2

מבנה התכנית

התכנית הראשונה שלי

מחלקה מגדירה עצם. עצמים מתקשרים ביניהם באמצעות תכנית מַנְחָה. בפרקים הבאים נכיר תחילה את מבנה התכנית המנחה ואת הוראות השפה, ובהמשך נלמד לשלב גם עצמים.

לפניך תכנית בשפת Java. התכנית קולטת את גילו של אדם ומדפיסה את גילו בשנה שאחרי. לבסוף מדפיסה ברכה להמשך יום נעים:

```
 /*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~*
  *            תכנית דוגמא             *
  *~~~~~~~~~~~~~~~~~~~~~~~~~~~~~*/
import java.util.Scanner;
public class Introduction
{
        public static void main(String [] args)
        {
                double age;
                String name;
                Scanner input = new Scanner(System.in);
                System.out.println("What is your name? ");
                name  = input.next();
                System.out.println("How old are you? ");
                age =input.nextDouble();
                age = age + 1;
                System.out.print ("Hello " + name+"\n");
                System.out.println ("Next year you will be "+ age);
                System.out.println ("Have a nice day   :o)");
        }
}
```

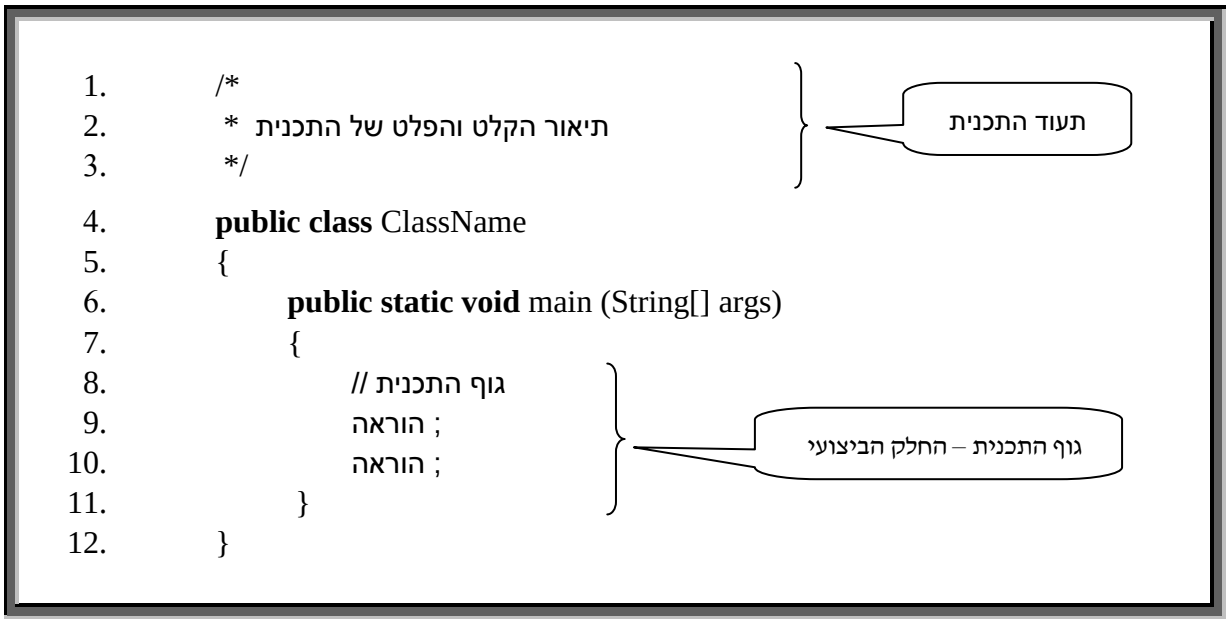
בתכנית מופיעים מאפיינים רבים של שפת Java, כגון שימוש במשתנים, הוראות **קלט**, **עיבוד** הנתונים שנקלטו, הוראות **פלט** והוראת השמה.

יתכן וקוד התכנית נראה לא ברור בשלב זה, אך בהמשך נכיר את כל המאפיינים שהוזכרו ונלמד לכתוב תכניות משלנו.

בסיומו של פרק זה, נהיה מסוגלים לכתוב תכניות כדוגמת תכנית זו.

המבנה הכללי של תכנית בשפת Java

במהלך לימוד פרקי הספר השונים, יובהרו כל מרכיבי התכנית.



הגדרות התחלת התוכנית

שורות 1 – 3: תעוד התכנית.

שורות אלו הן **הערות**. ההערות אינן הוראות לביצוע. הן משמשות להוספת מידע, הסבר והנחייה לתכנית, והמהדר מתעלם מהן.

שורה 4: הכרזה על מחלקה.

שפת Java היא שפה מכוונת עצמים. שפות מכוונות עצמים משתמשות במחלקות כדי להכריז על **עצמים (אובייקטים)**. התכנית שלנו הינה אובייקט. כדי ליצור אובייקט יש ליצור מחלקה שתגדיר אותו.

השם ClassName הוא השם שבחר כותב התכנית לקרוא למחלקה בעת יצירת התכנית החדשה. בשם זה יישמר קובץ התכנית בדיסק, ולכן רצוי מאוד לתת לתכנית שם משמעותי שיקל על זיהוייה ברשימת הקבצים.

שם מחלקה יתחיל תמיד באות גדולה, ומכאן שגם שם התכנית יתחיל באות גדולה.

הסוגריים המסולסלים בשורות 5 ו- 12 קובעים את גבולות בלוק המחלקה.

שורה 6: כותרת הפעולה הראשית ()main

המחשב צריך לדעת מהיכן להתחיל את ביצוע התכנית. התכנית ב- Java מתחילה תמיד מהפעולה הראשית main. פעולה הינה קטע קוד שנועד לבצע משימה מוגדרת.
כותרת הפעולה הראשית:

public static void main (String [] args)

הסוגריים המסולסלים בשורות 7 ו- 11 קובעים את גבולות הפעולה main.

שורות 8 - 10: החלק הביצועי - התכנית.

החלק הביצועי תחום בתוך סוגריים מסולסלים הפותחים את התכנית בשורה 7 ומסיימים אותה בשורה 11.

הערות ותיעוד מתכנית

על מנת שתכנית תהיה ברורה לקורא התכנית, היא חייבת להיות מוסברת (מתועדת) היטב, מעוצבת וכתובה לפי מוסכמות.

תיעוד (documentation)

נושא התיעוד הפך לנושא מרכזי וחשוב בפיתוח תוכנות מחשב. בתהליך פתוח התכנית משתתפים מתכנתים רבים. תיעוד התכנית מסייע למתכנת להבין את הרעיון שעומד מאחורי הקוד שכתב מתכנת אחר, או במקרה של מפתח יחיד, התיעוד מסייע להיזכר בקו המחשבה שליווה אותו במהלך פיתוח התכנית כשירצה לתחזק אותה או לשדרגה בשלב מאוחר יותר.

במשך הזמן התפתחו **מוסכמות** (conventions) של כללים בכתיבת תוכנה. אמנם המחשב אינו זקוק לכללי כתיבה - התוכנית תרוץ גם אם לא נעבוד לפיהם, אך הם בהחלט מחייבים בקהילת המתכנתים ובתי התוכנה. כללים אלו תורמים **לקריאות** (readability) התוכנית ומסייעים בתחזוקה השוטפת שלה.

סביבות העבודה החדשות, הנהוגות כיום, כוללות בתוכן הכנה לתעוד בצורות שונות.

הערות

ההערות הינן כלי תחבירי של השפה באמצעותו ירשם התיעוד בתכנית.

הערות הינן קטע בתכנית שאינו חלק מהוראות השפה ונועד להבהיר למתכנתים אחרים את התכנית. להערות סימון מיוחד שנועד להבהיר למהדר (קומפיילר) שלא מדובר בהוראות השפה ויש להתעלם מקטע זה. שים לב שבסביבת העבודה, מסומנות ההערות בצבע ירוק.

נבחין בין שני סוגים של הערות:

- **הערת קטע**: /* הערת קטע */ הערה היכולה להשתרע על פני מספר שורות.
הערת קטע נפתחת בסימן /* ומסתיימת בסימן */ (ההערה נתחמת בתוך סימני הלוכסן וכוכבית צמודים ללא תו רווח ביניהם). גוף ההערה יכול להיכתב בשפה חופשית, כולל בשפה העברית. שים לב שהעורך מוסיף כוכבית בתחילת כל שורה. כוכבית זו הינה רשות ולא חובה.
- **הערת שורה**: הערה מסוג זה מתחילה בסימן // ומסתיימת בסוף השורה.

תיעוד התוכנית

השורות הראשונות בתוכנית הן תיעוד התוכנית. שורות אלו כוללות מידע כללי על התוכנית, הרשום בצורה של הערות. לפי הדרישות תכנית הלימודים שלנו, חובה לרשום תיעוד בראש כל תכנית. התיעוד יכול להירשם בשפה חופשית המתארת את קלט והתכנית ומטרתה או בצורה פורמאלית כמתואר בדוגמא שלהלן. נהוג לרשום את המידע הבא:

- קלט - מאפייני הקלט של התוכנית.
- פלט - פלט של התוכנית.

תיעוד התכנית יסומן כהערת קטע: דוגמא לתעוד:

```
/*
*                    קלט: a צלע הריבוע
*                    פלט: s שטח הריבוע
*/
```

מילים שמורות

בכל שפת תכנות קיימות **מילים שמורות**. מילה שמורה היא מילה שהשימוש בה והמובן שלה שמורים לשימוש בלעדי ע"י השפה ולא ניתן לשנות את משמעותה. מילים שמורות מסומנות בצבע מיוחד בתכנית. בספר זה המילים השמורות מופיעות בקוד התכנית בגופן מודגש.

עיצוב כללי

- המספור שמשמאל לשורות אינו שייך לתוכנית, אך מסייע להתמצאות בתכנית. מרבית סביבות העבודה מאפשרות להציג מספור זה. הוספת שורות בין שורות התכנית תגרום לעדכון אוטומטי של המספור.
- התוכנית לביצוע תהיה מורכבת מהוראות השפה ותיכתב בתוך הסוגריים המסולסלים תוך שמירה על כללי כתיבה מקובלים כמו **הזחות** (indent) - הקובעות את מיקום תחילת השורה ביחס לשורה שלפניה. חשוב להקפיד על הזחות ברורות.
  בדרך כלל קובע העורך (editor) של סביבת העבודה את ההזחות בצורה אוטומטית. ההזחות תורמות לקריאות התוכנית.
- יש עורכים הקובעים אוטומטית את מיקומם של הסוגרים המסולסלים של תחילת הבלוק:

```
public class ClassName {
        ...
}
```

```
public class ClassName
{
        ...
}
```

שתי צורות הכתיבה נכונות, ושתיהן תופענה בספר.

- כל הוראה בשפת Java חייבת להסתיים בסימן נקודה-פסיק (**;**).
- **תווי רווח** (whitespace): תווי רווח הינם רווחים בין המילים, טאבים (tab) וירידת שורה (מקש Enter). המהדר מתעלם מתווי רווח. המשמעות - בכל מקום בו מותר תו רווח אחד, מותר לשים הרבה תווי רווח ואף לעבור לשורה חדשה. ריווח התכנית תורם לנוחות הקריאה.
  שים ♥: למרות האמור לעיל, המהדר אינו מתעלם מתווי רווח בתוך מחרוזת טקסט המופיעה בין גרשיים. הקשה על מקש Enter בתוך מחרוזת טקסט יגרור הודעת שגיאה.
- שפת Java הינה שפה **תלוית רישיות - case sensitive**, כלומר קיימת הבחנה בין אותיות גדולות (upper case) ואותיות קטנות (lower case). המילים Main ו- main נחשבות למילים שונות בשפה. חובה להקפיד על גודל אות מתאים. בכותרת התכנית חובה לרשום main באות קטנה.

משתנים

תכניות פועלות על מידע. המידע עליו פועלת התכנית נמצא בזיכרון המחשב בזמן הפעולה.

משתנה (variable) הוא תא זיכרון אשר ניתן לשמור בו ערך ולקרוא (לאַחזֵר - Retrieve) את הערך השמור בו במהלך ביצוע התכנית. ערך זה נקרא **ערך המשתנה**. הכינוי "משתנה" נובע מכך שבמהלך ביצוע התכנית ניתן לשנות את ערכו שוב ושוב.

בזמן פיתוח אלגוריתם/תכנית בוחר המתכנת במשתנים אשר ישמשו לשמירת נתוני קלט ולשמירת תוצאות חישוב שיתקבלו במהלך ביצוע התכנית. כדי להשתמש במשתנים אלו בתוכנית, יש **להכריז** (declare) על המשתנים איתם נעבוד. פניה למשתנה נעשית באמצעות שם הניתן לו על ידי המתכנת. שם זה הוא **שם המשתנה**.

משתנה יכול **לקבל ערך** באחת משתי הצורות הבאות:

- משפט השמה. המתכנת קובע את ערך המשתנה בזמן כתיבת התכנית.
- משפט קלט. התכנית קולטת ערך מהמקלדת במהלך הביצוע. הקלט מוזן על ידי המשתמש בתכנית.

שימוש במשתנה (בערך המשתנה):

- ניתן לבקש להדפיס את תוכן המשתנה בהוראת פלט.
- ניתן להתייחס לתוכן המשתנה לביצוע פעולות שונות.
- ניתן לשמור במשתנים את התוצאות של חישובים שונים שבוצעו במהלך התכנית.

תכונות כלליות של משתנים

- ניתן להכריז על משתנים בכל מקום בתכנית, ובלבד שנכריז עליהם לפני השימוש הראשון בהם. אף על פי כן, ולמען הסדר הטוב, נשתדל לרכז את כל שורות ההכרזה בתחילת התכנית.
- במשתנה נשמר הערך האחרון שהושם בתוכו. המשתנה אינו "זוכר" מה היה בו לפני כן.
- מתן ערך התחלתי למשתנה נקרא **אִתְחוּל**.
- לכל משתנה יש ערך. הערך של משתנה שעדיין לא ביצענו לו השמה אינו ידוע. מנקודת המבט של המתכנת ערך המשתנה אינו מוגדר למרות שתמיד יש בו ערך כלשהו. במהדרים מסוימים, אי אתחול משתנה יגרור הודעת שגיאה.
- יש מהדרים שמאתחלים אוטומטית חלק מהמשתנים. תכנות נכון אינו תלוי במהדר. **יש לוודא שכל משתנה מקבל ערך התחלתי לפני שמשתמשים בו**.

הכרזה על משתנים בתכנית

ככלל, משתנה חייב להיות מוגדר לפני השימוש הראשוני בו. נהוג להגדיר את משתני הפעולה בתחילתה, לפני ההוראות לביצוע, באופן הבא:

```
/*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~*
 *              (תעוד התכנית)              *
 *~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~*/

 public class ClassName {
        public static void main (String[] args){

                //---  הצהרה על משתני התכנית  ---
                טיפוס-משתנה  משתנה1 ;
                טיפוס-משתנה  משתנה2, משתנה3, משתנה4 ;

                // גוף התכנית
                הוראה ;
                הוראה ;
        }
}
```

- חובה להצהיר על כל משתני התכנית.
- מספר שורות ההגדרה אינו מוגבל.
- אין לכלול משתנים מטיפוסים שונים באותה שורת הגדרה.
- נהוג להכריז על משתנים בעלי תפקידים דומים באותה שורה, ולספק תיאור תפקיד המשתנה כהערה בקצה השורה.
- לכל משתנה יש **טווח הכרה** (scope). משתנה מוכר וקיים בתוך הבלוק שבו הוא נוצר. למשל: משתנה שהוגדר בפעולה אחת, אינו מוכר בפעולה אחרת.

שם משתנה

שם חוקי יכול להיות מורכב מסדרה של אותיות אנגליות, ספרות וקו תחתי _ , וחייב להתחיל באות. אסור לכלול תווי רווח וסימנים אחרים בשם המשתנה. שם משתנה אינו יכול להכיל אותיות עבריות. אין מגבלה מעשית לאורך השם.

שפת Java היא שפה תלוית רישיות (case sensitive), כלומר יש חשיבות לגודל אות. מקובל שאם שם המשתנה מורכב ממספר מילים, תתחיל כל מילה, החל מהמילה השניה, באות גדולה. לדוגמא: newNumber , myName , numberOfStudents . בגלל הרגישות לגודל אות, הרי שהשמות: newNum , NewNum , newnum מתייחסים לשלושה משתנים שונים.

שמות חוקיים: start, old, firstName, number, size37 .

שמות לא חוקיים: family name , name.2 , first-second , -number , a#5 , why? (מדוע?)

שים ♥ : **מילים שמורות אינן יכולות לשמש כשמות של משתנים**.

מוסכם ששמות משתנים יהיו תמיד שמות משמעותיים – המתארים את השימוש במשתנה.

דוגמאות: משתנה האמור להכיל שם של תלמיד ייקרא – **name** או **studentName** .
משתנה האמור להכיל גיל ייקרא – **gil** או **age**.
משתנה האמור להכיל את גודל החדר ייקרא - **size** או **roomSize** .

טיפוסי משתנים

לכל משתנה יש להגדיר את **סוג הנתונים** שלו (data type). סוג הנתונים של המשתנה קובע איזה ערכים יוכל המשתנה להכיל ואלו פעולות ניתן לבצע על המשתנה.

שפת Java מספקת מספר סוגים של **משתנים בסיסיים** (משתנים פרימיטיבים), שבהם ניתן לאחסן מידע מספרי, מידע טקסט (מחרוזת טקסט) או מידע לוגי ("אמת" או "שקר"). כל סוג משתנה הוא בעל דרישות אחסון שונות בזיכרון המחשב ונבדל בסוג המידע שניתן לאחסן בתוכו ובסוג הפעולות אותו ניתן לבצע על מידע זה. לכל משתנה בשפה יש **תכונות** (תחום הערכים שהוא יכול לקבל) ו**פעולות** אותן ניתן לבצע במשתנים אלו.

בתכנית הלימודים **יסודות** נשתמש רק בחלק מטיפוסי הנתונים הקיימים בשפה.

int – משתנה מטיפוס מספר שלם (integer)

int הוא משתנה מטיפוס מספרי שלם, התופס 4 בתים (32 סיביות) בזיכרון.
תחום הערכים שהוא יכול לקבל הוא מ- -2^{31} ועד $2^{31}-1$, כלומר מ- -2,147,483,648 ועד +2,147,483,647. ניתן להוסיף סימני +/- לנתון המספרי.

דוגמאות למספרים שלמים: 2- , 0 , 1996 , 641- , 17+ , 9

double – משתנה מטיפוס מספר ממשי (floating point)

לטיפול ואחסון מספרים ממשיים (מספרים בעלי נקודה עשרונית), קיימים שני סוגי משתנים עיקריים: float ו- double.

ברירת המחדל של Java הוא double המיוצג על ידי 8 בתים (64 סיביות). מידת הדיוק של משתנה מטיפוס זה הוא 15 עד 16 ספרות אחרי הנקודה העשרונית. **float** הוא משתנה המכיל מספר ממשי המיוצג ב- 4 בתים (32 סיביות). אם נרצה להשתמש בקבועים מספריים מטיפוס float (בעל 7 ספרות דיוק), נוסיף את האות f לקבוע המספרי. לדוגמא: 3.14f ישמר במשתנה מסוג float ולא double לפי ברירת המחדל.

דוגמאות למספרים ממשיים: 3.2456 , 583.12 - , 4.0 , 0.0091

char - משתנה מטיפוס תווי (character)

תווים הם אותיות, ספרות (להבדיל ממספרים) וסימני טקסט שונים. תו מיוצג בשפה על ידי תחימתו בין גרש שמאלי לגרש ימני: '3' , '@' , 'a' .

המחשב אינו מזהה תווים. הוא מסוגל לזהות רק מספרים, לכן כל התווים מאוחסנים כערכים מספריים. כדי להבטיח שכל יצרני המחשבים בעולם ישתמשו באותם ערכים, הוגדר תקן שנקרא בשם Unicode שבו כל תו וסימן מיוצגים על ידי מספר שלם המתאים רק לו.

משתנה מסוג char יכול להכיל ערך מספרי המתאים לתו אחד ויחיד. גודל הזיכרון שתופס משתנה זה הוא 2 בתים (16 סיביות) ומכאן שטבלת Unicode מכילה ייצוג של 65,536 (= 2^{16}) תווים שונים.

String – מחרוזת

מחרוזת בשפת Java הינה אובייקט מיוחד שנועד לעבודה עם טקסטים. מחרוזת מיוצגת בשפה על ידי תחימתה במירכאות כפולות (גרשיים): "Hello, have a nice day ☺" .

בשלב זה נעשה הכרות ראשונית עם המחרוזת. כדי שנוכל לכתוב תוכניות ברורות יותר. נושא המחרוזות נידון בהרחבה בפרק 12.

שים ♥: המילה String נכתבת באות גדולה.

boolean – משתנה מטיפוס בוליאני

טיפוס הנתונים הבוליאני משמש לבדיקת נכונותם של פסוקים. בעזרתו נוכל לדעת האם משהו מתקיים או לא מתקיים, אמת או שקר, נכון או לא נכון וכד'. משתנה מטיפוס בוליאני תופס בית אחד (8 סיביות) בזיכרון, ויכול לקבל ערך **אמת - true** או **שקר - false** בלבד.

דוגמא להצהרה על משתנים:

```
1.    /*              (תעוד התכנית)                */
2.    public class ClassName {
3.    public static void main (String[] args){
4.                  //--- הצהרה על משתני התכנית ---
5.                  int dd, mm, yy;     // משתני התאריך: יום, חודש, שנה
6.                  double age;         // גיל
7.                  String name;        // שם
8.                  // גוף התכנית
9.                  הוראה ;
10.                 הוראה ;
11.           }
12.   }
```

בדוגמא שלעיל שורה 5 מכריזה שבתוכנית נשתמש במשתנים ששמותיהם dd , mm ו- yy. הקידומת int פרושה שהמשתנים הם מ**טיפוס** (סוג) של **מספר שלם** (integer).

בשורה 6 הוצהר על משתנה בשם age מטיפוס double (טיפוס משתנה המכיל **מספר ממשי** - מספר הכולל חלק עשרוני) ובשורה 7 הוצהר על משתנה בשם name מטיפוס **מחרוזת** (טיפוס המכיל אותיות ומילים).

קבועים

לעיתים נוצר צורך להשתמש בנתון מסויים שיישאר קבוע בכל התכנית, ולא ניתן יהיה לשנותו. הנתון יכול להיות מחרוזת, מספר שלם או מספר ממשי. הכרזה על קבוע נעשית באמצעות המילה השמורה **final**, בחלק שבו מצהירים על המשתנים. ההכרזה קובעת ערך סופי לקבוע ואין אפשרות לשנותו בהמשך התכנית.

final <טיפוס-נתונים> <שם-הקבוע> = <ערך קבוע> ;

כדי להקל על זיהוי קבועים, נהוג לרשום את שמותיהם באותיות רישיות (upper case). אם שם הקבוע מכיל יותר ממילה אחת, נשתמש בקו תחתי כמפריד בין המילים. לדוגמא: MAX_VALUE.

הרחבה על קבועים ראה בפרק 7 - לולאות.

טיפול במשתנים – הוראות פלט השמה וקלט

כדי ליצור אינטראקציה בין המשתמש בתכנית והתכנית עצמה משתמשים בהוראות קלט ופלט.
בעוד שהוראות הקלט משמשות לקבלה והכנסה של נתונים מן המשתמש לתוך המשתנים, הרי שהוראות הפלט נועדו להציג את תוכן תאי הנתונים.

הוראות הקלט והפלט בשפת Java נעשות דרך אובייקט הפלט out שבמחלקה **System.**

הפעלת פעולה של **מחלקה** נעשית בדרך הבאה: **( ) שם-הפעולה.שם-המחלקה**
ClassName.MethodName ( )

הפעלת פעולה של **עצם / אובייקט**: **( ) שם-הפעולה.משתנה-העצם.שם-המחלקה**
ClassName.objectName.MethodName ( )

הוראות פלט

שליחת מידע לאמצעי הפלט הסטנדרטי (המסך):

System.out.print (המידע שיוצג על המסך) ;
System.out.println (המידע שיוצג על המסך) ;

המידע המועבר לפעולה יכול להיות:

- קבוע מספרי: **System**.out.println (5);
- קבוע מחרוזתי: **System**.out.println ("Have a nice day");
- תוכן של משתנה: **System**.out.println (num);
- צרוף של מחרוזות ומשתנים: **System**.out.println ("There were " + count + " items");
  צרוף של פרטי מידע נקרא **שרשור**.

הוראת **System**.out.print מציגה את המידע ומשאירה את הסמן במקום בו הסתיימה הכתיבה.
ההוראה **System**.out.println מציגה את המידע ומעבירה את הסמן לתחילת השורה הבאה.

מכאן שקטע ההוראות הבא:

```
System.out.print ("My lucky number ");
System.out.print ("is : ");
System.out.print (num);
System.out.println();
```

שקול להוראה: **System**.out.println ("My lucky number is : " + num);

שורות ריקות בפלט

לעיתים נרצה להציג שורות ריקות במסך הפלט. לשם כך נשתמש בפעולה **System**.out.println ();

אם נרצה להשאיר שתי שורות ריקות, נוכל לרשום: **System**.out.println ("\n");

המחרוזת "\n" משמעותה "רד לשורה הבאה" כלומר – השארת שורה ריקה. ההוראה **System**.out.println() תציג על המסך את תוכן המחרוזת \n (שמשמעותה "רד לשורה הבאה") ואחר כך תרד שורה (כי כך נוהגת println() להבדיל מ- print()).

אם נרצה שלוש שורות ריקות, נרשום: **System**.out.println ("\n\n");
(כל \n יגרום לירידת שורה נוספת).

כשמשתמשים בפעולה **print()**, נשאר ראש הכתיבה במקום בו הסתיימה הכתיבה למסך.
הפעלת הפעולה **println()** תגרום להעברת ראש הכתיבה לתחילתה של השורה הבאה.
ואילו ההוראה: **System.out.println ("\n");** תגרום לירידה לשורה הבאה ובנוסף השארת שורה ריקה.

שרשור (צירוף) מחרוזות

שרשור מחרוזות הוא צירוף מחרוזת אחת לסופה של האחרת. (שרשור מהמילה 'שרשרת', צירוף מחרוזות של תווים בזו אחר זו כמו חוליות בשרשרת). פעולת השרשור נעשית באמצעות האופרטור + (חיבור) הפועלת על משתני מחרוזת וקבועים מחרוזתיים.

פעולת השרשור להדפסת מחרוזות:

```
String name = "Dror";
System.out.println ("Hello, " + name);
```

אם נרצה לשרשר מחרוזת וביטוי חשבוני, נתחום את הביטוי החשבוני בסוגריים.

לדוגמא:

```
String str = "number is: ";
int  a = 5;
```

ההוראה	המחרוזת המתקבלת
System.out.println ("a is: " + a + " next " + str + a + 1);	a is: 5 next number is: 51
System.out.println ("a is: " + a + " next " + str + (a+1));	a is: 5 next number is: 6

עברית בסביבת Java

כאשר כותבים מחרוזות בשפה העברית, ומשלבים אותן במשתנים מחרוזתיים המכילים מחרוזות עברית, יש לשים לב לסדר הכתיבה למסך. לדוגמא:

תוכן המשתנה name	ההוראה	הפלט
דרור	**System**.out.println (name + "שלום");	דרור שלום
דרור	**System**.out.println ("שלום " + name);	שלום דרור

אם המחרוזת שבמשתנה name היא בשפה האנגלית, יתקבל פלט התואם את סדר הכתיבה:

תוכן המשתנה name	ההוראה	הפלט
Dror	**System**.out.println (name + "שלום");	שלום Dror
Dror	**System**.out.println ("Hello " + name);	Hello Dror

כאשר המשתנים הם מטיפוס מספרי, יתנהג הפלט כמו במקרה של מחרוזת בשפה האנגלית.

הוראת השמה

הוראת השמה משמשת את כותב התכנית להכנסת נתונים לתוך המשתנים. השמת נתון למשתנה מוחקת את הערך שהיה שמור במשתנה זה ושומרת את הערך החדש בלבד (ומכאן **משתנה** המשנה את ערכו).

א. השמת קבוע

מתכנת יכול להחליט מראש על ערך קבוע הדרוש לו בשלב מאוחר יותר בתכנית.

```
int       a, b, c ;
double  x, y, z ;
String    greeting ;

a =     5 ;
b =     -39 ;
x =     2.35 ;
greeting = "hello world !";
```

קרא:

```
<שם-משתנה> מקבל <קבוע> // <שם-תא-מקבל> = <קבוע> ;
```

השמה של קבוע למשתנה שימושית לפעולה של **אתחול בהגדרה**. ניתן לתת למשתנים ערך התחלתי בשלב ההגדרה.

דוגמא:

```
int  a = 5, b = 21 ;
double  x = 7.32 ;
String str = "abc" ;
```

שים ♥: טיפוס הערך המושם בתא חייב להיות תואם את טיפוס הנתונים של התא המקבל.

ב. השמת ערך ממשתנה אחר

ישנם מצבים בהם מעוניין כותב התכנית להעביר ערך ממשתנה אחד למשתנה אחר. למשל, כשהוא מעוניין לשמור, לשימוש בהמשך, ערך הנמצא במשתנה מסויים, כאשר למשתנה זה עומד להיכנס ערך חדש.

```
<שם-תא-מקבל> = <שם-תא-נותן> ;
```

```
int  a = 12, b = 5, c ;
double  x = 5.24, y, z ;

z = x ;     משתנה מטיפוס ממשי (double) יכול לקבל תוכן של משתנה ממשי
y = b ;     או של משתנה שלם.
c = a ;     משתנה מטיפוס שלם יכול לקבל תוכן של משתנה שלם בלבד.
```

ניסיון להשמת ערך ממשי (המכיל נקודה עשרונית), קבוע או מתוך משתנה, בתוך משתנה מטיפוס שלם הינו שגיאה. השמה הפוכה של שלם לתוך ממשי הינה אפשרית.

לדוגמא: השמת x לתוך c יגרור את הודעת השגיאה הבאה: *possible loss of precision*
שפירושה: אבדן אפשרי של מידע. הרחבה בפרק 4.

שים ♥: מועבר רק עותק מהמשתנה הנותן. ערכו של משתנה זה אינו משתנה.
אם במשתנה המקבל היה ערך כלשהו, ערך זה נמחק בעת ההצבה.

3. השמת תוצאה של חישוב

ברוב המקרים, הערך אותו רוצים להציב במשתנה הוא תוצאה של חישוב.

למשל, כאשר רוצים לחשב שטחו של מלבן, יש לכפול את אורכו ברוחבו. התכנית חייבת אם כן, לכלול פקודה שתורה למחשב להציב במשתנה המיועד לשטח המלבן את מכפלת אורך המלבן ברוחבו.

הערכים הנכללים בביטוי הם קבועים-מספרים או ערכים השמורים במשתנים מספריים. פעולת החישוב היא אחת מפעולות החשבון כפל, חילוק, חיבור ו/או חיסור.

מבנה השורה במקרה זה יהיה כמו במקרים הקודמים, אלא שמימין לסימן ההשמה יופיע ביטוי חשבוני.

```
<שם-משתנה-מקבל> = <ביטוי> ;
```

דוגמאות:

```
c = a + 5 ;
y = x – 2*a ;
z = (a – y) / (a * b) ;
```

הביטוי הוא כל החלק שנמצא מימין לסימן ההשמה =.
הביטוי יכול לכלול: קבועים מספריים, משתנים, חישובים מתמטיים ואף ערך מוחזר מפונקציות מתמטיות (כפי שנלמד בפרק 5).

הפעולות החשבוניות:

חיבור	+
חיסור	–
כפל	*
חילוק	/

שים ♥:

- סדר הקדימויות של החישובים המתמטיים הוא סדר הקדימויות המתמטי:
  כפל וחילוק קודמים לחיבור וחיסור.
  סוגריים משנים את סדר החישוב.
- המשתנה לא יכיל את הביטוי החישובי אלא את תוצאת הפעולה, כלומר – ערך מספרי.
- לאחר פעולת ההשמה, המחשב לא "זוכר" את הערך שהיה במשתנה קודם לכן.
- תוצאה של פעולה חישובית שבה מעורב ערך ממשי (קבוע או משתנה) תהיה תמיד ממשית. יש להקפיד שהמשתנה המקבל את התוצאה יהיה ממשי. ראה הרחבה בפרק 4.

דוגמא:

התכנית שלהלן ממירה סכום כסף הנתון בדולרים לערכם בש"ח בהתאם לשער המרה.
לצורך העניין, נבצע המרה של 100$ לשקלים, לפי שער המרה של: 1$ = 4.5 ₪.

טבלת משתנים:

שם המשתנה	טיפוס המשתנה	תפקיד המשתנה
dollar	int	הסכום בדולרים
yatzig	double	השער היציג
shekel	double	הסכום בשקלים

חשוב: מדוע יש לקבוע את משתנה השקלים כמשתנה מטיפוס ממשי? (מה יקרה אם נקבע אותו להיות משתנה מטיפוס שלם?).

האלגוריתם:

(1) dollar ← 100
(2) yatzig ← 4.5
(3) shekel ← dollar * yatzig
(4) פלט: הערך בשקלים הוא: shekel

התכנית:

```
/*~~~~~~~~~~~~~~~~~~~~~~~~~ *
 *      המרה מדולרים לשקלים      *
 *~~~~~~~~~~~~~~~~~~~~~~~~~ */

public class Money
{
        public static void main(String[] args)
        {
                int dollar;
                double yatzig, shekel;

                dollar = 100;
                yatzig = 4.5;
                shekel = dollar * yatzig;

                System.out.println ("הערך בשקלים הוא: " + shekel);
        }
}
```

החיסרון של התכנית הוא שתמיד יומר סכום של 100$ לפי שער יציג שנקבע ביום מסויים. כדי להמיר סכומים שונים בשער יציג עדכני, נזדקק לפעולת קלט, שתקלוט מן המשתמש את הסכום שברצונו להמיר ואת שער הדולר העדכני.

הוראת קלט

אף שהצגת הפלט בשפת Java נעשית בצורה קלה ונוחה, באמצעות ההוראה **System**.out.println(), אין בשפה דרך המאפשרת קבלת קלט מן המשתמש בצורה פשוטה. על מנת לבצע קליטת נתונים מן המשתמש, נשתמש במחלקה מיוחדת Scanner שקיימת החל מ- java 1.5 (Java 5).

בשלב הראשון נשתמש בהוראות הדרושות לקלט מבלי להבינן. המשמעות תתברר בהמשך הלימוד כשנעסוק בעצמים ומחלקות בהרחבה.

המחלקה אינה חלק משפת java אלא מודול חיצוני שיש לייבא לתכנית.
את הייבוא מבצעים באמצעות הוראת import שנרשמת כשורה ראשונה בתכנית, ואחריה את שם המחלקה ומיקומה:

```
import java.util.Scanner ;
```

(המחלקה Scanner נמצאת במחשב בתת-תיקייה java\util\.. של סביבת העבודה של java
C:\j2sdk\docs\api\java\util או C:\eclipse\docs\api\java\util).

כדי לקלוט נתונים, יש ליצור אובייקט (עצם) מטיפוס Scanner.

<u>הערה</u>: בספר זה נקרא לאובייקט הקלט בשם **input**. ניתן לבחור שם אחר למשל: kelet ,reader, in או כל שם משמעותי אחר המזכיר קלט.

ההוראה נרשמת בגוף התכנית, לפני הוראת הקלט הראשונה.

```
Scanner input = new Scanner (System.in);
```

בעזרת הוראה זו ניתן לקלוט, בזמן ריצת התכנית, נתונים מן המשתמש ולשמור אותם במשתנים שהחלטנו עליהם מראש.

ההוראה מאפשרת לתכנית לקרוא מן המסך מידע שנכתב עליו באמצעות המקלדת.

```
משתנה = input.next***();
```

עבור כל אחד מטיפוסי הנתונים int ו- double קיימת פעולת input.next***() המחזירה ערך מהטיפוס. במקום *** יירשם טיפוס מספרי <u>כשהוא מתחיל באות גדולה</u>.

קלט למשתנה מספרי

קלט מספר שלם למשתנה int:

```
int num ;
num = input.nextInt ();
```

קלט מספר ממשי למשתנה double:

```
double x;
x = input.nextDouble ();
```

קלט מחרוזות ותווים

קלט מחרוזת של תווים למשתנה String:

```
String str1, str2;
str1 = input.next ();          // קלט מילה
str2 = input.nextLine();       // קלט משפט (כולל תווי רווח)
```

קלט תו בודד למשתנה char: למחלקה Scanner אין פעולה מיוחדת לקליטת תו, לכן נקלוט מחרוזת וניקח ממנה את התו הראשון, התו שבמקום 0 (הפעולה תובהר בפרק מחרוזות).

```
char ch ;
ch = input.next().charAt(0);
```

שים ♥:

- קלט חוקי עבור מספר שלם מורכב מספרות וסימני +/- לפני הסיפרה הראשונה.
  ניסיון להקליד תווים או נקודה עשרונית, יגרור הודעת שגיאה:
  xxx is not of type int – try again. (xxx הינו הערך שהוקלד)
- קלט חוקי עבור מספרים ממשיים מורכב מספרות, סימני +/- לפני הסיפרה הראשונה והאות E כמעריך למספר בייצוג מדעי. ניסיון להקליד קלט אחר יניב הודעת שגיאה דומה: הקלט אינו מהטיפוס הדרוש.
- מספר שלם שייקלט למשתנה מטיפוס ממשי, יתווסף לו נקודה עשרונית ו- 0 והוא יהפוך להיות מספר ממשי.

נשנה את התכנית שלנו, כך שייקלט המידע הדרוש מן המשתמש:

האלגוריתם:

(1) קלט: הסכום בדולרים ← dollar
(2) קלט: השער היציג ← yatzig
(3) shekel ← dollar * yatzig
(4) פלט: הערך בשקלים הוא: shekel

התכנית:

```
/*~~~~~~~~~~~~~~~~~~~~~~~~~ *
 *       המרה מדולרים לשקלים       *
 *~~~~~~~~~~~~~~~~~~~~~~~~~ */

import java.util.Scanner;

public class Money
{
        public static void main(String[] args)
        {
                int dollar;
                double yatzig, shekel;

                Scanner input = new Scanner(System.in);

                System.out.print ("כמה דולרים ברצונך להמיר ? → ");
                dollar = input.nextInt();

                System.out.print ("מהו השער היציג של הדולר ? → ");
                yatzig = input.nextDouble();

                shekel = dollar * yatzig;

                System.out.println ("הערך בשקלים הוא: " + shekel);
        }
}
```

תרגילים

משתנים

1. קבע אלו מבין הביטויים הבאים הוא שם תקין של משתנה:

(1) Y _____ (6) X% _____

(2) 3to2 _____ (7) main _____

(3) x_to_y _____ (8) newNum _____

(4) A4 _____ (9) Abcd _____

(5) x y _____ (10) My-Name _____

2. מדוע רצוי ששמות משתנים בשפה יהיו משמעותיים? _____

_____

3. איך ניתן לשלב הערה בתוכנית? מה חשיבות כתיבת הערות בתכנית? _____

_____

האם ניתן לכתוב הערה בכל מקום בתכנית? (גם באמצע מילה?) _____

_____

4. ברצוננו לשמור במשתנים בתכנית את המידע הבא:

א. מספר התלמידים בכיתה. _____

ב. שם בית הספר. _____

ג. השנה הנוכחית. _____

ד. מספר קטלוגי של פריט. _____

ה. מחיר הפריט. _____

ו. כמות. _____

ז. הרווח לאותו פריט. _____

ח. שם הפריט. _____

קבע עבור כל משתנה את טיפוס המשתנה וכתוב הצהרות מתאימות למשתנים אלו.

מבנה התכנית והוראות פלט

5. בכל אחת מהקביעות הבאות, הקף בעיגול אם **נכון** או **לא נכון**. אם לא נכון, תקן את המשפט כך שיהיה נכון.

נכון / לא נכון א. בכל תכנית חייבת להיות הפעולה main. _____

נכון / לא נכון ב. כדי לקבל קובץ ריצה, חובה לבצע פעולת הידור על הקובץ. _____

נכון / לא נכון ג. חובה לצרף הוראות הסבר בתחילת כל תכנית מחשב._____

נכון / לא נכון ד. שם קובץ יכול להיות בעברית._____

נכון / לא נכון ה. שם קובץ יכול להכיל סימן רווח. _____

נכון / לא נכון ו. חובה לשים סימן נקודה-פסיק (;) בסוף כל הוראת התכנית. _____

נכון / לא נכון ז. תיעוד לתכנית פירושו שמוסיפים מסמכים ותעודות רשמיים לתכנית. _____

נכון / לא נכון ח. הזחה פירושה הכנסת הכתוב כמה תווי רווח מתחילת השורה. _____

נכון / לא נכון ט. כשכותבים תכנית, אין חובת הזחה._____

נכון / לא נכון י. תיעוד קטע יכול להימשך על פני מספר שורות. _____

נכון / לא נכון יא. מילה שמורה היא מילה שיש לה משמעות ותפקיד מיוחד בשפה. _____

נכון / לא נכון יב. הדרך הנכונה להשיג הזחה בתכנית היא באמצעות מקש tab. _____

נכון / לא נכון יג. כל עוד קיימות שגיאות בתכנית, לא יתקבל קובץ ריצה (בשפת מכונה). _____

נכון / לא נכון יד. חובה שיהיו משתנים בכל תכנית._____

נכון / לא נכון טו. הוראות הקלט והפלט בתכנית, שייכות למחלקה Scanner _____

6. אחדות מהוראות הפלט שלפניך שגויות. סמן אותן והסבר מהן הטעויות:

(1) **System**.out.println ("Programming in JAVA"); _____

(2) **System**.out.println (hello world!); _____

(3) **System**.out.print (" "); _____

(4) **System**.out.print ("System.out.println (hello world !) "); _____

(5) **System**.out.println (' SHANA TOVA '); _____

(6) **System**.out.println "we have not yet begun" ; _____

(7) **System**.out.println ("Have a nice day) ; _____

(8) **System**.out.print ("what a beautiful morning !") _____

(9) **System**.out.println (“שלום לעולם”); _____

7. בחלק מהתכניות הבאות נפלו שגיאות תחביריות. ציין והסבר את השגיאות:

(1)
```
public class Hello {
    public static void main(String[] args){
        System.out.println ("Hello world ! ");
    }
}
```

(2)
```
public class בוקר_טוב {
    public static void main(String[] args) {
        System.out.println (" בוקר טוב עולם! ");
    }
}
```

(3)
```
public class hello {
    public static void main(String[] args) {
        System.out.println ("Hello world ! ");
    }
}
```

(4)
```
public class NiceDay {
    static void main(String[] args) {
        System.out.println ("Have a nice day ! ");
    }
}
```

(5)
```
public class World {
    public static void main() {
        System.out.println ("It's a wonderful world!  ");
    }
}
```

(6)
```
public class Enough {
    public static void Main(String[] args) {
        System.out.println ("That's enough!  ");
    }
}
```

(7)

```
/ *     תכנית מחשב בשפת JAVA
 *      לכבוד חופשת סוכות.
 * /
public class Suka {
    public static void Main(String[] args) {
        System.out.print ("... נהדרת ויפה ")
        System.out.println ("  שלומית בונה סוכה ")
        System.out.println (" שלומית בונה סוכת שלום")
    }
}
```

כיצד יראה הפלט עבור קטע זה אחרי התיקון? הסבר! _____

_____

מה השינוי שיש לבצע בתכנית על מנת שסדר ההדפסה יהיה תואם למילות השיר? _____

_____

(8)

```
/ *     תכנית מחשב בשפת JAVA
 * /
public class Hag {
    public static void main(String[] args) {
        println ("יש לי יום יום חג ");
        println (".יש לי חג יום יום ");
    }
}
```

משפטי השמה

8. כתוב משפטי השמה לביצוע הפעולות הבאות:

פעולה	משפט השמה
א. הקטנת ערכו של משתנה a ב - 5.	_____
ב. הגדלת ערכו של משתנה b פי 3.	_____
ג. הכפלת ערכו של המשתנה a ב- 2.	_____
ד. החסרת ערך המשתנה b מהמשתנה a.	_____
ה. הכפלת ערכו של המשתנה c במשתנה a .	_____
ו. הוספת 5 למשתנה a .	_____
ז. איפוס המשתנה x.	_____
ח. הגדלת ערכו של המשתנה d ב- 8.	_____
ט. הגדלת ערכו של המשתנה c פי 8.	_____
י. הגדלת ערכו של המשתנה a במשתנה b.	_____
יא. הגדלת ערכו של המשתנה h בפעמיים המשתנה s	_____

יב. הקטנת ערכו של משתנה c פי 10. _____

יג. הגדלת ערכו של משתנה d ב- 20%. _____

יד. הקטנת ערכו של משתנה f ב- 30%. _____

9. כתוב הוראות השמה לביצוע הפעולות הבאות:

פעולה	משפט השמה
(1) שים במשתנה a את הסכום המתקבל מחיבור ערכו של המשתנה a לערכו של המשתנה b.	
(2) שים במשתנה c את ההפרש המתקבל מחיסור מכפלת ערכו של המשתנה d ב- 2 מהערך שבמשתנה c	
(3) שים במשתנה e את הסכום המתקבל מחיבור מחצית ערכו של המשתנה e לחמש פעמים ערכו של המשתנה f.	
(4) שים במשתנה g את ההפרש המתקבל מחיסור 80% מערכו של המשתנה g מפעמיים ערכו של המשתנה h.	

כתיבת תכנית מחשב

הוראות הדפסה

התרגילים 10 עד 16, מומלץ תחילה לתכנן את המטלה המבוקשת על נייר משבצות.

10. א. כתוב תכנית מחשב שתדפיס עבורך כרטיס ביקור הכולל את שמך, כתובתך ומספר הטלפון שלך.
ב. הוסף הוראות לתכנית כך שכרטיס הביקור יהיה תחום במסגרת.

11. כתוב תכנית מחשב שתקלוט מן המשתמש את שמו (לתוך משתנה מחרוזת) ואת גילו, ותדפיס עבורו כרטיס ביקור.

12. כתוב תכנית מחשב המשתמשת בהוראות הדפסה, ומציירת על המסך ריבוע מלא מתווי כוכביות. גודל הריבוע לא יהיה קטן מ-7*7.

13. כתוב תכנית מחשב המשתמשת בהוראות הדפסה, ומציירת על המסך ריבוע חלול בעזרת כוכביות. גודל הריבוע לא יהיה קטן מ-7*7.

14. כתוב תכנית מחשב המשתמשת בהוראות הדפסה, ומציירת על המסך שני מלבנים, זה לצד זה, בעזרת כוכביות. גודל כל מלבן לא יהיה קטן מ-5 שורות (אורך) ו- 10 עמודות (רוחב).

15. כתוב תכנית מחשב המשתמשת בהוראות הדפסה, ומציירת את שמך על המסך מתווי כוכביות. ציור השם יתפרס על לפחות מחצית המסך.

16. כתוב תכנית מחשב המשתמשת בהוראות הדפסה, ומציירת ציור כלשהו על המסך, לדוגמא: בית, מכונית וכדי.

קלט, פלט והוראות השמה

17. כתוב תכנית מחשב הקולטת למשתנה מטיפוס int מספר שלם, ומדפיסה אותו באופן הבא:
בשורה ראשונה יודפס המספר פעם אחת.
בשורה השנייה יודפס המספר 2 פעמים.
בשורה השלישית יודפס המספר 3 פעמים.

18. כתוב תוכנית מחשב, בה יוצב במשתנה a המספר 3, ובמשתנה b המספר 5. התכנית תדפיס את הסכום a+b.

19. כתוב תוכנית מחשב, בה יוצב 2 במשתנה a, 3 במשתנה b, ו- 5 במשתנה c. התכנית תדפיס את הערך של $4ac - b$.

20. כתוב תכנית מחשב, לקליטת שני מספרים והדפסת מכפלתם בתוספת לכותרת: "המכפלה:".

21. כתוב תכנית מחשב, לקליטת נתון מספרי למשתנה a והדפסת מספר הגדול ממנו ב- 2.

22. כתוב תכנית מחשב, לקליטת נתון מספרי למשתנה x והדפסת 3 מספרים עוקבים, החל מ- x.

23. כתוב תכנית מחשב, לקליטת נתון מספרי למשתנה x והדפסת 3 מספרים עוקבים, כלה ב- x (x יהיה המספר האחרון).

24. כתוב תכנית מחשב המציבה את הערך 5 במשתנה num. על התכנית לחשב ולהדפיס את המספרים הבאים, כל הדפסה תעשה בשורה נפרדת (כל ההוראות מתייחסות לערך שבמשתנה num):
   א. מספר הגדול ממנו ב- 5.
   ב. מספר הקטן ממנו ב- 10.
   ג. מספר הגדול ממנו פי 7.
   ד. מספר הקטן ממנו בחצי.

25. כתוב תכנית מחשב המציבה את המספרים 7 ו- 3½ במשתנים. על התכנית לחשב ולהדפיס:
   א. את סכומם.
   ב. את הפרשם.
   ג. את מכפלתם.
   ד. את תוצאת החילוק של הראשון בשני.

26. כתוב תכנית מחשב שתאחסן את הערך 5 במשתנה num1 ואת הערך 8 במשתנה num2, על התכנית לחשב ולהציג כפלט:
   א. מספר הגדול ב- 10 מסכום שני המשתנים.
   ב. מספר הקטן פי 3 מהפרש המשתנים.
   ג. מספר הגדול פי 4 מהמשתנה num1.
   ד. מספר הקטן ב- 25 ממשתנה num2.

27. כתוב תכנית מחשב הקולטת למשתנה מטיפוס int מספר שלם, ומדפיסה את הפלט הבא:
בשורה ראשונה יודפס המספר.
בשורה השנייה יודפס המספר, ואחר כך שני המספרים העוקבים לו.
בשורה השלישית יודפס המספר, מכפלתו ב- 2, מכפלתו ב- 3 .
להלן דוגמת פלט עבור הקלט 5:

```
5
5 6 7
5 10 15
```

דף עבודה 3
צבים
שילוב הוראות קלט

1. פתח את המחלקה DrawingWithTurtles שבפרויקט Chap1.

2. בדף העבודה הקודם, ה**מתכנת** הוא זה שקבע את גודל הצעד שיזוז הצב.
עליך להוסיף לתכנית הוראות קלט שיבקשו מה**משתמש** בתכנית את גודל הצעד שיזוז הצב.

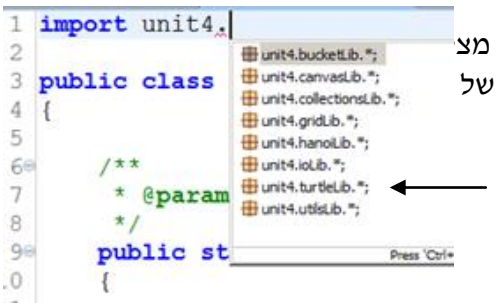
א. הגדר משתנה מטיפוס מספר שלם (**int**) בשם step, וכתוב הוראת קלט שתקלוט לתוכו את גודל הצעד שיבצע הצב (גודל הצעד = אורך הקו שיצייר הצב).

ב. שנה את ההוראה הגורמת לצב לנוע קדימה, מ-
```
t1.moveForward (100);
```
ל-
```
t1.moveForward (step);
```

כדי ליצור תכניות חדשות המציירות בעזרת הצב, נפעל באופן הבא:

א. בתוך הפרויקט ניצור מחלקה חדשה, ניתן לה שם מתאים לתוכן התכנית, ונסמן ☑ ליד ההוראה ליצירת הפעולה main.

ב. נרשום בשורה הראשונה הוראה לקישור ל- unit4 המגדיר את הצב, באופן הבא:
נרשום את המילים import unit4.
לאחר שנרשום נקודה, נשים לב שeclipse מצ
לבחור את הקובץ המתאים מתוך רשימה של
חבילות תוכנה.
(אם הרשימה לא מופיעה, נסה ללחוץ על צירוף
המקשים <מקש הרווח> + <Ctrl>)



נבחר בחבילת התוכנה: tutrleLib.*

לתכנית התווספה השורה:
```
import unit4.turtleLib.*;
```

ג. כדי ליצור צב חדש נרשום בתוך התכנית (בתוך הפעולה main) את ההוראה הבאה:
```
Turtle t1 = new Turtle ();
```

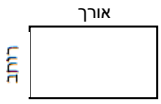
שים לב לגודל אות, לסוגריים ולסימן ה- ;

שם הצב שיצרנו הוא t1. ניתן לבחור שם אחר, ובלבד שיעמוד בכללים של מתן שם למשתנה.

```
import unit4.turtleLib.*;

/**
 * תכנית לציור ריבוע בעזרת הצב
 */
public class TurtleSquare
{
    public static void main(String[] args)
    {
        Turtle t1 = new Turtle ();

    }

}
```

3. כתוב שתגרום לצב לצייר ריבוע. על התכנית ליצור צב חדש, לקלוט עבורו את אורך צלע הריבוע למשתנה בשם side, ולצייר ריבוע שאורך צלעו הוא side.

4. כתוב שתגרום לצב לצייר מלבן. על התכנית ליצור צב חדש, לקלוט עבורו את אורך צלע המלבן למשתנה בשם side, ולצייר מבלן שאורכו side ורוחבו פי 2 מרוחבו.



5. על מנת לצייר מצולע משוכלל (שאורך כל צלעותיו שוות זו לזו) בעל n צלעות, יש לגרום לצב לפנות בכל פעם 360/n מעלות ימינה (או שמאלה).

לדוגמה: עבור מחומש משוכלל, יש לצייר 5 צלעות, וזווית הפנייה היא: 360/5 = 72

עבור משושה משוכלל, יש לצייר 6 צלעות, וזווית הפנייה היא: 360/6 = 60

כתוב תכנית שתקלוט את מספר הצלעות במצולע למשתנה n, ואת גודל הצלע למשתנה side, ויצייר מצולע בעל n צלעות באורך side כל אחת.

עבודה נעימה !