פרק 4 – הרחבה בפיתוח אלגוריתמים

בפרק הקודם הצגנו את התהליך של פיתוח ויישום אלגוריתם בשלבים. בפרק זה נרחיב בכמה מהשלבים: בניתוח הבעיה, בפירוק המשימה לתת-משימות, ובבדיקת הפתרון עבור דוגמאות קלט שונות. הבעיות שיוצגו בפרק זה יהיו מורכבות יותר מהבעיות שהוצגו בפרק הקודם, וכך, ככל שנתקדם בפרקי הלימוד, הבעיות יהיו מורכבות יותר ויותר, וחשיבות הפתרון בשלבים תהיה משמעותית יותר ויותר.

באמצעות הבעיות שיוצגו בפרק זה ופתרונן, נכיר גם פעולות נוספות על ערכים מטיפוס שלם, טיפוס חדש שערכיו הם תווים, ואת המחלקה האחראית לפעולות מתמטיות בשפת Java.

4.1 מבט נוסף אל התהליך של פיתוח אלגוריתם ויישומו

כזכור, תהליך של פיתוח אלגוריתם ויישומו בתוכנית מחשב כולל את השלבים הבאים:

1. ניתוח ראשוני של הבעיה בעזרת דוגמאות
2. פירוק הבעיה לתת-משימות
3. בחירת משתנים, הגדרת תפקידיהם וטיפוסי הערכים שיישמרו בהם
4. כתיבת האלגוריתם
5. יישום האלגוריתם בתוכנית מחשב
6. בדיקת נכונות עבור דוגמאות קלט מגוונות בטבלת מעקב
7. כתיבת התוכנית המלאה, ובדיקתה בהרצה על דוגמאות קלט נוספות

בעת פיתוח האלגוריתם אנו נשפר ונשנה אותו. לעתים בעת העבודה על שלב, מתברר כי יש לתקן שלב קודם (לדוגמה, בעת כתיבת האלגוריתם, מסתבר כי יש להוסיף משתנה). במקרה כזה נחזור אחורה לשלב הקודם ונתקן לפני שנמשיך לשלב הבא. כמו כן, בבדיקת התוכנית עלולות להתגלות שגיאות, ותיקונן עשוי להביא לבחירת משתנים נוספים ולשינוי הוראות האלגוריתם. שיפור של אלגוריתם ושל תוכנית או תיקונים תוך חזרה משלב מתקדם לשלב קודם הוא תהליך טבעי ומקובל.

שימו ♥ לשני השלבים האחרונים:
בשלב 6 נבדקת נכונות התוכנית באמצעות מעקב "ידני" אחר מהלך הביצוע עבור מספר מצומצם של דוגמאות קלט; זאת על מנת לאמת ולקבל ביטחון ראשוני שהתוכנית אכן משיגה את מטרתה המיועדת.

בשלב 7 מורצת התוכנית המלאה במחשב. ההרצה מאפשרת בדיקה מלאה יותר של התוכנית, כיוון שניתן להריץ את התוכנית במהירות עבור דוגמאות קלט רבות ומגוונות.

בעיה 1

מטרת הבעיה הבאה ופתרונה: הדגמה מפורטת של פיתוח ויישום בשלבים של אלגוריתם.

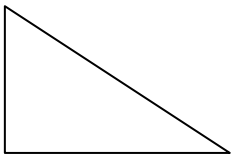
טלי ואודי מעוניינים לבנות גלגלת שתחבר בין החלונות שלהם. הגלגלת מורכבת מחבל כפול באורך של המרחק שבין החלונות. על טלי למדוד את המרחק שבין החלונות כדי לדעת מה אורך החבל שיזדקקו לו. טלי בדקה ומצאה שכדי להגיע מהחלון שלה לחלון של אודי עליה לצעוד ישר מספר צעדים, לפנות ימינה ולצעוד מספר צעדים נוספים. טלי גם בדקה ומצאה שאורך כל צעד שלה הוא 42 ס"מ. (0.42 מ').

פתחו אלגוריתם שהקלט שלו הוא מספר הצעדים שעל טלי לצעוד ישר, ומספר הצעדים שעל טלי לצעוד לאחר שפנתה ימינה, והפלט שלו הוא אורך החבל המבוקש. ישמו את האלגוריתם בתוכנית מחשב בשפת Java.

ניתוח הבעיה בעזרת דוגמאות

נבדוק עבור מספר דוגמאות מה הפלט הרצוי עבור הקלט:

אם טלי צועדת 30 צעדים ישר, ואחר כך 21 צעדים ימינה, ניתן לתאר זאת באמצעות משולש ישר זווית, שניצב אחד שלו הוא 30 צעדים, והשני 21 צעדים. החלון של טלי הוא הקדקוד התחתון הימני של המשולש, והחלון של אודי הוא הקדקוד העליון של המשולש.



המרחק המבוקש הוא בדיוק אורך היתר במשולש זה. לכן, נשתמש במשפט פיתגורס, האומר כי אורך יתר במשולש ישר זווית הוא שורש סכום ריבועי שני הניצבים, כלומר: $c = \sqrt{a^2 + b^2}$

לכן, אם טלי צועדת 30 צעדים ישר, ואחר כך 21 צעדים ימינה, ניתן לחשב את אורך החבל המבוקש באופן הבא: $\sqrt{(30 \cdot 0.42)^2 + (21 \cdot 0.42)^2} = 15.38$.

לבסוף נכפיל את התוצאה ב-2 (כיוון שהחבל הינו כפול) והערך המתקבל הוא הפלט הנדרש.

ומה אם הנתון השני בקלט הוא 0? כלומר, טלי אינה צריכה לצעוד כלל ימינה, משום שהחלון של אודי נמצא בדיוק מול החלון שלה? במקרה זה, ברור כי המרחק המבוקש ניתן לחישוב ישירות לפי מספר הצעדים הנתון בקלט, אבל מסתבר שהנוסחה הכללית שמצאנו תתאים גם למקרה הפרטי הזה:

$$2 \cdot \sqrt{(30 \cdot 0.42)^2 + (0 \cdot 0.42)^2} = 2 \cdot \sqrt{(30 \cdot 0.42)^2} = 2 \cdot 30 \cdot 0.42$$

תהליך בדיקת הפלט עבור דוגמאות שונות עוזר לנו להבין את מהות הבעיה, להבין מה נדרש מאתנו, לחשוב על התהליכים הדרושים לפתרון הבעיה, ובכך מכוון אותנו לקראת השלבים הבאים של תכנון ושל כתיבת האלגוריתם עצמו.

פירוק הבעיה לתת-משימות

נפרק את הבעיה לשלוש תת-משימות:

1. קליטת שני מספרים חיוביים שלמים
2. חישוב אורך החבל הנדרש
3. הצגה של אורך החבל כפלט

התת-משימה הראשונה והשלישית הן פשוטות. התת-משימה השנייה היא התת-משימה העיקרית והמורכבת יותר. ניתן לפרק גם אותה לתת-משימות ובכך לפרט יותר את הרעיון לפתרון הבעיה:

2.1. הכפלת המספר הראשון ב-0.42, והעלאת התוצאה בריבוע
2.2. הכפלת המספר השני ב-0.42, והעלאת התוצאה בריבוע
2.3. חיבור שני הערכים שהתקבלו והוצאת שורש מהסכום
2.4. הכפלת הערך שהתקבל ב-2

כל הפעולות המפורטות כאן הן פעולות חישוב פשוטות לביצוע, שאינן דורשות ניתוח נוסף. כעת, לאחר פירוק הבעיה לתת-משימות, ברור לנו הרעיון לפתרון הבעיה.

בחירת משתנים

שלושת המשתנים הראשונים שנבחר דרושים עבור פעולות הקלט והפלט:

stepForward – שלם, ישמור את מספר הצעדים קדימה

stepRight – שלם, ישמור את מספר הצעדים ימינה

ropeLength – ממשי, ישמור את אורך החבל הנדרש הסופי

נשתמש בעוד שני משתנים לשמירת ערכי הביניים של החישובים השונים:

side1 – ממשי, ישמור את ריבוע האורך של ניצב אחד במשולש

side2 – ממשי, ישמור את ריבוע האורך של ניצב שני במשולש

האלגוריתם

לפי החלוקה לתת-משימות ותוך שימוש במשתנים שבחרנו, נקבל את האלגוריתם הבא:

1. *קלוט מספר שלם וייבי ב-*stepForward
2. *קלוט מספר שלם וייבי ב-*stepRight
3. *חשב את מכפלת* stepForward *ב-0.42 והשם ב-*side1
4. *חשב את מכפלת* stepRight *ב-0.42 והשם ב-*side2
5. *העלה את* side1 *בריבוע*
6. *העלה את* side2 *בריבוע*
7. *חשב את סכום הערכים שהתקבלו*
8. *חשב את השורש הריבועי של הערך שהתקבל*
9. *הכפל את הערך שהתקבל ב-2 והשם ב-*ropeLength
10. *הדפס כפלט את ערכו של* ropeLength

יישום האלגוריתם

הוראה 8 באלגוריתם כוללת חישוב שורש ריבועי. כיצד מבצעים זאת בשפת Java?

את הוראות 5 עד 8 ניתן לבטא בביטוי:

```
Math.sqrt((side1 * side1) + (side2 * side2));
```

המחלקה הכוללת פעולות מתמטיות שונות

הפעולה לחישוב השורש הריבועי של מספר

הפעולה `sqrt` מקבלת בתוך הסוגריים מספר מטיפוס ממשי, ומחזירה מספר ממשי השווה לשורש הריבועי של המספר שקיבלה.

פעולה זו שייכת למחלקה `Math`. מחלקה זו מוכרת בשפת Java, ומוכנה לשימוש. היא נמצאת במארז של המחלקות הבסיסיות של השפה ולכן לא צריך להודיע למהדר באופן מפורש כי אנו מתכוונים להשתמש בה.

את הערך 0.42 נגדיר כקבוע, כדי ליצור תוכנית קריאה ועמידה בפני שינויים.

הנה היישום של הוראות האלגוריתם:

```
1. System.out.print("Enter number of steps forward: ");
2. stepForward = in.nextInt();
3. System.out.print("Enter number of steps to the right: ");
4. stepRight = in.nextInt();
5. side1 = stepForward * STEP_SIZE;
```

```
6. side2 = stepRight * STEP_SIZE;
7. ropeLength = 2 * Math.sqrt((side1 * side1) + (side2 * side2));
8. System.out.println ("The length of the rope is: " + ropeLength);
```

מעקב

נעקוב אחר מהלך ביצוע משפטי התוכנית עבור הקלט 21 30:

שורה לביצוע	stepForward	stepRight	side1	side2	ropeLength	פלט
1	30	?	?	?	?	Enter number of steps forward
2	30	21	?	?	?	
3	30	21	?	?	?	Enter number of steps to the right
4	30	21	12.6	?	?	
5	30	21	12.6	?	?	
6	30	21	12.6	8.82	?	
7	30	21	12.6	8.82	30.76	
8	30	21	12.6	8.82	30.76	The length of the rope is: 30.76

על פי המעקב הזה התוכנית מבצעת את מטרתה.

התוכנית המלאה

```
/* קלט: מספר הצעדים בין הבתים של טלי ושל אודי
פלט: אורך החבל הנדרש */
import java.util.Scanner;
public class Rope
{
    public static void main (String[] args)
    {
        Scanner in = new Scanner(System.in);
        // הגדרת משתנים
        int stepForward, stepRight;  // המספרים הניתנים כקלט
        double side1, side2;         // ערכי הניצבים של המשולש
        double ropeLength;           // אורך החבל
        final double STEP_SIZE = 0.42;     // קבוע - גודל צעד
        // קלט
        System.out.print("Enter number of steps forward: ");
        stepForward = in.nextInt();
        System.out.print("Enter number of steps to the right: ");
        stepRight = in.nextInt();
        // חישוב אורך החבל
        side1 = stepForward * STEP_SIZE;
        side2 = stepRight * STEP_SIZE;
        ropeLength = 2 * Math.sqrt((side1 * side1) +(side2 * side2));
        // פלט
        System.out.println ("The rope length is: " +  ropeLength);
    } // main
} // class Rope
```

סוף פתרון בעיה 1

פתרון הבעיות הבאות בפרק, בעיות מורכבות יותר מבעיה 1, יסתמך על התהליך שהצגנו. מעתה ואילך נניח שכאשר נדרשים פיתוח אלגוריתם ויישומו בשפת תכנות, נדרש פיתוח בשלבים, ולכן לא נציין זאת במפורש.

שאלה 4.1

דן, בן וחן קיבלו כל אחד דמי חנוכה מהוריהם. כל אחד קיבל סכום הגדול מ-20 ₪. שלושת החברים החליטו לאחד את כל סכום הכסף לקופה אחת ולקנות יחדיו כדורסל שמחירו 50 ₪. ביתרת הכסף יקנו מסטיקים שעלותם 1 ₪ כל אחד. פתחו אלגוריתם המקבל כקלט את דמי החנוכה שקיבל כל אחד מהחברים, ומציג כפלט את כמות המסטיקים שאפשר לקנות. ישמו את האלגוריתם בתוכנית בשפת Java.

המחלקה המתמטית

בפתרון בעיה 1 חישבנו שורש ריבועי על ידי שימוש בפעולה `Math.sqrt`.

זהו שמה המלא של הפעולה `sqrt` השייכת למחלקה המתמטית `Math`. כאמור, לפעולה אנו מעבירים ערך (פרמטר) שהוא מטיפוס ממשי, והיא מחזירה את השורש הריבועי של המספר שניתן לה. טיפוס הערך המוחזר הוא ממשי.

המחלקה המתמטית `Math` מכילה עוד פעולות מתמטיות רבות אשר נתונות לשימושנו, וכמוה יש מחלקות נוספות שנוכל להשתמש בהן לפעולות עזר מסוגים שונים.

פעולות שימושיות מהמחלקה המתמטית `Math`

הפעולה	תיאור הפעולה	טיפוסי פרמטרים	טיפוס ערך מוחזר	דוגמה	
				הפעולה	הערך המוחזר
`abs(num)`	ערך מוחלט	שלם	שלם	`Math.abs(63)`	63
		ממשי	ממשי	`Math.abs(-12.7)`	12.7
`sqrt(num)`	שורש ריבועי	ממשי	ממשי	`Math.sqrt(6.25)`	2.5
`pow(num1,num2)`	חזקה $num1^{num2}$	ממשי,ממשי	ממשי	`Math.pow(3,2)`	9.0
`min(num1,num2)`	הקטן מבין השניים	שלם, שלם	שלם	`Math.min(3,8)`	3
		ממשי,ממשי	ממשי	`Math.min(8.0,8.8)`	8.0
`max(num1,num2)`	הגדול מבין השניים	שלם, שלם	שלם	`Math.max(3,8)`	8
		ממשי,ממשי	ממשי	`Math.max(8.0,8.8)`	8.8
`round(num)`	עיגול מספר ממשי	ממשי	שלם	`Math.round(7.9)`	8

בנוסף לפעולות המתוארות בטבלה זו שייכות למחלקה `Math` פעולות רבות נוספות. אתם מוזמנים לחפש ברשת ולעיין בממשק של המחלקה המתמטית בשפת Java (`java.lang.Math`) ולמצוא פעולות נוספות השייכות למחלקה זו.

עד כה הכרנו כמה מחלקות שימושיות: המחלקה `System` שאחראית לפעולות מערכת כלליות (בתוכה מוגדר נתיב הפלט `out`); המחלקה `Scanner` האחראית לפעולות הקלט; המחלקה `Math` האחראית לפעולות מתמטיות. בהמשך לימודיכם תפגשו עוד מחלקות רבות המשמשות לצרכים שונים.

שאלה 4.2

פתחו אלגוריתם אשר הקלט שלו הוא גובהי שני תלמידים, נתונים במספרים ממשיים, והפלט שלו הוא הערך המוחלט של הפרשי הגבהים שלהם. ישמו את האלגוריתם בתוכנית בשפת Java. שימו לב לבחירת דוגמאות קלט מגוונות.

שאלה 4.3

שנו את התוכנית שכתבתם כפתרון לשאלה 4.2 כך שפלט התוכנית יהיה גובה התלמיד הנמוך מבין השניים.

שאלה 4.4

פתחו אלגוריתם אשר הקלט שלו הוא אורך ורוחב צלעות של מלבן (מספרים שלמים) והפלט שלו הוא שטח המלבן ואורך אלכסון המלבן.

4.2 פעולות חלוקה בשלמים

בפרק 3 אמרנו כי הגדרת טיפוס כוללת גם את פירוט הפעולות הניתנות לביצוע על ערכי הטיפוס. בפרק 3 הצגנו את הטיפוסים שלם וממשי, ורשימת הפעולות שהצגנו עבור כל אחד מהם כללה את הפעולות החשבוניות המוכרות. בסעיף זה נכיר פעולות חשבוניות המוגדרות רק עבור ערכים מטיפוס שלם.

בעיה 2

מטרת הבעיה הבאה ופתרונה: שימוש בפעולות לחישוב מנה ושארית בחלוקת מספרים שלמים.

ליונתן אוסף מטופח של גולות. הוא שומר את כל האוסף בקופסה, בקבוצות של 20, כלומר, כל קבוצה של 20 גולות ארוזה בשקית נפרדת. הגולות הנותרות מפוזרות בתחתית הקופסה. פתחו בשלבים אלגוריתם אשר הקלט שלו הוא מספר הגולות שיש ליונתן, והפלט הוא מספר השקיות שיש בקופסה ומספר הגולות המפוזרות בתחתיתה. ישמו את האלגוריתם בתוכנית מחשב בשפת Java.

למשל, עבור הקלט 135, הפלט הדרוש הוא: 15 6 כלומר, 6 שקיות ו-15 גולות פזורות.
(משום ש-135=15+20*6).

ניתוח הבעיה בעזרת דוגמאות

נבדוק את הפלט הרצוי עבור כמה דוגמאות קלט:

קלט	פלט
13	0 שקיות ו-13 פזורות
60	3 שקיות ו-0 פזורות
64	3 שקיות ו-4 פזורות
82	השלימו: _____ שקיות ו-_____ פזורות

על פי הדוגמאות אנו רואים כי מספר השקיות הוא מספר הפעמים שנכנס המספר 20 במספר הגולות הכולל, שנקלט מהקלט. הגולות הפזורות הן השארית, אלה שנותרו אחרי שאספנו קבוצות של 20 גולות ככל שיכולנו.

כלומר עלינו לבצע כאן שתי פעולות, כל אחת מהן מתבצעת על מספרים שלמים ומחזירה מספר שלם. הפעולה הראשונה מקבלת מספר x (במקרה שלנו, המספר הכולל של הגולות) ומספר y (במקרה שלנו, מספר הגולות בכל שקית, כלומר 20) ומחזירה את מספר הפעמים שהמספר y נכנס במספר x. הפעולה השנייה משלימה אותה, במובן מסוים: היא מקבלת x ו-y (כמו קודם) ומחזירה את מה שנותר אחרי שמחסירים מ-x כפולות שלמות של y ככל שניתן.

אלו הן פעולות המוגדרות על מספרים שלמים, כלומר על ערכים מטיפוס שלם: הפעולה הראשונה מחשבת את ה**מנה** של x חֶלְקֵי y. הפעולה השנייה היא פעולת ה**שארית (modulus).**

בעברית מנה היא תוצאה של פעולת חלוקה. קיים קשר בין הפעולות האלו לפעולת החלוקה הרגילה. כאמור, אם נבצע חלוקה רגילה של מספר הגולות x במספר 20, לא נקבל בהכרח מספר שלם. למשל, אם יש 105 גולות, 105/20=5.25. אבל אם ננסה למלא מתוך 105 גולות כמה שיותר שקיות של 20 גולות, נצליח למלא בדיוק 5 שקיות. זהו בדיוק החלק השלם של תוצאת החלוקה 105/20.

שתי פעולות אלו נקראות פעולות חלוקה בשלמים:

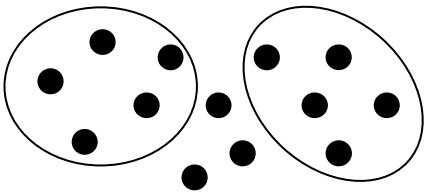
פעולות החלוקה בשלמים מוגדרות רק על מספרים שלמים, וגם תוצאת הפעלתן היא מספר שלם:

מנת החלוקה של מספר שלם x במספר שלם y שווה לחלק השלם של x/y, ובמילים אחרות, שווה למספר הפעמים שהערך y נכנס בערך x.

שארית החלוקה (modulus) של מספר שלם x במספר שלם y מבטאת את השארית הנותרת לאחר חלוקה בשלמים של x ב-y, ובמילים אחרות, מבטאת את מה שנותר אחרי שמפחיתים מ-x כפולות שלמות של y ככל שניתן.

השוני בין פעולות חלוקה בשלמים ובין פעולת החלוקה במספרים ממשיים מדגים את האבחנה בין הטיפוסים שלם לממשי.

למשל מנת החלוקה של 13 ב-5 שווה ל-2, משום ש-5 נכנס ב-13 פעמיים (ואכן 13/5=2.6 והחלק השלם הוא 2). לעומת זאת, שארית החלוקה של 13 ב-5 שווה ל-3, משום שאחרי שנפחית מ-13 שתי כפולות של 5 (כלומר 10), נקבל 3. נדגים זאת בעזרת האיור הבא, המציג חלוקת 13 נקודות לקבוצות של 5:



בחלוקה של 13 לקבוצות של 5 אנו אכן מקבלים כי מנת החלוקה היא 2, כיוון שהתקבלו שתי קבוצות שלמות בגודל 5 כל אחת. שארית החלוקה היא 3, כיוון שנשארו שלוש נקודות שלא שייכות לאף קבוצה.

את בעיית החלוקה הזו ניתן לראות באופן נוסף: אם נרצה לחלק 13 סוכריות באופן שווה ל-5 ילדים, כמה סוכריות יקבל כל ילד? כמה יישארו לנו? והתשובה תהיה זהה – כל ילד יקבל 2 סוכריות, ולנו יישארו 3.

פירוק הבעיה לתת-משימות

נפרק את הבעיה לארבע תת-משימות:

1. קליטת מספר הגולות
2. חישוב מספר השקיות
3. חישוב מספר הגולות הפזורות
4. הצגה של מספר השקיות ושל הגולות הפזורות כפלט.

הערכים בעיבוד הנתונים הם מספרים שלמים. אין לנו כאן עניין במספרים ממשיים, ולכן הפעולה שנרצה להשתמש בה לביצוע התת-משימה השנייה והשלישית איננה פעולת חלוקה רגילה, כיוון שפעולה כזאת תיתן תוצאה מטיפוס ממשי. למשל תוצאה של 64/20 תיתן 3.2, אך מספר זה אינו נותן לנו את מספר השקיות המלאות ואת מספר הגולות הפזורות. הפעולות המתאימות הן פעולות החלוקה בשלמים שהוצגו לעיל.

את מספר השקיות המלאות נחשב כמנה של חלוקת מספר הגולות ב-20; את מספר הגולות הפזורות נחשב כשארית של אותה חלוקה. שתי פעולות אלה הן שתי פנים של תבנית שימושית: חלוקת כמות פריטים לקבוצות בגודל נתון.

בחירת משתנים

amount – שלם, ישמור את המספר הניתן כקלט, מספר הגולות.
bags – שלם, ישמור את מספר השקיות המכילות 20 גולות כל אחת.
remainder – שלם, ישמור את מספר הגולות שנשארו פזורות.

האלגוריתם

1. *קלוט את מספר הגולות ב-*amount
2. *חשב את מספר השקיות המלאות בחישוב מנת החלוקה של* amount *ב-20 והשם את התוצאה ב-*bags
3. *חשב את מספר הגולות שנותרו פזורות בחישוב שארית החלוקה של* amount *ב-20 והשם את התוצאה ב-*remainder
4. *הצג כפלט את הערך* bags *ואת הערך* remainder

יישום האלגוריתם

בשפת Java, הפעולה לחישוב שארית החלוקה של הערך a בערך b נכתבת בצורה a % b, כאשר גם a וגם b הם מטיפוס שלם.

למשל: 7%3=1, 3%5=3, ו-12%4=0

בשפת Java, הפעולה לחישוב מנת החלוקה של הערך a בערך b נכתבת בצורה a / b, בתנאי שגם a וגם b הם מטיפוס שלם.

למשל: 7/3=2, 13/5=2 ו-12/4=3.

שימו לב שאותו סימן משמש ב-Java הן לפעולת חלוקה במספרים ממשיים והן לפעולת חישוב של מנת החלוקה בשלמים. אם כך, איך נבדיל בין פעולת חישוב של מנת החלוקה בשלמים לבין פעולת חלוקה בממשיים? איך נדע אם הביטוי 13/5 ערכו שווה ל-2 או ל-2.6?

סוג החלוקה נקבע על פי חוק פשוט:

חלוקת שלם בשלם תתפרש **תמיד** כפעולת חלוקה בשלמים.
כל אחת מהחלוקות הבאות מתפרשת כפעולת חלוקה של מספרים ממשיים:
שלם/ממשי
ממשי/שלם
ממשי/ממשי
למשל 13/5=2, 13.0/5=2.6 ו-13/5.0=2.6.

ניישם את האלגוריתם בשימוש בפעולות החלוקה בשלמים בשפה, ובהגדרת קבוע (PER_BAG) עבור מספר הגולות בשקית:

```
1.  System.out.print("Enter amount of marbles: ");
2.  amount = in.nextInt();
3.  bags = amount / PER_BAG;
4.  remainder = amount % PER_BAG;
5.  System.out.println("There are " + bags + " bags, " + remainder +
                                                    " are left over");
```

מעקב

נעקוב אחר מהלך ביצוע משפטי התוכנית עבור הקלט 92:

משפט לביצוע	number	bags	remainder	פלט
1 System.out...	?	?	?	Enter amount of marbles:
2 amount = in.nextInt()	92	?	?	
3 bags = amount / PER_BAG	92	4	?	
4 remainder = amount % PER_BAG	92	4	12	
5 System.out.println(…)	92	4	12	There are 4 bags, 12 are left over

התוכנית המלאה

```
/*
התוכנית קולטת את מספר הגולות שיש ליונתן
ונותנת כפלט את מספר השקיות המלאות, ואת מספר הגולות שנותרו פזורות
*/
import java.util.Scanner;
public class MarblesInBox
{
    public static void main (String[] args)
    {
        Scanner in = new Scanner(System.in);
        // הגדרת משתנים
        int amount;                    // כמות הגולות
```

```
        int bags;                    // מספר שקיות
        int remainder;               // מספר גולות פזורות
        final int PER_BAG = 20;      // מספר הגולות בשקית אחת
        // קלט
        System.out.print("Enter amount of marbles: ");
        amount = in.nextInt();
        // חישוב מספר השקיות
        bags = amount / PER_BAG;
        // חישוב מספר הגולות הפזורות
        remainder = amount % PER_BAG;
        // פלט
        System.out.println("There are " + bags + " bags, and " +
                                remainder + " marbles are left over");
    } // main
} // class MarblesInBox
```

סוף פתרון בעיה 2

שאלה 4.5

בנו טבלת מעקב אחר מהלך ביצוע התוכנית `MarblesInBox` עבור הקלט 123.

שאלה 4.6

יונתן בעל הגולות החליט לשדרג את אופן האחסון של האוסף ושינה את פקודות התוכנית שבפתרון בעיה 2 כך שגם מספר הגולות בשקית יהיה חלק מהקלט, ולא קבוע:

```
System.out.print("Enter amount of marbles: ");
amount = in.nextInt();
System.out.print("Enter capacity: ");
capacity = in.nextInt();
bags = amount / capacity;
remainder = amount % capacity;
System.out.println("There are " + bags + " bags, " + remainder +
                                                 " are left over");
```

מה יהיה פלט הפתרון המשודרג לבעיית אחסון הגולות של יונתן עבור הקלט 30 125? מה יהיה הפלט עבור הקלט 50 200?

שאלה 4.7

השלימו את תוצאות הפעולות הבאות של חלוקה בשלמים:

א. 7/3 = _____

ב. 7%3 = _____

ג. 20/4 = _____

ד. 20%4 = _____

ה. 3/7 = _____

ו. 3%7 = _____

שאלה 4.8

נתון קטע התוכנית הבא, שהקלט שלו הוא שני מספרים שלמים, והמשתנים בו הם מטיפוס שלם:

```
System.out.print("Enter first number: ");
num1 = in.nextInt();
System.out.print("Enter second number: ");
```

```
num2 = in.nextInt();
a = num1 / num2;
b = num1 % num2;
System.out.println("a: " + a + " b: " + b);
```

א. רשמו את פלט קטע התוכנית עבור כל אחד מהקלטים הבאים (משמאל לימין):

1. 5 2 _____
2. 5 4 _____
3. 5 5 _____

ב. תנו דוגמה לקלט עבורו הפלט יהיה `a: 2 b: 2` _____

להעמקה בתבנית ***חלוקת כמות פריטים לקבוצות בגודל נתון*** פנו לסעיף התבניות המופיע בסוף הפרק.

עוד על פעולת השארית

❓ בשאלה 4.7, בסעיף ד, התקבל הערך 0. מה משמעות הביטוי 20%4=0? באופן כללי, מתי פעולת שארית נותנת תוצאה השווה ל-0?

התוצאה 0 מצביעה על כך שהמספר הראשון (במקרה זה 20) מתחלק במספר השני (במקרה זה 4) **ללא שארית**. כל ומר, המספר 4 הוא אחד המחלקים של 20.

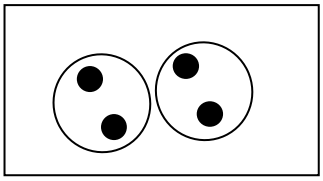
אם כך כיצד נבדוק אם מספר נתון הוא זוגי? מספר זוגי הוא מספר המתחלק ב-2 ללא שארית. לכן, נוכל לחשב את שארית החלוקה של המספר הנתון ב-2. אם המספר אכן זוגי השארית שנקבל תהיה שווה ל-0.

שימו ♥: שארית החלוקה של ערך x בערך y **אינה** שווה לחלק הלא שלם המתקבל מפעולת החלוקה בממשיים של x ב-y. למשל, אם נחשב 13/5 בפעולת חלוקה ממשית נקבל 2.6. לעומת זאת, שארית החלוקה 13%5 היא 3, ולא 6.

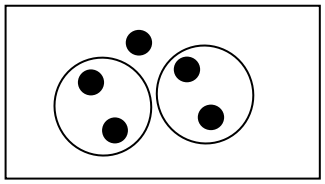
❓ אם נחלק מספר ב-2 ונבדוק את השארית. אילו מספרים יכולים להתקבל כתוצאה?
נבדוק את המקרים הבאים:

4 % 2 = 0
5 % 2 = 1
6 % 2 = 0
7 % 2 = 1

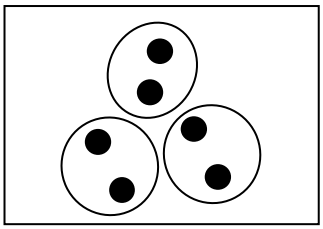
המספרים האפשריים כתוצאה הם אך ורק 0 ו-1. לא ייתכן כי נקבל תוצאה הגדולה מ-1. נראה זאת שוב בעזרת איור: כאשר אנו מחלקים מספר כלשהו של נקודות לקבוצות של שתי נקודות כל אחת, ובודקים כמה נקודות לא נכללות באף קבוצה, ייתכן כי לא נשארת אף נקודה ללא קבוצה, למשל, עבור 2 % 4:



ייתכן כי נשארת נקודה בודדת ללא קבוצה, למשל, עבור 2 % 5:



לא ייתכן שיישארו שלוש נקודות מחוץ לקבוצות, משום שאז ניתן לקחת שתיים מהן וליצור קבוצה חדשה: אם נגדיל את מספר הנקודות ב-1 ונחשב את 2 % 6, נראה כי נוצרה קבוצה חדשה מהנקודה הבודדת שנותרה קודם ומהנקודה החדשה:



אם כך, השארית של חלוקת מספר כלשהו ב-2 לא יכולה להיות גדולה מ-1.
מה השארית של חלוקת מספר כלשהו ב-3? באותו אופן בדיוק ניתן להראות כי השארית של חלוקת מספר כלשהו ב-3 לא יכולה להיות גדולה מ-2.
ובאופן כללי:

הערכים האפשריים לשארית של חלוקת מספר כלשהו ב-n הם בין 0 ל-(n-1).

שאלה 4.9

השלימו:

א. הערכים האפשריים לשארית החלוקה של מספר כלשהו ב-5 הם _____ .

ב. הערכים האפשריים לשארית החלוקה של מספר כלשהו ב-6 הם _____ .

שאלה 4.10

נתון קטע התוכנית הבא אשר הקלט שלו הוא פרק זמן הנתון בשעות, השמור במשתנה `hours`, והפלט שלו הוא מספר היממות השלמות והשעות הנותרות בפרק הזמן הנתון.
השלימו את קטע התוכנית.

```
// קלט
hours = in.nextInt();
days = _____ ;
hoursLeft = _____ ;
// פלט
System.out.println("There are " + days + "days and " + hoursLeft +
                                                        "hours");
```

שאלה 4.11

פתחו אלגוריתם שהקלט שלו הוא פרק זמן הנתון בדקות, והפלט שלו הוא מרכיבי השעות והדקות בפרק זמן זה. ישמו אותו בשפת התכנות Java.

למשל, עבור הקלט 90 הפלט יהיה: 1 שעות, 30 דקות.

עבור הקלט 30 הפלט יהיה: 0 שעות, 30 דקות.

עבור הקלט 300 הפלט יהיה: 5 שעות, 0 דקות.

שאלה 4.12

טייס חלל יוצא ביום ראשון לטיסה ממושכת בחלל. עבור כל יום טיסה, יש לצייד את הטייס ב-3.8 ליטרים של מים.

פתחו אלגוריתם שהקלט שלו הוא מספר ימי הטיסה בחלל, והפלט שלו הוא מספר השבתות שייעדר הטייס מביתו, וכמות המים שיש לציידו לקראת הטיסה. ישמו את האלגוריתם בשפת התכנות Java.

שאלה 4.13

פקיד חרוץ ממלא טופס במשך 10 דקות. עבור כל טופס שהפקיד ממלא הוא מקבל שכר של 6.3 ₪. הפקיד עובד ללא הפוגה.

פתחו בשלבים אלגוריתם אשר הקלט שלו הוא מספר הטפסים שעל הפקיד למלא, והפלט שלו הוא משך העבודה, בשעות ובדקות, והשכר שהפקיד יקבל. ישמו את האלגוריתם בשפת Java.

למשל עבור הקלט 55 יהיה הפלט: משך העבודה הוא: 9 שעות ו-10 דקות והשכר הוא: 346.5.

שאלה 4.14

בפס ייצור במפעל לייצור משאיות מרכיבים גלגלים במשאיות.

פתחו בשלבים אלגוריתם אשר הקלט שלו הוא מספר הגלגלים הכולל בפס הייצור ומספר גלגלים למשאית, והפלט שלו הוא מספר המשאיות שיורכבו להן גלגלים.

למשל עבור הקלט 6 1000, שפירושו 1000 גלגלים בסך הכול בפס הייצור, ו-6 גלגלים למשאית, יהיה הפלט 166.

המרת ערך שלם לממשי

ראובן החליט, בנדיבות לבו, כי את דמי החנוכה שהוא מקבל מסבתו, הוא יחלק באופן שווה בין החברים שיגיעו למסיבת החנוכה שלו.

למשל, אם יקבל ראובן מסבתו 25 ₪, ולמסיבה יגיעו 10 חברים, יקבל כל אחד מהחברים 2.5 ₪.

ראובן כתב את ההוראות הבאות כדי לחשב את הפלט הנדרש:

```
   int money, friends;
   double gift;
1. System.out.print("Enter sum of money: ");
2. money = in.nextInt();
3. System.out.print("Enter number of friends: ");
4. friends = in.nextInt();
5. gift = money / friends;
6. System.out.println("Every friend will get: " + gift + " shekels");
```

נעקוב אחר מהלך ביצוע משפטי התוכנית עבור הקלט 25 10:

משפט לביצוע	money	friends	gift	פלט
1 System.out.print(…)	?	?	?	Enter sum of money:
2 money = in.nextInt()	25	?	?	
3 System.out.print(…)	25	?	?	Enter number of friends:
4 friends = in.nextInt()	25	10	?	
5 gift = money / friends	25	10	2	
6 System.out.println(…)	25	10	2	Every friend will get: 2.0 shekels

מדוע הפלט שגוי? ניזכר בהגדרת פעולת החילוק (/) בשפת Java: כאשר מחלקים שלם בשלם, פעולת החילוק מתפרשת **תמיד** כחלוקה בשלמים.

נתבונן בשורה השלישית: בביטוי `money / friends` מתבצעת חלוקה של שני ערכים מטיפוס שלם. אמנם תוצאת החלוקה מושמת במשתנה מטיפוס ממשי, אך דבר זה נעשה רק לאחר ביצוע פעולת החלוקה, וזו הייתה פעולת חלוקה בשלמים ולא פעולת חלוקה בממשיים, כפי שנדרש. עבור הקלט 10 25, מחושבת מנת החלוקה של 25 ב-10, והתוצאה המתקבלת היא שלמה (2). תוצאה זו מוצבת במשתנה `gift` המכיל כעת את הערך הממשי 2.0 (הצורה הממשית של המספר 2). אם כך, התשובה שקיבלנו היא שגויה, מכיוון שמחֲלֵקַת 25 ב-10 ציפינו במקרה זה לקבל 2.5 ולא 2.0.

כדי לפתור את הבעיה יש להודיע כי **אף על פי** שהמשתנה `money` הוא משתנה מטיפוס שלם השומר ערך שלם, יש להתייחס זמנית אל ערכו כאל ערך ממשי. בכך החלוקה הופכת להיות חלוקה של ערך ממשי בערך שלם – לכן היא חלוקה בממשיים, כפי שנדרש. דבר זה מתבצע בשפת Java באמצעות פעולת **המרה** (casting).

לפני הביטוי שאת ערכו אנו מעוניינים לפרש זמנית כממשי ולא כשלם (במקרה זה, הביטוי `money`), נוסיף את הוראת ההמרה שמשמעותה: המר, רק לצורך החישוב הנוכחי, את ערכו של ביטוי זה לערך ממשי. ההוראה מיושמת בכתיבת שם הטיפוס שנרצה להמיר אליו, עטוף בסוגריים, לפני הביטוי המומר:

```
gift = (double)money / friends;
```

שימו ♥: המשתנים המעורבים בביטוי המומר (במקרה זה, המשתנה `money`) **אינם** הופכים מרגע זה למשתנים מטיפוס ממשי. הם נשארים משתנים מטיפוס שלם בדיוק כפי שהיו עד כה. ההמרה גורמת לכך שרק **בעת חישוב הביטוי**, ערכם נראה כממשי ולא כשלם.

בהתאם לכך, קטע הקוד הנכון הוא:

```
int money, friends;
double gift;
System.out.print("Enter sum of money: ");
money = in.nextInt();
System.out.print("Enter number of friends ");
friends = in.nextInt();
gift = (double) money / friends;
System.out.println("Every friend will get: " + gift + "shekels");
```

שאלה 4.15

לפניכם סדרת הוראות. השלימו בטבלה את ערכי המשתנים לאחר כל הוראה.

	num	fnum
`int num, x = 22, y = 5;`		
`double fnum;`		
`num = x / y;`		
`fnum = x / y;`		
`fnum = (double)x / y;`		
`num = x % y;`		

שאלה 4.16

אהרון מעוניין לדעת את ממוצע ציוניו במקצועות היסטוריה, תנ"ך וספרות.

פתחו אלגוריתם המקבל כקלט את שלושת הציונים במקצועות אלו (מספרים שלמים בין 0 ל-100), ומציג כפלט את הממוצע של שלושת הציונים. ישמו את האלגוריתם בשפת התכנות Java.

שאלה 4.17

דינה מעוניינת לדעת את הציון הכולל שלה במדעי המחשב.
ידוע כי שתי היחידות הראשונות מהוות 33% מהציון, היחידה השלישית מהווה 17% מהציון, ושתי היחידות האחרונות מהוות 50% מהציון הכללי.
פתחו אלגוריתם המקבל כקלט את שלושת ציוניה של דינה במדעי המחשב (בשתי היחידות הראשונות, ביחידה השלישית ובשתי היחידות האחרונות) כמספרים שלמים בין 0 ל-100, ומציג כפלט את ציונה הכולל. ישמו את האלגוריתם בשפת התכנות Java.

פירוק מספר דו-ספרתי לספרותיו

בסעיף זה נכיר שתי תבניות שימושיות. התבנית של פירוק מספר לספרותיו תודגם בבעיה 3. תבנית שמשלימה אותה במובן מסוים היא בניית מספר מִסְפָּרוֹת, ואליה נתייחס בשאלה 4.19.

בעיה 3

מטרת הבעיה הבאה ופתרונה: פיתוח ויישום בשלבים של אלגוריתם ושימוש בפעולות חלוקה בשלמים לצורך פירוק מספר דו-ספרתי לספרותיו.

פתחו אלגוריתם שהקלט שלו הוא מספר דו-ספרתי, חיובי או שלילי, והפלט שלו הוא סכום ספרות המספר. ישמו את האלגוריתם בתוכנית מחשב בשפת Java.

ניתוח הבעיה בעזרת דוגמאות

נבדוק עבור מספר דוגמאות מה הפלט הרצוי עבור הקלט:

קלט	פלט
71	8
-53	8
49	13

גם במקרה זה, תהליך בדיקת הפלט עבור דוגמאות שונות מסייע לנו בהבנת משמעות הבעיה ובזיהוי התהליכים הדרושים לפתרונה. למשל עבור הקלט 71 והקלט 53 קיבלנו את אותו הפלט. כעת אנו רואים כי ייתכנו קלטים שונים אשר מתאים להם אותו פלט. לקלט 53- ולקלט 53 מתאים אותו פלט, וכך אנו רואים כי אין חשיבות לסימנו של המספר שבקלט. אלו הן אבחנות חשובות לצורך כתיבת אלגוריתם נכון לפתרון הבעיה.

פירוק הבעיה לתת-משימות

את הבעיה הזו נפרק לשלוש תת-משימות:

1. קליטת מספר דו-ספרתי חיובי או שלילי
2. פירוק המספר לספרותיו וסיכום הספרות
3. הצגה של סכום הספרות כפלט

ניתן לפרק גם את התת-משימה השנייה לתת-משימות ובכך לפרט יותר את הרעיון לפתרון הבעיה:

2.1. פירוק המספר לספרותיו
2.2. סיכום הספרות

כיצד נפרק את המספר לספרותיו?

ידוע לנו כי המספר הוא דו-ספרתי, ולכן פירוקו לספרותיו פירושו חישוב ספרת העשרות וספרת האחדות. נפרק אם כן את תת-משימה 2.1, פירוק המספר לספרותיו, כך:

2.1.1. חישוב ספרת העשרות של המספר
2.1.2. חישוב ספרת האחדות של המספר

❓ כיצד נבצע פעולות אלו?

נוכל להיעזר בפעולות חלוקה בשלמים, של המספר הדו-ספרתי הנתון במספר 10:

שארית החלוקה של מספר שלם **כלשהו** ב-10 נותנת תמיד את **ספרת האחדות** של המספר. מנת החלוקה של מספר שלם **דו-ספרתי** חיובי ב-10 נותנת תמיד את **ספרת העשרות** של המספר.

למשל 24%10=4 ו-24/10=2.

שימו ♥: חישוב מנת החלוקה של מספר דו-ספרתי ב-10 נותנת את ספרת העשרות של המספר, אם הוא חיובי, אך לא אם הוא שלילי! למשל 2-=20/10-, ולא 2!

לכן יש לטפל באופן שונה במספר שלילי.

בדוגמאות ראינו כי הפלט עבור מספר חיובי ועבור המספר הנגדי שלו צריך להיות זהה. לכן, כל שעלינו לעשות עבור מספר שלילי, הוא "להיפטר" מהסימן -, כלומר להפוך את המספר לחיובי, עוד לפני פירוקו לספרותיו.

❓ כיצד נהפוך מספר שלילי לחיובי?

נוכל לעשות זאת על ידי חישוב ערכו המוחלט של המספר לפני פירוקו לספרותיו.

אם כן, בסך הכול, תת-משימה 2.1, פירוק המספר לספרותיו תורכב משלוש תת-משימות:

2.1.1. חישוב ערכו המוחלט של המספר
2.1.2. חישוב ספרת העשרות של המספר
2.1.3. חישוב ספרת האחדות של המספר

כעת, לאחר פירוק הבעיה לתת-משימות ברור לנו הרעיון לפתרון הבעיה וקל לכתוב את האלגוריתם עצמו.

בחירת משתנים

שני המשתנים הראשונים שנבחר דרושים לנו עבור פעולות הקלט והפלט:

num – שלם, ישמור את המספר הניתן כקלט
sum – שלם, ישמור את סכום ספרות המספר

המשתנים לשמירת החישובים השונים:

absNum – שלם, ישמור את ערכו המוחלט של המספר
tens – שלם, ישמור את ספרת העשרות של המספר
units – שלם, ישמור את ספרת האחדות של המספר

האלגוריתם

האלגוריתם, לפי החלוקה לתת-משימות ובשימוש במשתנים שבחרנו, יהיה:

1. קלוט מספר שלם דו-ספרתי ב-num
2. חשב את ערכו המוחלט של המספר והשם ב-absNum
3. חשב את מנת החלוקה של absNum ב-10 והשם ב-tens
4. חשב את שארית החלוקה של absNum ב-10 והשם ב-units
5. חשב את הסכום של tens ושל units והשם ב-sum
6. הדפס כפלט את ערכו של sum

יישום האלגוריתם

כזכור, לצורך חישוב ערכו המוחלט של מספר, נוכל להשתמש בפעולה לחישוב הערך המוחלט ששייכת למחלקה המתמטית, ומתוארת בטבלת הפעולות. המשפט המתאים ליישום הוראה 2 באלגוריתם הוא:

```
absNum = Math.abs(num);
```

התוכנית המלאה

```
/*
קלט: מספר דו-ספרתי שלם חיובי או שלילי
פלט: סכום ספרות המספר
*/
import java.util.Scanner;
public class DigitSum
{
    public static void main (String[] args)
    {
        Scanner in = new Scanner(System.in);
        // הגדרת משתנים
        int num;          // המספר הניתן כקלט
        int absNum;       // ערכו המוחלט של המספר
        int tens;         // ספרת העשרות
        int units;        // ספרת האחדות
        int sum;          // סכום הספרות
        // קלט
  1.    System.out.print("Enter a two digit number: ");
  2.    num = in.nextInt();
        // קבלת ערכו המוחלט של המספר
  3.    absNum = Math.abs(num);
        // פירוק המספר לספרותיו
  4.    tens = absNum / 10;
  5.    units = absNum % 10;
        // סכום ספרות המספר
  6.    sum = tens + units;
        // פלט
  7.    System.out.println("The sum of the digits is: " + sum);
    }  // main
}  // class DigitSum
```

מעקב

נעקוב אחר מהלך ביצוע משפטי התוכנית עבור הקלט 17- :

משפט לביצוע	num	absNum	tens	units	sum	פלט
1	?	?	?	?	?	Enter a two digit number:
2	-17	?	?	?	?	
3	-17	17	?	?	?	
4	-17	17	1	?	?	
5	-17	17	1	7	?	
6	-17	17	1	7	8	
7	-17	17	1	7	8	The sum of the digits is: 8

על פי המעקב הזה התוכנית ביצעה את מטרתה.

אך יש עוד לבדוק את התוכנית עבור **דוגמאות קלט מגוונות** (כלומר, בעלות מאפיינים שונים אשר מבטאים את מגוון הקלטים האפשריים) כדי להשתכנע שאכן היא מבצעת את מטרתה עבור כל קלט אפשרי.

בטבלה זו בדקנו דוגמה לקלט שלילי. כדאי לבדוק אם התוכנית מבצעת את מטרתה גם עבור קלט חיובי.

שאלה 4.18

בנו טבלת מעקב אחר ביצוע משפטי התוכנית DigitSum עבור הקלט 53.

סוף התרון בעיה 3

להעמקה בתבנית ***פירוק מספר חיובי לספרותיו*** פנו לסעיף התבניות המופיע בסוף הפרק.

שאלה 4.19

שנו את התוכנית DigitSum (התוכנית מפתרון בעיה 3) כך שהקלט יהיה המספר שהתקבל והפלט יהיה המספר בסדר ספרות הפוך. למשל עבור הקלט 43 הפלט הנדרש הוא 34.

הדרכה: בידיכם שני משתנים המבטאים את ספרת האחדות (units) ואת ספרת העשרות (tens) של המספר. חשבו על ביטוי חשבוני אשר משתמש בשני ערכים אלה ונותן את המספר הנדרש.

להעמקה בתבנית ***בניית מספר*** פנו לסעיף התבניות המופיע בסוף הפרק.

שאלה 4.20

שנו את התוכנית DigitSum (התוכנית מפתרון בעיה 3) כך שהפלט יהיה **המרחק** בין ספרות המספר. למשל עבור הקלט 41, הפלט הנדרש הוא 3.

הדרכה : היעזרו בפעולה לחישוב ערך מוחלט.

4.3 הטיפוס התווי

עד עתה עסקנו רק במספרים מטיפוס שלם או מטיפוס ממשי. בפרק זה נלמד כי עיבודים ניתן לבצע לא רק על מספרים, אלא גם על תווים. תו הוא כל סימן אשר ניתן להציג על מסך המחשב.

כגון: אותיות ('a', 'b', 'A', ...), סימנים מתמטיים ('<', '+', '*', '/', '%', ...), ספרות ('0', '1', '2', ...), סימני תחביר ('!', ',', ':', ...), רווח (' '), וכן הלאה.

לצורך הטיפול הפנימי של המחשב בתווים מותאם לכל תו מספר שלם, על פי קוד סטנדרטי, השומר על העקרון: לתווים עוקבים מותאם ערך מספרי עוקב. למשל אם לתו 'b' מותאם מספר כלשהו, לתו 'c' מותאם המספר העוקב לו. למעשה, כל התווים מסודרים בתת-קבוצות המכילות תווים עוקבים: האותיות האנגליות הגדולות (A..Z) מסודרות בתת קבוצה אחת והאותיות הקטנות (a..z) מסודרות בתת קבוצה אחרת, אותיות עבריות בתת-קבוצה נפרדת, וכך גם הספרות.

בהמשך לימודינו נלמד כיצד ניתן לעבד מילים ואף משפטים שלמים, אך בסעיף זה נתמקד כאמור בתווים בודדים.

ערך מטיפוס תווי מצוין בשפת Java בין גרשיים בודדים. למשל 'Z'.

בעיה 4

מטרת הבעיה הבאה ופתרונה: הצגת הטיפוס התווי

פתחו אלגוריתם אשר הקלט שלו הוא אות אנגלית גדולה והפלט שלו הוא האות האנגלית הקטנה המתאימה. ישמו את האלגוריתם בתוכנית מחשב בשפת Java.

ניתוח הבעיה בעזרת דוגמאות

הדוגמאות הבאות מבהירות את הנדרש בבעיה 4:

קלט	פלט
'A'	'a'
'D'	'd'
'V'	'v'

פירוק הבעיה לתת-משימות

1. קליטת אות אנגלית גדולה
2. חישוב האות האנגלית הקטנה המתאימה
3. הצגת האות האנגלית הקטנה המתאימה כפלט

? כיצד נחשב את האות האנגלית הקטנה המתאימה?

אנו יודעים כי לכל תו מתאים מספר, ולתווים עוקבים מתאימים מספרים עוקבים.
הנה דוגמה אפשרית למספור כזה:

'A' – 65	'a' – 97
'B' – 66	'b' – 98
'C' – 67	'c' – 99
'D' – 68	'd' – 100
...	...

מכיוון שמספור האותיות הגדולות מתחילות ממספר מסוים, וממשיך משם בסדר עולה רציף, וכך גם לגבי האותיות הקטנות, הרי **ההפרש** בין המספר המייצג אות גדולה למספר המייצג אות קטנה

המתאימה לה הוא קבוע. כלומר, אם למשל, ההפרש בין 'a' ל-'A' הוא 32 (כפי שמתקיים בדוגמה שלעיל כי 32=65-97), זהו גם ההפרש בין 'b' ל-'B' (32=66-98), בין 'c' ל-'C', וכן הלאה.

אם כך, כל שיש לעשות הוא להוסיף למספר המייצג את האות האנגלית הגדולה שקלטנו את ההפרש הקבוע שבין אות קטנה והאות הגדולה המתאימה לה. את ההפרש הקבוע הזה נוכל לחשב עבור זוג אותיות כלשהו למשל עבור האותיות 'a' ו-'A'.

הנה הפירוק של תת-משימה 2:

2.1. חישוב ההפרש הקבוע בין אות קטנה לאות הגדולה המתאימה לה, על ידי הפחתת המספר המייצג את 'A' מהמספר המייצג את 'a'.
2.2. הוספת ההפרש שחושב למספר המייצג את האות הגדולה שנקלטה

בחירת משתנים

נשתמש במשתנים הבאים:

capitalLetter – מטיפוס תווי, ישמור את התו הניתן כקלט, כלומר את האות הגדולה
smallLetter – מטיפוס תווי, ישמור את תו הפלט, כלומר את האות הקטנה המתאימה

האלגוריתם

הנה האלגוריתם המתאים לפירוק לתת-משימות ולמשתנים שנבחרו:

1. *קלוט אות גדולה ב-*capitalLetter
2. *חשב את ההפרש בין המספר המייצג את 'a' למספר המייצג את 'A' הוסף את ההפרש שחושב ל-*capitalLetter *והשם את התוצאה ב-*smallLetter
3. *הצג כפלט את תוכן* smallLetter

יישום האלגוריתם

האלגוריתם משתמש במשתנים מטיפוס תווי, ולכן צריך כמובן להצהיר עליהם.

משתנה מטיפוס תווי מוצהר באמצעות המילה **char** (קיצור של המילה character, ומשמעותה תו באנגלית).

למשל:

```
char capitalLetter;
```

יישום הוראה 1 באלגוריתם צריך להיות משפט קלט, הקולט תו. פעולת הקלט של ערכים מטיפוס תווים היא הפעולה `in.next().charAt(0)`, והשימוש בה דומה לשימוש בפעולות קלט אחרות שראינו. לכן, היישום של הוראה 1 יהיה:

```
System.out.print("Enter a capital letter: ");
capitalLetter = in.next().charAt(0);
```

כיצד נבצע את החישובים הנדרשים בהוראה 2?

בשפת Java תו מזוהה עם הערך המספרי המתאים לו. לכן ניתן לשלב תווים בביטויים חשבוניים, והפעולות שכלולות בביטוי יפעלו על הערכים המספריים המתאימים לתווים. לכן, הביטוי 'a'-'A' נותן לנו את ההפרש שאנו זקוקים לו. אם כך, הביטוי הבא מבטא את החישובים בהוראה 2:

```
capitalLetter +  'a' - 'A';
```

אבל מאחר שבשפת Java לכל תו יש ערך מספרי, הרי כאשר אנו מבצעים פעולות חשבוניות על ערכים מטיפוס תווי, התוצאה היא מטיפוס שלם. ליישום הוראה 2, נרצה להשים את ערך הביטוי בתוך משתנה מטיפוס תווי, smallLetter, ולכן עלינו להשתמש בהמרה (casting), הפעם כדי להמיר את ערך הביטוי מטיפוס שלם לטיפוס תווי.

נשתמש בפעולת ההמרה באותו האופן שהשתמשנו בה בסעיף הקודם. כאן נמיר ערך ביטוי מטיפוס שלם להיות ערך מטיפוס תווי. שימו לב שההמרה צריכה לחול על כל הביטוי, ולכן נעטוף אותו בסוגריים.

אם כך, היישום ב-Java של הוראה 2 הוא:

```
smallLetter = (char)(capitalLetter +  'a' - 'A');
```

שימו ♥: הוראת ההמרה לטיפוס תווי מבוטאת על ידי זוג הסוגריים השמאלי. זוג הסוגריים הימני מגדיר את התחום שמופעלת עליו ההמרה.

שימו ♥: בפתרון זה אנו מתייחסים הן למשתנים מטיפוס תווי (smallLetter, capitalLetter) והן לערכים תוויים, כלומר ערכים קבועים מטיפוס תווי ('a', 'A'). הערכים התוויים מוקפים בגרשיים. ההתייחסות למשתנים היא דרך שמותיהם, בדיוק כפי שנעשה עבור משתנים מטיפוס מספרי.

פלט של תווים נעשה ב-Java בעזרת הוראות הפלט המוכרות לנו.

ניישם את האלגוריתם תוך שימוש בהוראות קלט ופלט עבור תווים:

```
1. System.out.print("Enter a capital letter: ");
2. capitalLetter = in.next().charAt(0);
3. smallLetter = (char)(capitalLLetter + 'a' - 'A');
4. System.out.println("The corresponding small letter is: " +
                                                    smallLetter);
```

שימו ♥: הוראות אלו אינן מסתמכות על ידיעת ערך ההפרש עצמו, או על ידיעה מפורשת של הערך המספרי המותאם לכל תו. בכך אנו מסייעים לתוכנית להיות עמידה יותר: גם אם גרסאות מאוחרות יותר של Java ישתמשו בקידוד אחר לתווים, הרי כל עוד יישמר העיקרון שתווים עוקבים מתאימים למספרים עוקבים, התוכנית תישאר נכונה.

התוכנית המלאה

```
/* התוכנית קולטת אות אנגלית גדולה
ומציגה כפלט את האות האנגלית הקטנה המתאימה לה */
import java.util.Scanner;
public class CapitalLetterToSmall
{
    public static void main (String[] args)
    {
        // הגדרת משתנים
        char capitalLetter;    //  תו הקלט - אות אנגלית גדולה
        char smallLetter;      //   תו הפלט - אות אנגלית קטנה
        Scanner in = new Scanner(System.in);
        // קלט
       System.out.print("Enter a capital letter: ");
        capitalLetter = in.next().charAt(0);
        // חישוב האות הקטנה
        smallLetter = (char)(capitalLetter + 'a' - 'A');
        // פלט
```

```
        System.out.println("The corresponding small letter is: " +
                                                        smallLetter);
    } // main
} // class CapitalLetterToSmall
```

סוף התרון בעיה 4

שאלה 4.21

שנו את התוכנית `CapitalLetterToSmall` כך שתקבל כקלט אות אנגלית **קטנה**, ותציג כפלט את האות האנגלית **הגדולה** המתאימה לה.

רמז: שימו לב לסדר פעולות חשבון! בדקו עצמכם בזהירות!

שאלה 4.22

פתחו אלגוריתם הקולט תו ומציג כפלט את התו הבא אחריו בסדר הקידוד המספרי. ישמו את האלגוריתם כתוכנית בשפת Java.

שימו ♥: מה התו אחרי התו 'a'? מה התו אחרי התו 'S'? מה התו אחרי התו 'z'? בחנו דוגמאות קלט מגוונות ככל שניתן.

המרה מתו המייצג ספרה לערך מספרי מתאים

כפי שנאמר בתחילת הפרק, ערכי הטיפוס התווי כוללים כל סימן אשר ניתן להציג על מסך המחשב. בין סימנים אלה נכללות גם הספרות. למשל ערך מטיפוס תו יכול להיות '4' או '7'. לעתים, בהינתן תו המייצג ספרה, נרצה לדעת את ערכה המספרי של הספרה (כלומר להתאים לתו '4' את הערך השלם 4).

הנה דוגמה אפשרית למספור הפנימי עבור התווים המייצגים ספרות:

'0' – 48
'1' – 49
'2' – 50
'3' – 51
...

התווים המייצגים ספרות ('0', '1', '2' וכו') הם תווים עוקבים, ולכן גם המספור שלהם עוקב.

? כיצד ניתן לנצל את העובדה שמספור התווים המייצגים ספרות הוא עוקב כדי לחשב, עבור תו המייצג ספרה, את ערכה המספרי של הספרה?

נשים לב כי התו '1' מרוחק תו אחד מהתו '0'. בדומה, התו '2' מרוחק שני תווים מהתו '0', התו '3' מרוחק שלושה תווים מהתו '0', וכך הלאה. לכן נוכל להשתמש בפעולת חיסור כדי לחשב את הערך הדרוש. למשל אם נפחית מהמספר המתאים לתו '3' את המספר המתאים לתו '0', נקבל בדיוק 3 (3 = 51-48 = '0'-'3').

גם הפעם אין אנו משתמשים באופן ישיר בערכי הקידוד עצמם, אלא רק מסתמכים על כך שהקידוד נותן ערכים עוקבים לספרות עוקבות.

לכן, אם במשתנה `charDigit` (מטיפוס תווי) שמור תו המייצג ספרה, אז לאחר החישוב שלהלן יכיל המשתנה `digit`, מטיפוס שלם, את ערכה המספרי של הספרה המיוצגת על ידי התו שב-`charDigit`.

```
digit = charDigit - '0';
```

שאלה 4.23

פתחו אלגוריתם המקבל כקלט תו המייצג ספרה.
(יש עשרה תווים אפשריים לקלט זה: '0', '1', '2' ... '9'). פלט האלגוריתם יהיה מורכב משלושה מספרים: הספרה המיוצגת על ידי תו הקלט (שנקרא לה לשם קיצור ספרת הקלט), ספרת הקלט לאחר הכפלתה ב-10, וספרת הקלט לאחר הכפלתה ב-15. ישמו את האלגוריתם כתוכנית בשפת Java.

הדרכה: חשבו מהו המספר המתקבל אחרי הכפלת ספרת הקלט ב-10.

שאלה 4.24

פתחו אלגוריתם המקבל כקלט שני תווים, כל אחד מהם מייצג ספרה. פלט האלגוריתם יהיה תוצאת החיבור בין שני המספרים המתאימים. למשל, אם הקלט הוא '5' '9' הפלט יהיה 14.
ישמו את האלגוריתם כתוכנית בשפת Java.

בשפת Java קיימות מחלקות מוכנות המיועדות לשימוש המתכנת. על אחת מהן למדנו בתחילת פרק זה, המחלקה Math ובה פעולות מתמטיות. בסעיף הבא נלמד על מחלקה נוספת בשם Random שיכולה לשמש אותנו ליצירת מספרים אקראיים.

4.4 בחירה אקראית

הגרלת מספרים אקראיים היא פעולה שימושית מאוד במדעי המחשב. למשל, בתחום בדיקות התוכנה, השימוש במספרים אקראיים מאפשר בדיקת תוכנה על אוסף קלטים גדול ואקראי שנוצר אוטומטית. גם בתחום ההצפנה מספרים אקראיים הם שימושיים מאוד. בסעיף זה נכיר מחלקה המוגדרת בשפת Java ומאפשרת עבודה נוחה עם מספרים אקראיים.

כיצד מגרילים מספר שלם ?

כדי ליישם הוראת הגרלה בשפת Java נזדקק ל**מחלקת מספרים אקראיים:** `Random`.
בעזרת עצם מהמחלקה `Random` ניתן לבצע הגרלות של ערכים בכל תחום שנבחר. כדי שנוכל להשתמש בעצם מהמחלקה עלינו להצהיר עליו וליצור אותו באופן הבא:

```
Random rnd = new Random();
```

הוראה זו מצהירה על עצם מהמחלקה `Random`, בשם `rnd`, יוצרת אותו ומקצה לו מקום בזיכרון. כעת, כשיש בידינו עצם מהמחלקה `Random` אנו יכולים להגריל בעזרתו מספר שלם על ידי הפעלת הפעולה `nextInt` של העצם. השימוש בפעולה זו דומה לשימוש בפעולות של המחלקה Math שנלמדו בפרק זה. פעולה זו מקבלת בסוגריים פרמטר יחיד, שהוא ערך שלם המתאר את טווח המספרים להגרלה. אם נכתוב `nextInt(n)` יוגרל מספר שלם בתחום שבין 0 ל-(n-1). מאחר שזו פעולה של העצם, יש להשתמש בסימון הנקודה כדי להפעילה. למשל בעקבות ביצוע ההוראות:

```
int num;
num = rnd.nextInt(6);
```

יושם במשתנה `num` מספר אקראי כלשהו בין 0 ל-5.

שימו ♥: כאשר משתמשים בפעולות של המחלקה Math מפעילים אותן ישירות מהמחלקה Math (למשל: `Math.sqrt(x)`), ואילו כאשר משתמשים בפעולה `nextInt` של Random, צריך ליצור תחילה עצם מסוג Random ועליו להפעיל את הפעולה `nextInt`. הבדל זה נובע מכך שהפעולות במחלקה Math מוגדרות כפעולות סטטיות השייכות למחלקה ולא לעצם. פרק 11 מרחיב בנושא זה.

בעיה 5

מטרת הבעיה הבאה ופתרונה: הצגת מחלקת מספרים אקראיים ואופן השימוש בה.

פתחו אלגוריתם אשר ידמה הטלה של 2 קוביות, כלומר יגריל שני ערכים בתחום 1-6 ויציג אותם כפלט. ישמו את האלגוריתם בשפת התכנות Java.

בבעיה זו אנו מתבקשים לדמות הטלת שתי קוביות, כלומר להגריל שני ערכים בתחום 1-6.

פירוק הבעיה לתת-משימות

נפרק את הבעיה באופן הבא:

1. הגרלת 2 מספרים בין 1 ל-6.
2. הצגה של המספרים שהוגרלו כפלט.

בחירת משתנים

die1 – מספר שלם שערכו בין 1-6 ובו המספר הראשון שיוגרל.
die2 – מספר שלם שערכו בין 1-6 ובו המספר השני שיוגרל.

האלגוריתם

1. *הגרל מספר בין 1 ל-6 והשם ב-die1*
2. *הגרל מספר בין 1 ל-6 והשם ב-die2*
3. *הדפס כפלט:" סכום שתי הקוביות הוא " die1, die2*

יישום האלגוריתם

אם נעביר לפעולה `nextInt` את הערך 6, נקבל מספר אקראי בין 0 ל-5. אם נעביר לפעולה `nextInt` את הערך 7, נקבל מספר אקראי בין 0 ל-6. אם כך:

? כיצד נדמה הטלת קובייה? כלומר כיצד נגריל מספר בין 1 ל-6?
נעביר לפעולה `nextInt` את הערך 6, על מנת לקבל מספר אקראי בין 0 ל-5. כדי "להמיר" את הערך שהתקבל לערך אפשרי עבור הטלת קובייה, נוסיף לו את הערך 1. כך יהיה בידינו מספר אקראי בין 1 ל-6.

בתוכנית זו עלינו לבצע שתי הגרלות, ואכן לאחר שהקצינו לעצם מהמחלקה `Random` מקום בזיכרון, אנו יכולים להשתמש בו שוב ושוב להגרלות נוספות (אפילו מתוך תחומים שונים).

עלינו להצהיר בתחילת התוכנית על כוונתנו להשתמש במחלקה `Random`, כפי שאנו עושים כאשר אנו משתמשים במחלקה `Scanner`. המחלקה `Random` נמצאת במארז `java.util` ולכן נציין שאנו "מייבאים" את המחלקה באמצעות ההצהרה הבאה:

```
import java.util.Random;
```

התוכנית המלאה

```
/*
התוכנית מדמה שתי הטלות קובייה ומדפיסה את ערכן
*/
import java.util.Random;
public class TwoDice
```

הודעה כי נשתמש במחלקה Random

```
{
    public static void main (String[] args)
    {
            // הגדרת משתנים
            int die1;          // תוצאת הטלת קובייה אחת
            int die2;          // תוצאת הטלת קובייה שנייה
            Random rnd = new Random(); // יצירת עצם מסוג Random
            die1 = rnd.nextInt(6) + 1;    // הטלת קובייה ראשונה
            die2 = rnd.nextInt(6) + 1;    // הטלת קובייה שנייה
            // הדפסות
            System.out.println("The first one : " + die1);
            System.out.println("The second one : " + die2);
    }// main
}// class TwoDice
```

סוף פתרון בעיה 5

נסכם את המושגים החדשים שהוצגו בפתרון בעיה 5:

בשפת Java מוגדרת **מחלקה למספרים אקראיים** הנקראת Random. באמצעות עצם מהמחלקה ניתן להגריל מספרים אקראיים בתחום מבוקש. כדי להשתמש במחלקה בתוך תוכנית Java צריך לייבא את המחלקה באמצעות ההוראה הבאה:

```
import java.util.Random;
```

לפני ביצוע הגרלות יש להצהיר על עצם מהמחלקה `Random` וליצור אותו, למשל כך:

```
Random rnd = new Random();
```

הפעולה `nextInt` של עצם מהמחלקה `Random` מקבלת כפרמטר מספר שלם חיובי n ומחזירה ערך אקראי שלם בתחום 0 עד n-1. מאחר שזו פעולה של עצם, יש להפעילה באמצעות סימון הנקודה, למשל כך:

```
rnd.nextInt(10)
```

ניתן לשלב את הפעולה בתוך ביטויים חשבוניים, כך שערכם יהיה ערך אקראי בתחום רצוי.

```
rnd.nextInt(6) + 1
```

באמצעות עצם מהמחלקה `Random`, ניתן לבצע הגרלות חוזרות ונשנות, גם מתחומים שונים.

נחזור ונדגים את המושגים החדשים שהוצגו בפתרון בעיה 5:

? כיצד נדע מה טווח הערכים האפשריים עבור ביטוי הכולל הגרלה של מספר אקראי?

נתבונן בביטוי ונבדוק מה טווח הערכים האפשריים עבורו, לדוגמה:

```
rnd.nextInt(11) + 100
```

הפעולה `rnd.nextInt(11)` מחזירה מספר אקראי שלם בתחום שבין 0 ל-10. לאחר מכן נוסף לערך המוגרל המספר 100. לכן אם יוגרל המספר 0 ערך הביטוי כולו הוא 100, ואם יוגרל המספר 10 ערך הביטוי כולו הוא 110. לפיכך טווח הערכים האפשריים עבור הביטוי הוא בין 100 ל-110.

? כיצד נגריל מספר בטווח הערכים בין 400 ל-500?

תחילה נבדוק כמה מספרים יש בין 400 ל-500. בתחום זה יש 101 מספרים (מדוע?). אם כך עלינו להגריל אחד מתוך 101 מספרים, וזאת נעשה בעזרת הפעולה `rnd.nextInt(101)`. באופן זה יוגרל מספר בין 0 ל-100. כעת כדי לקבל מספר בתחום שבין 400 ל-500, עלינו להוסיף

לערך המוגרל 400. אכן אם יוגרל 0 נקבל לאחר ההוספה את הערך 400, ואם יוגרל 100 נקבל לאחר ההוספה את הערך 500. לכן, הביטוי המתאים הוא:

```
rnd.nextInt(101) + 400
```

שאלה 4.25

בהינתן העצם `rnd` מהמחלקה `Random`,

א. מהו טווח הערכים האפשריים למשתנה השלם `num` בעקבות ביצוע ההוראה הבאה?

```
num = rnd.nextInt(6) + 1;
```

ב. מהו טווח הערכים האפשריים שיוצגו על המסך בעקבות ביצוע ההוראה הבאה?

```
System.out.println(rnd.nextInt(10));
```

ג. מהו טווח הערכים האפשריים שיוצגו על המסך בעקבות ביצוע ההוראה הבאה?

```
System.out.println(rnd.nextInt(11) + 100);
```

שאלה 4.26

בהינתן העצם `rndNum` מהמחלקה `Random`,

מהו טווח הערכים האפשריים עבור כל אחד מהביטויים הבאים?

א. `rndNum.nextInt(100)` _____

ב. `rndNum.nextInt(100) + 1` _____

ג. `rndNum.nextInt(100 + 1)` _____

ד. `rndNum.nextInt(101) + 100` _____

ה. `rndNum.nextInt(100) + 200` _____

ו. `rndNum.nextInt(50) + 7` _____

שאלה 4.27

בכל אחד מהסעיפים הבאים תארו ביטוי, המתייחס לעצם `rndNum` מהמחלקה `Random`, כך שערך הביטוי הוא בתחום המבוקש.

א. מספר אקראי שלם בין 0 ל-9 _____

ב. מספר אקראי שלם בין 10 ל-100 _____

ג. מספר אקראי שלם בין 100 ל-500 _____

ד. מספר אקראי שלם **וזוגי** בין 0 ל-100 _____

ה. מספר אקראי שלם **ותלת-ספרתי** _____

שאלה 4.28

א. נסו לחשוב על נוסחה כללית כיצד נגריל מספר בטווח הערכים בין שני ערכים שלמים כלשהם x ו-y, כאשר ידוע ש-x הוא הערך הקטן מבין השניים.

ב. כתבו קטע תוכנית בשפת Java שיקלוט שני מספרים שלמים x ו-y (כאשר x<y), ויגריל מספר בתחום שבין x ל-y. השתמשו בנוסחה שכתבתם בסעיף א'.

4.5 שימוש במחלקות גרפיות

בסעיף זה נציג שתי מחלקות גרפיות מתוך המארז unit4 שאפשר להוריד מאתר הספר. מחלקה אחת מגדירה צב גרפי, המסוגל לנוע על פני המסך ולצייר את מסלול תנועתו, והמחלקה השנייה מגדירה דלי גרפי שניתן למלאו במים. כל שימוש באחת ממחלקות אלה דורש כתיבת משפט

import המייבא את המחלקה הרצויה בתחילת התוכנית. למשל כדי לייבא את מחלקת הצב נרשום: import unit4.turtleLib.Turtle.

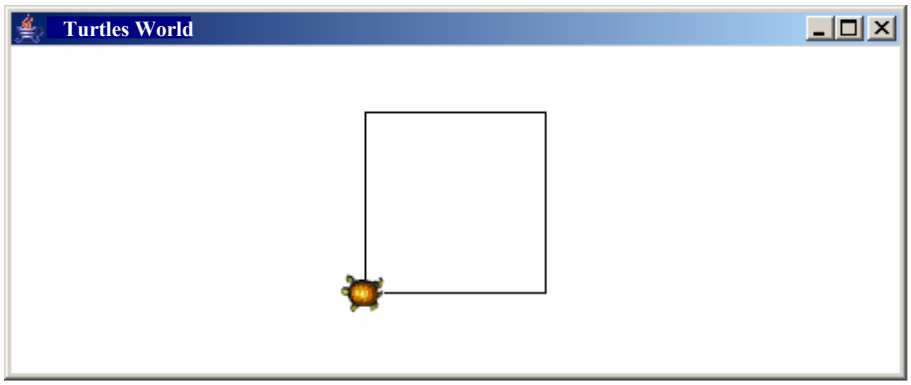
מחלקת הצב: מחלקה זו מגדירה צב גרפי, שבאמצעותו אפשר לצייר קווים. כאשר יוצרים צב (בעזרת הפעולה **new**) הוא ימוקם במרכז המסך כשפניו פונות למעלה. כעת ניתן לתת לצב "פקודות" על מנת שינוע על המשטח הגרפי. הצב נע בקו ישר בכיוון שפניו פונות אליו. ניתן לסובב את הצב כדי שינוע בכיוון הרצוי. בנוסף לכך לצב יש זנב, שניתן להרים ולהוריד. כאשר הצב נע בזנב מורד הוא מצייר קו במסלול שהוא נע בו. התבוננו בתוכנית הנתונה:

```
import unit4.turtleLib.Turtle;
public class TurtleDrawRectangle
{
        public static void main(String[] args)
        {
                // מייצרים צב חדש שפניו מופנות צפונה
                Turtle t1 = new Turtle();
                // מורידים את הזנב כדי שהצב יצייר קו במסלול שהוא נע בו
                t1.tailDown();
                t1.moveForward(100);  // הצב צועד 100 צעדים קדימה
                t1.turnRight(90);     // הצב פונה 90 מעלות ימינה
                t1.moveForward(100);  // הצב שוב צועד 100 צעדים קדימה
                t1.turnRight(90);     // הצב שוב פונה 90 מעלות ימינה
                t1.moveForward(100);
                t1.turnRight(90);
                t1.moveForward(100);
        } // main
} // class TurtleDrawRectangle
```

הודעה כי נשתמש במחלקה Turtle הנמצאת במארז unit4.turtleLib

מהתבוננות בתוכנית ניתן לראות כי הצב "מטייל" על המסך במסלול המשאיר על המסך ריבוע שצלעותיו הם בגודל של 100 צעדי צב. חישבו: היכן ימוקם הצב בסוף התוכנית?

כיוון שלא הזזנו את הצב לאחר ציור הצלע האחרונה של הריבוע הוא יעמוד בדיוק בסופה, כלומר בפינה השמאלית התחתונה של הריבוע ופניו פונות שמאלה, הנה כך:



בנוסף לפעולות שהכרנו בקטע התוכנית הקודם (הורדת זנב, פנייה ימינה, וצעידה קדימה) קיימות פעולות נוספות לצב הגרפי. להלן רשימת הפעולות של צב:

הפעולה	טיפוס הפרמטר	טיפוס הערך המוחזר	תיאור הפעולה	דוגמה לשימוש בפעולה
`Turtle()`	–	צב	הפעולה יוצרת צב שפניו מופנות למעלה	`t1 = new Turtle()`
`moveBackward(x)`	ממשי	–	הפעולה מזיזה את הצב x צעדים אחורה	`t1.moveBackward(100)`
`moveForward(x)`	ממשי	–	הפעולה מזיזה את הצב x צעדים קדימה	`t1.moveForward(100)`
`tailDown()`	–	–	הפעולה מורידה את זנב הצב	`t1.tailDown()`
`tailUp()`	–	–	הפעולה מרימה את זנב הצב	`t1.tailUp()`
`turnLeft(x)`	ממשי	–	הפעולה מפנה את הצב x מעלות נגד כיוון השעון	`t1.turnLeft(45)`
`turnRight(x)`	ממשי	–	הפעולה מפנה את הצב x מעלות בכיוון השעון	`t1.turnRight(45)`

שימו ♥: הפעולות "קדימה" ו"אחורה" מניעות את הצב בכיוון שהוא פונה אליו.

שאלה 4.29

כתבו תוכנית המשתמשת בצב הגרפי כדי לצייר מקבילית שאורך צלעותיה הן 100 ו-50. בסיום הציור יש להעמיד את הצב 20 צעדים משמאל למקבילית, ללא קו מחבר.

הדרכה: הרימו את הזנב של הצב לאחר ציור המקבילית, והזיזו אותו שמאלה.

שאלה 4.30

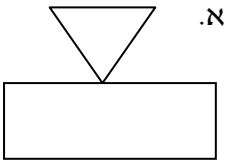
כתבו תוכנית המציירת על המסך מתומן משוכלל שאורך צלעו 70 באמצעות הצב הגרפי.

חִשבו מה צריך להיות גודל הזוויות.

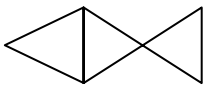
שאלה 4.31

כתבו תוכנית המציירת את הציורים הבאים באמצעות הצב הגרפי. עליכם לצייר כל ציור מבלי להרים את זנבו של הצב אפילו פעם אחת, ומבלי לחזור על אותו קו פעמיים, במשיכת קולמוס אחת:

א.



ב.



בסוף השרטוט העמידו את הצב עם הפנים למעלה במרחק מה מן הציור, כך שלא ייגע בו כלל.

מחלקת הדלי: מחלקה זו מגדירה דלי גרפי. את הדלי אפשר למלא, לרוקן ואפשר לבדוק את הקיבולת שלו. בכל פעם שנייצר דלי, נידרש לתת לו שם, וקיבולת מקסימלית. פעולה זו גם תציג את הדלי על המסך. כדי לייצר את הדלי נשתמש בפעולה בונה בצורה הבאה:

```
Bucket b = new Bucket(50, "BuckBuck")
```

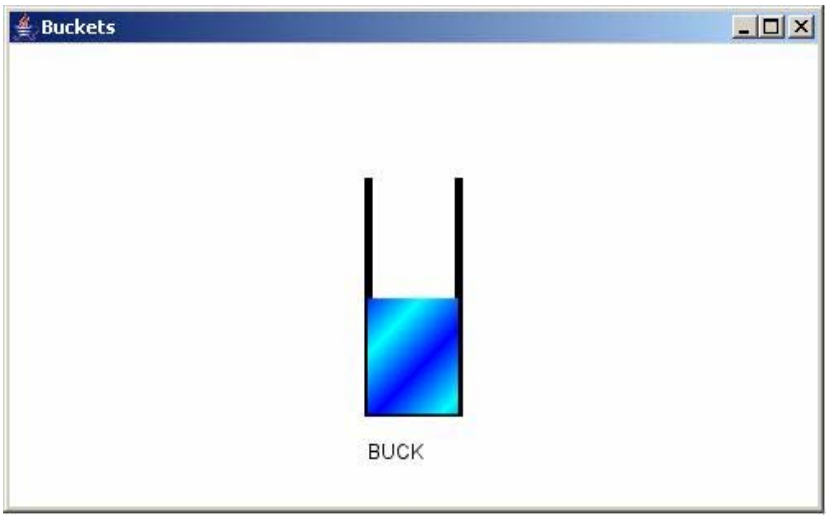
פעולה זו מייצרת דלי, שהקיבולת המקסימלית שלו היא 50 ליטר, ושמו הוא `BuckBuck`. הדלי מוצג על המסך כאשר שמו מופיע מתחתיו. על דלי אפשר לבצע פעולות שונות, והנה חלק מהן:

הפעולה	טיפוסי הפרמטרים	טיפוס הערך המוחזר	תיאור הפעולה	דוגמה לשימוש בפעולה
Bucket(x,"שם הדלי")	שלם, מחרוזת	דלי	הפעולה יוצרת דלי ריק, בקיבולת מקסימלית של x ליטרים, ושמו הוא השם הנבחר. הדלי מוצג על המסך.	b=**new** Bucket(100,"aa")
getCurrentAmount()	–	שלם	הפעולה מחזירה את כמות המים הקיימת כרגע בדלי.	x= b.getCurrentAmount()
getCapacity()	–	שלם	הפעולה מחזירה את הקיבולת המקסימלית של הדלי.	x=b.getCapacity()
pourInto(b2)	Bucket	–	הפעולה מעבירה את כמות המים המקסימלית האפשרית מהדלי עליו מופעלת הפעולה לדלי אחר שהתקבל פרמטר.	b.pourInto(b2) b2 הוא דלי היעד
empty()	–	–	הפעולה מרוקנת את הדלי.	b.empty()
fill(x)	שלם	–	הפעולה מקבלת כמות של מים וממלאת את הדלי בכמות זו. אם הכמות גדולה מקיבולת הדלי, הוא מתמלא, ויתר המים נשפכים החוצה.	b.fill(100)

התבוננו בתוכנית הבאה:

```
import unit4.bucketLib.*;
public class BucketTest
{
      public static void main(String[] args)
      {
            Bucket b = new Bucket(100,"BUCK");
            b.fill(50);
      } // main
} // class BucketTest
```

התוכנית מייצרת דלי בשם BUCK בקיבולת מקסימלית של 100 ליטר. בהמשך ממלאים את הדלי במחצית מהקיבולת שלו (50 ליטר). במהלך הרצת התוכנית אפשר לראות את הדלי הגרפי מתמלא עד למחצית. התוצאה על המסך תיראה כך:



שמו של הדלי כתוב מתחתיו, וכמות המים נראית בכחול.

שאלה 4.32

כתבו תוכנית שתייצר שני דליים בשמות כרצונכם. הקיבולת של הדלי הראשון תהיה 200 ליטר, ושל השני תהיה 100 ליטר. מלאו את הדלי הראשון ב-200 ליטר, והעבירו מים מהדלי הראשון לשני. כמה מים עברו? הציגו כערך מספרי את כמות המים בכל דלי. לאחר מכן, רוקנו את הדלי הראשון.

שאלה 4.33

כתבו תוכנית שתייצר שלושה דליים בקיבולת זהה. התוכנית תמלא את הדלי הראשון, ותעביר את המים לדלי השני ואחר כך לדלי השלישי. בסוף התוכנית כל המים יהיו בדלי השלישי. הציגו כערך מספרי את כמות המים בכל דלי.

שאלה 4.34

כתבו תוכנית המייצרת דלי בשם כרצונכם ובעל קיבולת אקראית בין 100 ל-200 ליטר. מלאו את הדלי עד החצי והציגו כערך מספרי את כמות המים בדלי.

סיכום

פיתוח ויישום אלגוריתמים

הדגמנו בפרק זה את שלבי התהליך של פיתוח ויישום אלגוריתם, באמצעות בעיות ברמות שונות של קושי ושל מורכבות. למרות הבדלי הרמות, הקפדנו על שלבי התהליך בפתרון כל הבעיות: פיתוח ראשוני של הבעיה תוך בחינת הפלט עבור דוגמאות קלט מגוונות, ניסוח רעיון לפתרון על ידי פירוק לתת-משימות, בחירת משתנים, כתיבת אלגוריתם, יישום האלגוריתם במשפטי תוכנית, בדיקת מהלך ביצוע התוכנית באמצעות טבלת מעקב וכתיבת התוכנית המלאה והרצתה.

גם בשאר פרקי הספר, בעת פתרון בעיות אלגוריתמיות, נשתדל להקפיד על פיתוח אלגוריתם בשלבים ועל יישומו על פי שלבים אלה.

ייתכן כי בפתרון בעיות בהמשך, נאחד לפעמים חלק מן השלבים. נעשה זאת רק לאחר רכישת מיומנות מספקת בפתרון מפורט בשלבים. בכל מקרה נקפיד תמיד על ביצוע שלבי הבדיקה – הן הבדיקה ה"ידנית" והן הבדיקה באמצעות ההרצה.

פעולות חלוקה בשלמים

על ערכים מטיפוס שלם מוגדרות שתי פעולות: **מנת חלוקה ושארית חלוקה** (modulus):
מנת החלוקה של מספר שלם x במספר שלם y שווה לחלק השלם של x/y, ובמילים אחרות, למספר הפעמים שהערך y נכנס בערך x.

שארית החלוקה של מספר שלם x במספר שלם y מבטאת את השארית הנותרת לאחר חלוקה בשלמים של x ב-y, ובמילים אחרות, את מה שנותר אחרי שמפחיתים מ-x כפולות שלמות של y ככל שניתן.

תוצאת שארית החלוקה של מספר שלם כלשהו ב-n, יכולה להיות כל מספר שלם בתחום בין 0 ל-(n-1).

פעולות אלו מוגדרות עבור ערכים שלמים בלבד!

ניתן להיעזר בפעולות חלוקה בשלמים כדי לבצע **פירוק מספר דו-ספרתי לספרותיו**.

הטיפוס התווי

לכל תו **מותאם ערך מספרי** השמור לו.

לתווים עוקבים יש ערכים מספריים עוקבים.

סיכום מרכיבי שפת Java שנלמדו בפרק 4

פעולות חלוקה בשלמים

פעולת **מנת החלוקה** בשלמים מסומנת בשפת Java בסימן /.

כאשר הפעולה / מופעלת על שני ערכים שלמים היא מפורשת בשפת Java כפעולת חלוקה בשלמים.

בכל מקרה אחר (שלם וממשי, ממשי ושלם, ממשי וממשי) הפעולה / מפורשת בשפת Java כפעולת חלוקה על ערכים ממשיים.

פעולת **שארית החלוקה** בשלמים מסומנת בשפת Java בסימן %.

כדי לבצע פעולת חלוקה ממשית בין שני ערכים שלמים יש לבצע **המרה** (casting) של המחלק או של המחולק לערך ממשי.

המרה של ערך שלם x לערך ממשי נעשית בשפת Java כך: (**double**) x.

לפעולת ההמרה אין השפעה על טיפוס המשתנה המעורב בביטוי המומר. היא רק מנחה להתייחס אל ערכו כאילו היה מטיפוס ממשי, באופן זמני, רק לצורך החישוב הנוכחי.

הטיפוס התווי

ערך מטיפוס תווי מצוין ב-Java בין **גרשיים** בודדים, למשל כך: 'a'.

הצהרה על משתנה מטיפוס תווי נעשית באמצעות המילה **char**.

ניתן לשלב תווים בביטויים חשבוניים. בזמן חישוב הביטוי נלקחים הערכים המספריים המותאמים לכל תו ותו.

הטיפוס של ביטוי חשבוני הכולל תווים הוא מספרי. במידת הצורך, ניתן להמיר את ערכו של ביטוי כזה לערך מטיפוס תווי באמצעות פעולת ההמרה, למשל: (**char**)('a' + 3).

קלט של ערך תווי מתבצע באופן הבא:

```
Scanner in = new Scanner(System.in);
שם משתנה = in.next().charAt(0)
```

בחירה אקראית

בשפת Java ניתן **להגריל מספר אקראי** באמצעות המחלקה Random.

כדי להשתמש במחלקה Random בתוך תוכנית Java צריך לייבא אותה באמצעות ההוראה הבאה:

```
import java.util.Random;
```

לפני ביצוע הגרלות יש להצהיר על עצם מהמחלקה וליצור אותו. פעולת היצירה **new** מקצה לו מקום בזיכרון:

```
Random שם העצם = new Random();
```

הגרלת מספר אקראי שלם בתחום שבין 0 ל-n-1 מתבצעת באמצעות הפעלת הפעולה `nextInt(n)` על עצם מהמחלקה `Random`. הפעולה מופעלת באמצעות סימון הנקודה, למשל כך: `rnd.nextInt(5)`, ומחזירה ערך אקראי שלם בתחום המבוקש.

ניתן לשלב פעולת הגרלת איבר בתוך ביטויים חשבוניים, כך שערכם יהיה ערך אקראי בתחום רצוי.

באמצעות עצם מהמחלקה `Random`, שהוקצה עבורו מקום בזיכרון, ניתן לבצע הגרלות חוזרות ונשנות, גם מתחומים שונים.

שאלות נוספות

שאלות נוספות לסעיף 4.1

1. פתחו אלגוריתם אשר הקלט שלו הוא מקדמים של משוואה ריבועית: a, b ו- c והפלט הוא שני הפתרונות האפשריים של המשוואה הריבועית. הניחו כי הקלט תקין ולמשוואה הריבועית אכן קיימים שני פתרונות. ישמו את האלגוריתם בשפת Java.
להזכירכם, הנה הנוסחה לחישוב פתרונותיה של משוואה ריבועית:

$$X_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

שאלות נוספות לסעיף 4.2

1. פתחו אלגוריתם אשר הקלט שלו הוא פרק זמן הנתון בימים והפלט שלו הוא מספר השבועות השלמים הכלולים בפרק הזמן הנתון. לדוגמה עבור הקלט 18 הפלט הוא 2. ישמו את האלגוריתם כתוכנית בשפת Java.

2. נהגי מוניות שירות יוצאים לדרכם רק כאשר כל המושבים במונית תפוסים.
פתחו אלגוריתם אשר הקלט שלו הוא מספר הנוסעים הממתינים למונית בתחנה ומספר המושבים במונית, והפלט שלו הוא מספר המוניות שניתן למלא במלואן ומספר הנוסעים שייוותרו בתחנה (הניחו כי לכל המוניות בתחנה יש מספר מושבים זהה).
למשל עבור הקלט 7 25 יהיה הפלט: `ניתן למלא 3 מוניות, 4 נוסעים יוותרו בתחנה.`
עבור הקלט 35 7 יהיה הפלט: `ניתן למלא 5 מוניות, 0 נוסעים יוותרו בתחנה.`

3. נתון קטע התוכנית הבא: (`num1` ו-`num2` הם משתנים מטיפוס שלם)

```
a = num1 / 2;
b = num1 % 2;
c = num2 / 10;
d = num2 % 10;
e = num1 / num2;
d = num1 % num2;
```

א. ציינו ערך תחילי של `num1` שעבורו ערכיהם הסופיים של `a` ושל `b` יהיו 3 ו-0 בהתאמה.

ב. ציינו ערך תחילי של num2 שעבורו ערכיהם הסופיים של c ושל d יהיו 6 ו-3 בהתאמה.
ג. תנו שתי דוגמאות לערכים תחיליים של num1 ושל num2 שעבורם ערכיהם הסופיים של e ושל f יהיו 3 ו-2, בהתאמה.

שאלות מסכמות לפרק 4

1. במכולת של חנניה מוכרים מסטיקים בקבוצות על פי גודל האריזות הקיימות בחנות, והמסטיקים הנותרים נמכרים בודדים. שווי כל מסטיק הוא 0.2 ₪.
יש לפתח אלגוריתם שהקלט שלו הוא מספר המסטיקים הנמצא במכולת וגודל האריזות הקיימות בחנות (כל האריזות באותו הגודל). הפלט של האלגוריתם הוא השווי הכולל של המסטיקים בחבילות השלמות ושווי המסטיקים שנותרים לא ארוזים.
למשל, עבור הקלט 7 100 הפלט הדרוש הוא 0.4 19.6.
בחנו את הפלט עבור דוגמאות קלט מייצגות, והציגו את חלוקת המשימה המתוארת לתת-משימות.

2. חנניה מהמכולת מעוניין באלגוריתם אשר יעזור לו להחזיר עודף במטבעות בצורה היעילה ביותר, כלומר במספר המטבעות הקטן ביותר.
בהנחה כי חנניה יכול להחזיר עודף אך ורק במטבעות של 1 ₪, של 5 ₪ ושל 10 ₪, כתבו אלגוריתם המקבל כקלט את הסכום שחנניה צריך להחזיר כעודף (מספר שלם), ומציג כפלט:
א. את מספר המטבעות שיחזיר חנניה מכל סוג .
ב. את מספר המטבעות הכולל שיחזיר חנניה.
למשל עבור הקלט 18 יתקבל הפלט:
א. 1 ₪: 3, 5 ₪: 1, 10 ₪: 1
ב. 5 מטבעות.
ישמו את האלגוריתם בשפת Java.

3. כתבו קטע תוכנית בשפת Java המגריל מספר תלת-ספרתי. התוכנית תדפיס את ספרות המספר כל אחת בשורה נפרדת. מאות עשרות ואחדות.

תבניות – פרק 4

פירוט מלא של התבניות ושל שאלות שבפתרונן יש שימוש בתבניות ניתן למצוא באתר הספר ברשת האינטרנט.

חלוקת כמות פריטים לקבוצות בגודל נתון

שם התבנית: **מנת החלוקה לקבוצות של כמות פריטים**
נקודת מוצא: שני מספרים שלמים חיוביים; quantity (כמות הפריטים) ו-num (מספר הפריטים בקבוצה)
מטרה: מספר הקבוצות המלאות מחלוקה של quantity הפריטים לקבוצות בגודל num
אלגוריתם:
השם ב-groups את מנת החלוקה של quantity ב-num

שם התבנית: **שארית החלוקה לקבוצות של כמות פריטים**
נקודת מוצא: שני מספרים שלמים חיוביים; quantity (כמות הפריטים) ו-num (מספר הפריטים בקבוצה)
מטרה: מספר הפריטים העודף בחלוקה של quantity הפריטים לקבוצות בגודל num
אלגוריתם:
*השם ב-*remainder *את שארית החלוקה של* quantity *ב-*num

פירוק מספר חיובי לספרותיו

שם התבנית: **ספרת האחדות של מספר**
נקודת מוצא: מספר דו-ספרתי חיובי num
מטרה: חישוב ספרת האחדות של num
אלגוריתם:
*השם ב-*units *את שארית החלוקה של* num *ב-10*

שם התבנית: **ספרת העשרות של מספר**
נקודת מוצא: מספר דו-ספרתי חיובי num
מטרה: חישוב ספרת העשרות של num
אלגוריתם:
*השם ב-*tens *את מנת החלוקה של* num *ב-10*

בניית מספר

שם התבנית: **בניית מספר**
נקודת מוצא: שתי ספרות left ו-right
מטרה: בניית מספר דו-ספרתי מהספרות הנתונות
אלגוריתם:
*השם ב-*num *את הערך של הביטוי החשבוני* left * 10 + right