BankAccount. 2. BankBranch.java – הקוד של המחלקה BankBranch. 3. BankBranchTester.java – הקוד של המחלקה BankBranchTester. 4. BankBranchTesterResults.txt – דוגמאות ההרצה של התכנית BankBranchTester.

בהצלחה רבה !

שאלה 1. (10 נק'). איבר מקסימלי במערך

כתבו תכנית `LargestInArray.java` שמקבלת מהמשתמש סדרה של מספרים ממשיים ('Q' מסמן סיום הקלט), שומרת אותם במערך, ולאחר מכאן מדפיסה את המערך תוך ציון של האיבר המקסימלי.

אם ברשימה יש כמה איברים מקסימליים, יש לסמן כל אחד מהם.

אם רשימת הערכים ריקה, יש להדפיס הודעה מתאימה.

דוגמאות ההרצה של התכנית:

```
Please enter integer values, Q to quit:
-10
-12
35
79
-1
0
56
79
-134
9
0
Q
       -10
       -12
        35
        79 <== largest value
        -1
         0
        56
        79 <== largest value
      -134
         9
         0
```

דוגמת ההרצה עבור רשימה ריקה של ערכים:

```
Please enter integer values, Q to quit:
Q
The list of values is empty
```

הנחיות:

הגדרת הרשימה של מספרים שלמים:

```
ArrayList<Integer> values = new ArrayList<Integer>();
```

קבצי ההגשה:

`LargestInArray.java` – הקוד של התכנית

`LargestInArrayResults.txt` – פלט של 3 דוגמאות הרצה שונות

שאלה 2. (10 נק'). מיון של רשימה ומציאת החציון

כתבו תכנית `Median.java` שמבצעת את הפעולות הבאות:

א. מקבלת מהמשתמש רשימה של מספרים שלמים, Q מסמן סיום הקלט.

ב. ממיינת את רשימת הערכים בסדר עולה.

ג. מדפיסה את האיבר המינימלי, המקסימלי ואת החציון. אם ברשימה מספר זוגי של איברים, למשל 10, אז בתור החציון אפשר לבחור את האיבר ה-5 או את האיבר ה-6.

ד. מדפיסה את רשימת הערכים (בסדר עולה) ומסמנת בהדפסה את החציון.

ה. אם רשימת הערכים ריקה, התכנית מדפיסה הודעה מתאימה.

דוגמת הרצה 1:

```
Please enter integer values, Q to quit:
-10
-12
35
79
-1
0
56
79
-134
9
88
90
Q
Min     =       -134
Max     =         90
Median  =         35
Sorted list with median :
      -134
       -12
       -10
        -1
         0
         9
        35 <== median
        56
        79
        79
        88
        90
```

דוגמת הרצה 2 (רשימה של ערך בודד):

```
Please enter integer values, Q to quit:
1
Q
Min     =          1
Max     =          1
Median  =          1
Sorted list with median :
                1 <== median
```

דוגמת הרצה 3 (רשימה של 2 ערכים):

```
Please enter integer values, Q to quit:
2
```

```
-1
Q
Min     =         -1
Max     =          2
Median  =          2
Sorted list with median :
        -1
         2 <== median
```

דוגמת הרצה 4 (רשימה של 3 ערכים):

```
Please enter integer values, Q to quit:
1
3
2
Q
Min     =          1
Max     =          3
Median  =          2
Sorted list with median :
         1
         2 <== median
         3
```

דוגמת הרצה 5 (רשימה ריקה):

```
Please enter integer values, Q to quit:
Q
The list of values is empty
```

הנחיות:

א. למיון הרשימה `values` ניתן להשתמש במתודה `values.sort(null)`.

ב. אחרי מיון הרשימה, באיזה אינדקס נמצא האיבר המינימלי ? המקסימלי ? החציון ?

קבצי ההגשה:

`Median.java` – הקוד של התכנית

`MedianResults.txt` – פלט של דוגמת ההרצה

שאלה 3. (10 נק'). תכנית חסכון

כתבו תכנית `Savings.java` שמבצעת את הפעולות הבאות:

א. מקבלת מהמשתמש את הסכום של הפקדה חד-פעמית, ריבית שנתית ומספר השנים של תכנית החיסכון.

ב. יוצרת רשימה `ArrayList<Double>` כך האיבר ה-`i` בתוך הרשימה שווה לערך החיסכון `i` שנים אחרי ההפקדה. (האיבר ה-0 ברשימה שווה לסכום ההפקדה.)

ג. מדפיסה את מספר השנים ואת הערך העתידי של החיסכון בצורה של טבלה כמו בדוגמה להלן:

```
Enter the deposit amount: 1000
Enter the interest rate (in %): 5
Enter the number of years: 10
Year     |    Amount
--------------------
      0 |   1000.00
      1 |   1050.00
      2 |   1102.50
      3 |   1157.63
      4 |   1215.51
      5 |   1276.28
      6 |   1340.10
      7 |   1407.10
      8 |   1477.46
      9 |   1551.33
     10 |   1628.89
```

הנחיות:

א. הפלט צריך להיות מיושר כמו בדוגמה לעיל, עם קו "|" שמפריד בין עמודת השנים לעמודת החיסכון. מומלץ להשתמש במתודה `System.out.printf`.

קבצי ההגשה:

`Savings.java` – הקוד של התכנית

`SavingsResults.txt` – פלט של 3 דוגמאות הרצה

שאלה 4. (20 נק'). סטטיסטיקה של ציונים

כתבו מחלקה `SchoolGradesList` מיועדת לניהול של רשימת הציונים בבית ספר. הציונים בבית ספר הם מספרים שלמים בטווח מ-0 עד 10.

המשתנים הפרטיים של המחלקה:

רשימת הציונים (הציונים הם מספרים שלמים מ-0 עד 10)	`ArrayList<Integer> gradesList`

על המחלקה לתמוך במתודות הבאות:

הוספת ציון grade לרשימת הציונים. אם הציון לא נמצא בטווח הנכון (בין 0 ל-10) המתודה מדפיסה הודעת שגיאה ולא מוסיפה את הציון לרשימה.	`public void addGrade(int grade)`
הדפסת ההתפלגות של הציונים. עבור כל ציון בטווח מ-0 עד 10 יש להדפיס את הציון, מספר הפעמים שהציון מופיע ברשימה (שכיחות), ואת ההיסטוגרמה שמציגה את השכיחות באופן חזותי. מספר הכוכביות בהסטוגרמה שווה לשכיחות של הציון.	`public void printGradesDistribution()`

דוגמה לפלט של המתודה `printGradesDistribution()` :

```
Print Grade List
Grade                Frequency            Histogram
0                    2                    **
1                    0
2                    0
3                    1                    *
4                    3                    ***
5                    10                   **********
6                    8                    ********
7                    18                   ******************
8                    25                   *************************
9                    13                   *************
10                   5                    *****
```

הדרכה:

א. כדי לחשב את השכיחות של הציונים, ניתן להשתמש ברשימת עזר `ArrayList<Integer> frequencies`. האביר ה-i של `frequencies` מייצג את השכיחות של הציון i ברשימת הציונים.

ב. יש לאתחל את הרשימה `frequencies` על ידי הוספה של 11 ערכים 0.

ג. ניתן לחשב את כל הערכים בתוך הרשימה `frequencies` תוך מעבר אחד על רשימת הציונים `gradesList`. כל פעם שרואים ציון i מעלים את הערך של האיבר ה-i ברשימת `frequencies` ב-1.

ד. אחרי החישוב של כל הערכים ברשימת `frequencies` ניתן להדפיס את ההיסטוגרמה על ידי לולאת for מקוננת.

כתבו מחלקה `SchoolGradesListTester` שמקבלת מהמשתמש רשמה של ציונים (Q לסיום הקלט) ומדפיסה את התפלגות הציונים, כולל ההיסטוגרמה.

קבצי ההגשה:

`SchoolGradesList.java` – הקוד של המחלקה `SchoolGradesList`

`SchoolGradesListTester.java` – הקוד של המחלקה `SchoolGradesListTester`

`SchoolGradesListResults.txt` – פלט של 3 דוגמאות ההרצה

שאלה 5. (50 נק'). סניף בנק

מחלקה BankAccount (10 נק'). כתבו מחלקה `BankAccount` שמתארת חשבון בנק. המחלקה מאפשרת להפקיד כסף לחשבון, למשוך כסף מהחשבון ולבדוק את היתרה בחשבון.

המשתנים הפרטיים של המחלקה:

מספר החשבון (מחרוזת)	`accountNumber`
שם הלקוח (מחרוזת)	`customerName`
יתרה בחשבון (מספר עשרוני)	`balance`
יתרה מינימלית (מספר עשרוני)	`minBalance`

המתודות של המחלקה:

בנאי שמקבל כפרמטרים את מספר החשבון, שם הלקוח, יתרת פתיחת החשבון ויתרה מינימלית.	`BankAccount(String accountNumber, String customerName, double initialBalance, double minBalance)`
הפקדה של סכום amount לחשבון. המתודה תמיד מחזירה true.	`public boolean deposit(double amount)`
משיכת סכום amount מהחשבון. אם אחרי המשיכה היתרה תהיה נמוכה מminBalance, המשיכה לא מתבצעת והמתודה מחזירה false. אם המשיכה הייתה מוצלחת המתודה מחזירה true.	`public boolean withdraw(double amount)`
המתודה מחזירה את מספר החשבון	`public String getAccountNumber()`
המתודה מחזירה את היתרה בחשבון	`public double getBalance()`

המתודה מחזירה מחרוזת עם כל פרטי החשבון. למשל:
`Account Number = 001`
`Customer Name = Alice`
`Balance = -10000.00`
`Minimal balance = -20000.00`
	`public String toString()`

מחלקה BankBranch (25 נק'). תבו באופן מלא מחלקה `BankBranch` שמתארת סניף של בנק.

המשתנים הפרטיים של המחלקה:

מספר הסניף (מחרוזת)	branchNumber
כתובת הסניף (מחרוזת)	branchAdress
רשימה (ArrayList) של חשבונות	accounts

המתודות של המחלקה:

בנאי שמקבל כפרמטרים את מספר הסניף ואת כתובת הסניף	`public BankBranch(String branchNumber, String branchAdress)`
הוספת חשבון account לרשימת החשבונות של הסניף	`public void addAccount(BankAccount account)`

המתודה מקבלת כפרמטר את מספר החשבון ומחזירה הפניה לחשבון אם הוא קיים, או null אם בסניף אין חשבון עם המספר הנתון.
 	
הקוד של המתודה מופיעה בשלמותו.
	`public BankAccount findAccount(String accountNumber)`
	`{`
	`for(BankAccount acc: accounts)`
	`{`
	`if(accountNumber.equals(acc.getAccountNumber()))`
	`{`
	`return acc;`
	`}`
	`}`
	`return null;`
	`}`

הפקדה של סכום amount לחשבון מספר accountNumber. אם החשבון לא קיים, המתודה מחזירה false. אחרת מתבצעת הפקדה והמתודה מחזירה true.	`public boolean deposit(String accountNumber, double amount)`
משיכת סכום amount מהחשבון מספר accountNumber. אם החשבון לא קיים או שהמשיכה נכשלה (בגלל מסגרת האשראי של החשבון) המתודה מחזירה false. אם המשיכה הצליחה, המתודה מחזירה true.	`public boolean withdraw(String accountNumber, double amount)`
העברת סכום amount מהחשבון מספר fromAccountNumber לחשבון מספר toAccountNumber. אם אחד מהחשבונות לא קיים, או שהעברה בלתי אפשרית בגלל מסגרת האשראי, ההעברה לא מתבצעת ומצב החשבונות נשאר ללא שינוי. במקרה זה המתודה מחזירה false. אם ההעברה הצליחה, המתודה מחזירה true.	`public boolean transfer(String fromAccountNumber, String toAccountNumber, double amount)`
המתודה מחזירה מחרוזת עם פרטי הסניף והפרטים של כל החשבונות. בסוף המחרוזת מופיע סך מספר	`public String toString()`

| החשבונות בסניף והיתרה הכוללת של כל החשבונות. למשל: | ```
----------
Branch number = 987, branch address = Havazelet 10 TA
----------
Accounts details:

Account Number  =        001
Customer Name   =      Alice
Balance         =  -10000.00
Minimal balance =  -20000.00

Account Number  =        002
Customer Name   =        Bob
Balance         =   20000.00
Minimal balance =  -20000.00

----------
Number of accounts: 2
Total balance:  10000.00
----------
```  |
| המתודה מחזירה מחרוזת עם פרטי הסניף ורשימת כל החשבונות שנמצאים במינוס (יתרה שלילית). בסוף המחרוזת מופיע סך מספר החשבונות במינוס והיתרה הכוללת של החשבונות במינוס. למשל: | ```
public String accountsInDebt()

----------
Branch number = 987, branch address = Havazelet 10 TA
----------
Accounts in debt:

Account Number  =        001
Customer Name   =      Alice
Balance         =  -10000.00
Minimal balance =  -20000.00

----------
Number of accounts in debt:                1
Total balance of accounts in debt: -10000.00
----------
```  |

תכנית BankBranchTester (5 נק'). כתבו מחלקה BankBranchTester שבודקת את כל המתודות של המחלקה BankBranch. התכנית מציגה למשתמש תפריט פעולות, ומבצעת פעולות בהתאם לבחירה. התבנית של התכנית (הקוד שמציג את התפרט ומקבל קלט מהמשתמש) מופיעה בהמשך.

דוגמאות ההרצה:

1. יצירת החשבון:

```
Please choose action:
  1 - create account
  2 - print account details
  3 - deposit
  4 - withdraw
  5 - transfer
  6 - print all accounts
  7 - print accounts in debt
 -1 - quit
Your choice : 1
Please enter the new account details:
Account number:             001
Customer name:              Alice
```

```
Initial balance:            0
Minimal allowed balance:    -10000
```

2. הצגת פרטי החשבון:

```
Please choose action:
  1 - create account
  2 - print account details
  3 - deposit
  4 - withdraw
  5 - transfer
  6 - print all accounts
  7 - print accounts in debt
 -1 - quit
Your choice : 2
Please enter the account number:001
Acount details:
Account Number  =        001
Customer Name   =      Alice
Balance         =       0.00
Minimal balance =  -10000.00
```

3. הפקדה:

```
Your choice : 3
Please enter the account number:001
Please enter amount to deposit:1000
Deposit of   1000.00 to account 001 succeeded.
```

4. הפקדה שנכשלה (כי החשבון לא קיים) :

```
Your choice : 3
Please enter the account number:002
Please enter amount to deposit:10000
Deposit of  10000.00 to account 002 failed.
```

5. משיכה מהחשבון:

```
Your choice : 4
Please enter the account number:001
Please enter amount to witdraw:10000
Withdraw of  10000.00 to account 001 succeeded.
```

6. משיכה שנכשלה (בגלל מסגרת האשראי) :

```
Your choice : 2
Please enter the account number:001
Acount details:
Account Number  =        001
Customer Name   =      Alice
Balance         =   -9000.00
Minimal balance =  -10000.00
```

```
Please choose action:
  1 - create account
  2 - print account details
  3 - deposit
  4 - withdraw
  5 - transfer
  6 - print all accounts
  7 - print accounts in debt
 -1 - quit
Your choice : 4
Please enter the account number:001
Please enter amount to witdraw:5000
Withdraw of   5000.00 to account 001 failed.
Please choose action:
  1 - create account
  2 - print account details
  3 - deposit
  4 - withdraw
  5 - transfer
  6 - print all accounts
  7 - print accounts in debt
 -1 - quit
Your choice : 2
Please enter the account number:001
Acount details:
Account Number  =        001
Customer Name   =      Alice
Balance         =   -9000.00
Minimal balance =  -10000.00
```

7. העברה בין החשבונות:

```
Your choice : 6
-----------------------------------------------------
Branch number = 987, branch address = Havazelet 10 TA
-----------------------------------------------------
Accounts details:

Account Number  =        001
Customer Name   =      Alice
Balance         =   46000.00
Minimal balance =  -10000.00

Account Number  =        002
Customer Name   =        Bob
Balance         =   -5000.00
Minimal balance =  -10000.00

-----------------------------------------------------
Number of accounts: 2
Total balance:  41000.00
-----------------------------------------------------

Please choose action:
  1 - create account
  2 - print account details
  3 - deposit
  4 - withdraw
  5 - transfer
```

```
  6 - print all accounts
  7 - print accounts in debt
 -1 - quit
Your choice : 5
Transfer from account : 001
Transfer to account   : 002
Amount to transfer    : 50000
Transfer of  50000.00 from account 001 to account 002 succeeded.
Please choose action:
  1 - create account
  2 - print account details
  3 - deposit
  4 - withdraw
  5 - transfer
  6 - print all accounts
  7 - print accounts in debt
 -1 - quit
Your choice : 6
-----------------------------------------------------
Branch number = 987, branch address = Havazelet 10 TA
-----------------------------------------------------
Accounts details:

Account Number  =        001
Customer Name   =      Alice
Balance         =   -4000.00
Minimal balance =  -10000.00

Account Number  =        002
Customer Name   =        Bob
Balance         =   45000.00
Minimal balance =  -10000.00

-----------------------------------------------------
Number of accounts: 2
Total balance:  41000.00
-----------------------------------------------------
```

8. העברה בין החשבונות שנכשלה (בגלל מסגרת האשראי):

```
Your choice : 6
-----------------------------------------------------
Branch number = 987, branch address = Havazelet 10 TA
-----------------------------------------------------
Accounts details:

Account Number  =        001
Customer Name   =      Alice
Balance         =   -4000.00
Minimal balance =  -10000.00

Account Number  =        002
Customer Name   =        Bob
Balance         =   45000.00
Minimal balance =  -10000.00

-----------------------------------------------------
```

```
Number of accounts: 2
Total balance:  41000.00
-----------------------------------------------------

Please choose action:
  1 - create account
  2 - print account details
  3 - deposit
  4 - withdraw
  5 - transfer
  6 - print all accounts
  7 - print accounts in debt
 -1 - quit
Your choice : 5
Transfer from account : 001
Transfer to account   : 002
Amount to transfer    : 10000
Transfer of  10000.00 from account 001 to account 002 failed.
Please choose action:
  1 - create account
  2 - print account details
  3 - deposit
  4 - withdraw
  5 - transfer
  6 - print all accounts
  7 - print accounts in debt
 -1 - quit
Your choice : 6
-----------------------------------------------------
Branch number = 987, branch address = Havazelet 10 TA
-----------------------------------------------------
Accounts details:

Account Number  =        001
Customer Name   =      Alice
Balance         =   -4000.00
Minimal balance =  -10000.00

Account Number  =        002
Customer Name   =        Bob
Balance         =   45000.00
Minimal balance =  -10000.00

-----------------------------------------------------
Number of accounts: 2
Total balance:  41000.00
-----------------------------------------------------
```

9. הדפסת הפרטים של כל החשבונות:

```
Your choice : 6
-----------------------------------------------------
Branch number = 987, branch address = Havazelet 10 TA
-----------------------------------------------------
Accounts details:

Account Number  =        001
Customer Name   =      Alice
Balance         =   -4000.00
```

```
Minimal balance =  -10000.00

Account Number  =        002
Customer Name   =        Bob
Balance         =   45000.00
Minimal balance =  -10000.00

-----------------------------------------------------
Number of accounts: 2
Total balance:  41000.00
-----------------------------------------------------
```

10. הדפסת פרטי החשבונות במינוס:

```
Your choice : 7
-----------------------------------------------------
Branch number = 987, branch address = Havazelet 10 TA
-----------------------------------------------------
Accounts in debt:

Account Number  =        001
Customer Name   =      Alice
Balance         =   -4000.00
Minimal balance =  -10000.00

Account Number  =        003
Customer Name   =      Cindy
Balance         =   -5000.00
Minimal balance =  -10000.00

-----------------------------------------------------
Number of accounts in debt:                2
Total balance of accounts in debt:  -9000.00
-----------------------------------------------------
```

בדיקות יחידה (10 נק').

עבור כל פעולה בתפריט (1 – 5) יש להציג שתי דוגמאות הרצה מוצלחות ושתי דוגמאות הרצה בהן הפעולה נכשלת. עבור כל פעולה, יש להציג את מצב החשבון או החשבונות הרלוונטיים לפני ואחרי הפעולה, באמצעות פעולות 2 או 6.

עבור פעולה 7 מספיק 2 דוגמאות הרצה במצבים שונים.

קבצי הגשה:

1. `BankAccount.java` – הקוד של המחלקה `BankAccount`.
2. `BankBranch.java` – הקוד של המחלקה `BankBranch`.
3. `BankBranchTester.java` – הקוד של המחלקה `BankBranchTester`.

4. BankBranchTesterResults.txt – דוגמאות ההרצה של התכנית BankBranchTester.

תבנית של התכנית BankBranchTester (תפריט וקבלת קלט מהמשתמש):

```
// BankBranchTester.java
import java.util.Scanner;

public class BankBranchTester {

   public static void main(String[] args) {

      Scanner inScanner = new Scanner(System.in);

      final int CREATE_ACCOUNT               = 1;
      final int FIND_ACCOUNT                       = 2;
      final int DEPOSIT                            = 3;
      final int WITHDRAW                           = 4;
      final int TRANSFER                           = 5;
      final int PRINT_ALL_ACCOUNTS           = 6;
      final int PRINT_ACCOUNTS_IN_DEBT = 7;
      final int QUIT                                       = -1;

    BankBranch branch = new BankBranch("987", "Havazelet 10 TA");

      boolean done = false;
      while(!done)
      {

         System.out.printf("Please choose action:\n");
         System.out.printf("%3d - create account\n", CREATE_ACCOUNT);
         System.out.printf("%3d - print account details\n", FIND_ACCOUNT);
         System.out.printf("%3d - deposit\n", DEPOSIT);
         System.out.printf("%3d - withdraw\n", WITHDRAW);
         System.out.printf("%3d - transfer\n", TRANSFER);
         System.out.printf("%3d - print all accounts\n", PRINT_ALL_ACCOUNTS);
         System.out.printf("%3d - print accounts in debt\n", PRINT_ACCOUNTS_IN_DEBT);
         System.out.printf("%3d - quit\n", QUIT);

         System.out.print("Your choice : ");


         int action = inScanner.nextInt();
         if(action == CREATE_ACCOUNT)
         {
               System.out.printf("Please enter the new account details:\n");
               System.out.printf("Account number:             ");
               String accountNumber = inScanner.next();
               System.out.printf("Customer name:              ");
               String customerName = inScanner.next();
               System.out.printf("Initial balance:            ");
               double initialBalance = inScanner.nextDouble();
               System.out.printf("Minimal allowed balance:    ");
               double minBalance = inScanner.nextDouble();

               // TODO ...

         }
         else if(action == FIND_ACCOUNT)
```

```
{
        System.out.printf("Please enter the account number:");
        String accountNumber = inScanner.next();
        BankAccount account = branch.findAccount(accountNumber);
        if(account == null)
        {
                // TODO ...

        }
        else
        {
                // TODO ...
        }

}
else if(action == DEPOSIT)
{
        System.out.printf("Please enter the account number:");
        String accountNumber = inScanner.next();

        System.out.printf("Please enter amount to deposit:");
        double amount = inScanner.nextDouble();

        boolean isSuccessfull =                // TODO ...

        if(isSuccessfull == true)
        {
                // TODO ...
        }
        else
        {
                // TODO ...
        }
}
else if(action == WITHDRAW)
{
                System.out.printf("Please enter the account number:");
        String accountNumber = inScanner.next();

        System.out.printf("Please enter amount to witdraw:");
        double amount = inScanner.nextDouble();

        boolean isSuccessfull =                // TODO ...

        if(isSuccessfull == true)
        {
                // TODO ...
        }
        else
        {
                // TODO ...
        }
        }
else if(action == TRANSFER)
{
                System.out.printf("Transfer from account : ");
        String accountFrom = inScanner.next();

        System.out.printf("Transfer to account   : ");
        String accountTo = inScanner.next();

        System.out.printf("Amount to transfer    : ");
```

```
                double amount = inScanner.nextDouble();

                boolean isSuccessfull =                // TODO ...

                if(isSuccessfull == true)
                {
                        // TODO ...
                }
                else
                {
                        // TODO ...
                }
                }
        else if(action == PRINT_ALL_ACCOUNTS)
        {
                // TODO ...
        }
        else if(action == PRINT_ACCOUNTS_IN_DEBT)
        {
                // TODO ...
        }
        else if(action == QUIT)
        {
                done = true;
                System.out.printf("Goodby !\n");

        }
        else
        {
                System.out.printf("Incorrect input. Please try again\n");
        }
    }
  }

}
```

בהצלחה רבה !