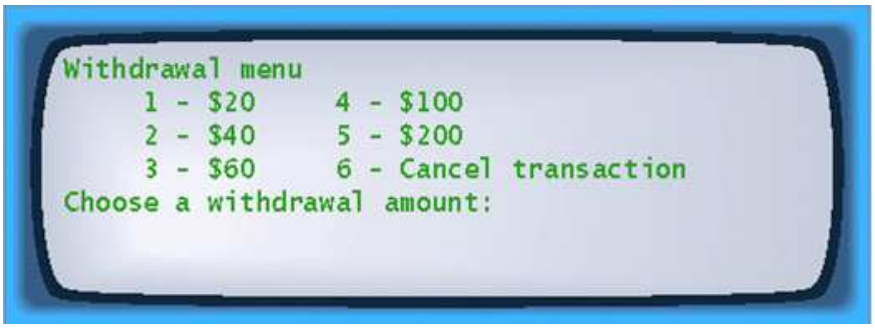
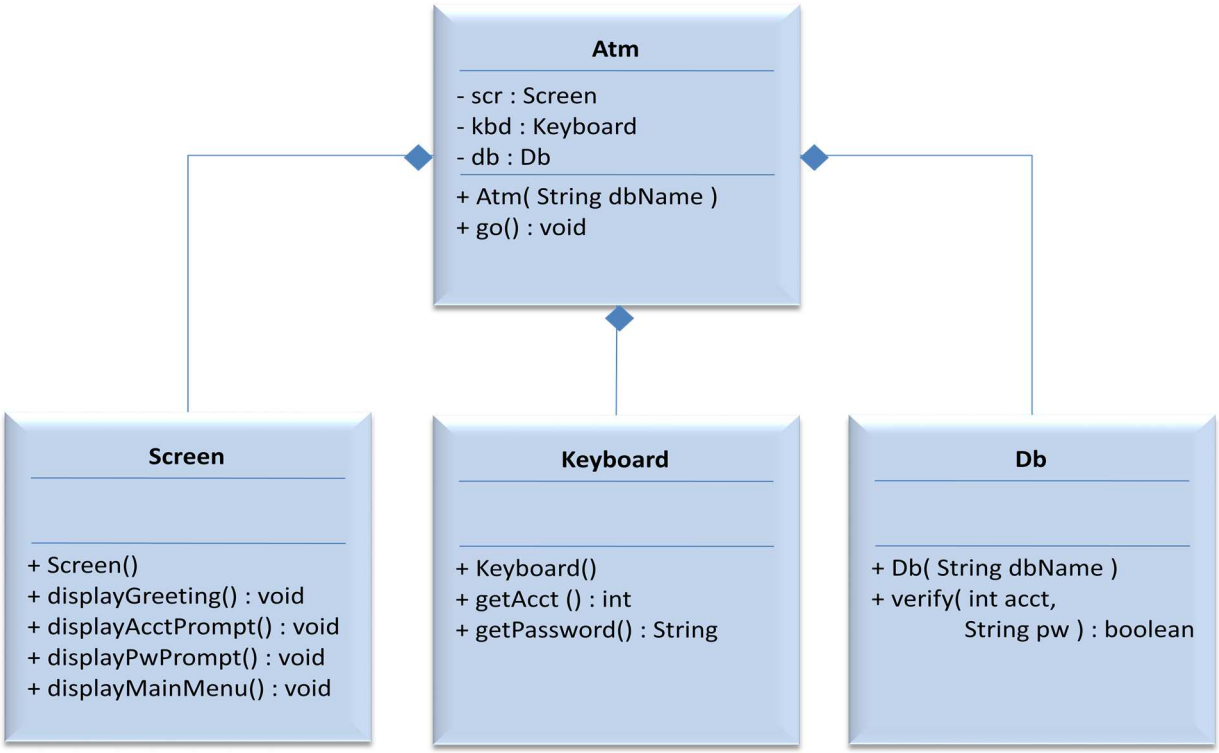
תרגול 3#– כספומט

בתרגול הזה נעסוק במימוש כספומט: נתחיל בשלד פשוט ונוסיף תכונות ויכולות ככל שהשכלתנו ב-Java תתרחב בהמשך הקורס.

הנה כמה מסכים ראשונים שמשתמש בכספומט עשוי לראות, וזאת על מנת להדגים את דרך פעולתו:



בשלב הנוכחי נממש ארבע מחלקות, המייצגות את הכספומט, את המסך והמקלדת של הכספומט ואת מסד הנתונים אתו הוא מתייעץ. להלן ה-UML של מחלקות אלה. זהו רק תיכון, ואפשר לכלול במימוש משתנים נוספים על אלה שב-UML וכן מתודות נוספות.



סימן '+' לפני שם של מאפיין של מחלקה בא לציין שהמאפיין הוא ציבורי –**public**, בשעה ש '-' מציין מאפיין פרטי, **private**.

בכתיבת הקוד יש להקפיד על הכללים והקונוונציות של כתיבת קוד Java:

1. הקוד של כל מחלקה נמצא ב-_____ נפרד.
2. שמות מחלקות מתחילים ב-_____ .
3. בהגדרת מתודה, יש לכתוב את הדברים הבאים. באיזה סדר?

a. פרמטרים בתוך _____ . גם אם אין פרמטרים למתודה, על ה-_____ להופיע בכל זאת.

b. מזהה ניראות (**public**, **private**)

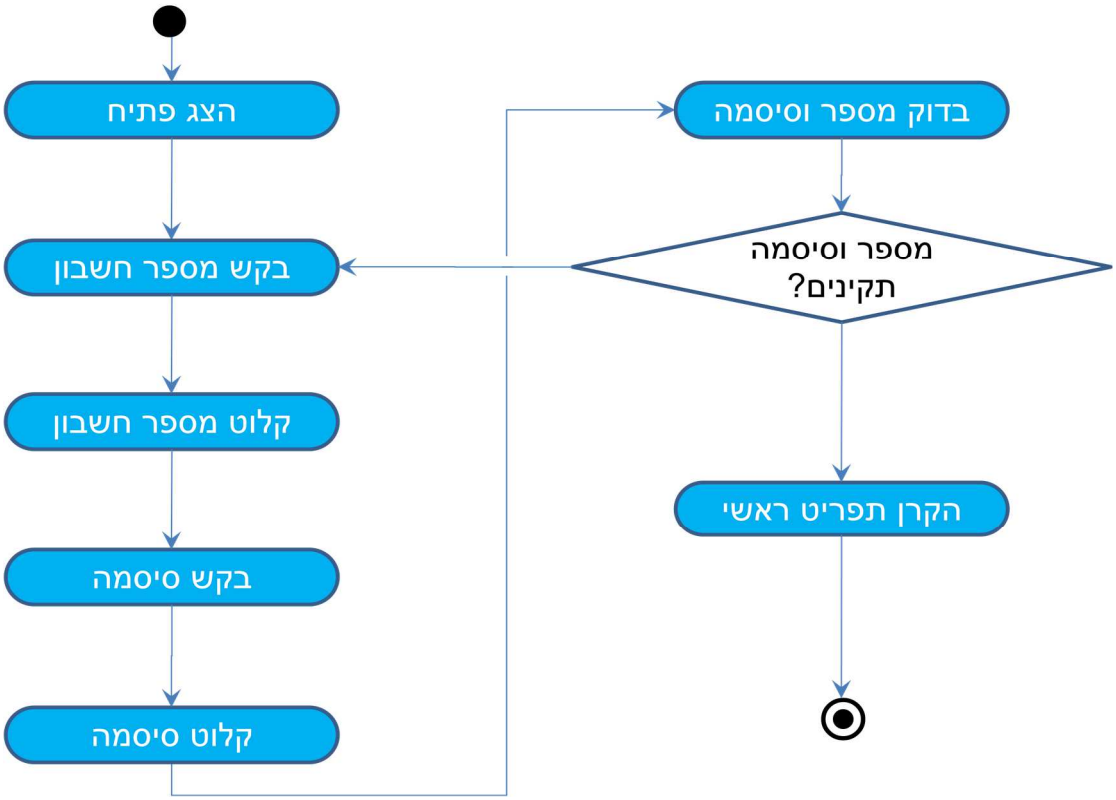
c. ערך מוחזר

d. גוף המתודה: הפקודות אותן יש לבצע, בתוך _____ . גם אם המתודה ריקה ואינה מבצעת דבר, על ה-_____ להופיע בכל זאת.

e. שם המתודה, שמתחיל ב-_____ .

איזה מהפריטים לעיל אסור שיופיע בהגדרת בנאי?

להלן דיאגרמת מצבים :

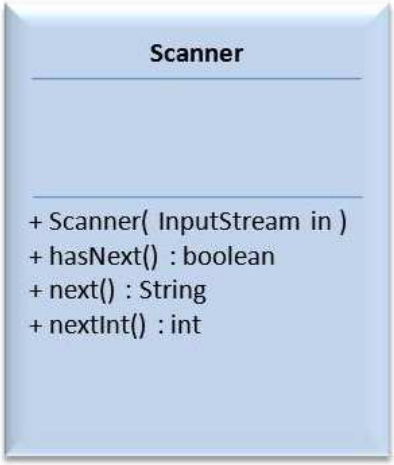


הדיאגרמה מתארת את מהלך העניינים הקשור רק במסך הראשון מבין השלושה שהוצגו לעיל.
אתם מוזמנים לייצר דיאגרמת מצבים עבור המסכים האחרים. מי שרוצה להרחיב את התכנית לכלול תפריטים נוספים בהחלט מוזמן לנסות זאת ולהגיש לי בנפרד.

את המתודה main אפשר לממש בכל אחת מהמחלקות, אך המקום הטבעי עבורה הוא Atm.
(אפשר גם להגדיר מחלקה מיוחדת שתכיל את main בלבד, אך זה מיותר במקרה זה.) כדי להתניע את העניינים, הנה הקוד של המתודה main - אשמח אם תתאימו את הקוד שלכם כך שהמתודה הנתונה בזה תפעל עמו בהרמוניה ☺

```
public static void main( String[] args )
{
	Atm atm = new Atm( "myDatabase" );
	atm.go();
}
```

לצורך קריאת קלט מהמקלדת, נשתמש במחלקה שעוד לא פגשנו, המוגדרת ב-JDK, ששמה Scanner. כמו System.out.println() שמשמש להדפסת תוצאות למסך, Scanner משמש לקריאת נתונים מהמקלדת, בדומה ל-scanf של C. הנה תיאור המחלקה ב-UML:



לפני שאפשר להשתמש במחלקה, יש להודיע על קיומה לקומפיילר (Scanner אינה כלולה באותה חבילת מחדל בה String ו-System נמצאות, ועל כן יש להכריז עליה במפורש):

```
import java.util.Scanner;          // הכרז על השימוש במחלקה Scanner
```

על פקודת ה-import להופיע בראש הקובץ, לפני הגדרת המחלקה.

על מנת לקרוא ממקור יחיד כלשהוא, יש ליצור מופע של המחלקה Scanner (באמצעות **new**), ולספק את שם המקור לבנאי. במקרה שלנו, המקור הוא המקלדת, כך שעלינו להשתמש בקלט הסטנדרטי, המקביל ל-stdin ב-C: בג'אווה הוא נקרא System.in:

```
Scanner s = new Scanner( System.in );    // צור מופע שקורא מהמקלדת
```

כעת המופע s יכול לשמש לקריאת נתונים מהמקלדת. ניתן לקרוא מספר רב של פעמים באמצעות אותו המופע s ע"י השימוש חוזר במתודה next(): בכל פעם שהיא נקראת, המתודה מחזירה את הרצף הבא של סימנים ברי הדפסה כפי שהוא ללא תרגום, בצורת String. הרצף מסתיים עם הסימן הראשון בקלט שאינו משאיר רישום על נייר (רווח, סימן ה-tab, סוף שורה וכדומה).

אנו רגילים משפת C לקרוא עד שהקובץ נגמר, תופעה המסומנת ע"י החזרת הערך **null**. זהו מנגנון לא מקובל ב-Java: כאן רגילים לדעת מראש על הסוף המגיע. לשם כך קיימת המתודה hasNext() המחזירה ערך בוליאני: **true** אם יש עוד מה לקרוא בקלט, ו-**false** אם הגענו אל סופו. יש לקרוא למתודה זו **תחילה**, ורק אם היא מחזירה **true** מותר לקרוא ל-next().

יש ל-Scanner מתודות נוספות היכולות לשמש במקרים אחרים, ונמנה אותן בהזדמנות שאנו משוחחים על אודות מחלקה שימושית זו:

- nextInt() – קורא מהקלט את רצף הסמנים הארוך ביותר שניתן להבנה כמספר שלם (זה יכול להיות סימן + או – ואחריו רצף של ספרות). הקריאה נעצרת בסימן הראשון שאינו יכול להחשב חלק ממספר שלם – רווחים (כמו במקרה של next()) או סימן דפיס שאינו ספרה. המתודה מחזירה ערך מטיפוס int.

- hasNextInt() – מחזירה **true** אם ורק אם יש אפשרות לקרוא **int** מהקלט. יש לקרוא לה לפני שקוראים ל-nextInt(). אם היא מחזירה' **false** אין זה אומר שאין יותר קלט...
- nextLine() – קורא מהקלט את כל הסימנים שבשורה אחת ומחזיר את השורה בשלמותה כ-String.
- hasNextLine() – מחזירה **true** אם ורק אם יש שורה נוספת בקלט שאפשר לקרוא.

לדוגמא:

```
import java.util.Scanner;

class TryScanner {
    public static void main( String[] args )
    {
        Scanner s = new Scanner( System.in );
        String a;
        if( s.hasNext() )  {
            a = s.next();
            System.out.println( a );
        }
        if( s.hasNext() )  {
            a = s.next();
            System.out.println( a );
        }
        if( s.hasNext() )  {
            a = s.next();
            System.out.println( a );
        }
        int i;
        if( s.hasNextInt() )  {
            i = s.nextInt();
            System.out.println( i );
        }
    }
}
```

התכנית לעיל, אם תפעל על הקלט הבא:

```
abcde   12345
!@#$%999abc
```

תדפיס את הפלט הבא:

```
abcde
12345
!@#$%
999
```

שימו לב ש מופע יחיד של Scanner משמש למספר רב של קריאות מהקלט.

התכנית הבאה מעתיקה את הקלט אל הפלט:

```
import java.util.Scanner;

class TryScanner2
{
    public static void main( String[] args )
    {
        Scanner s = new Scanner( System.in );
        while( s.hasNext() )
        {
                System.out.print( s.next() );

        }
    }
}
```

בהצלחה!