

פרק 11 תרגיל מסכם : מפה

בפרק זה נציג אוסף כללי חדש שייקרא "מפה". דרך הצגת הטיפוס ומימושו נעבור על המושגים ועל התהליכים שלמדנו ביחידת הלימוד הזו. בתום התרגול והחזרה נגלה כי בידינו טיפוס אוסף חדש ושימושי לניהול אוספי נתונים. כך מצד אחד ישמש פרק זה תרגיל רחב היקף שבו חזרה, יישום וסיכום של יחידת הלימוד, ומצד שני הוא יחדש ויציג דרך נפוצה לניהול אוספי נתונים.

נתחיל בהצגת משימה וננתח אותה במושגים של טיפול באוסף כללי, כפי שהוצגו ביחידת הלימוד. נגדיר את הפעולות הדרושות על אוסף הנתונים, ועל פיהן נגדיר את ממשק טיפוס הנתונים החדש. לאחר מכן נבחן ייצוגים שונים עבור טיפוס האוסף. ייצוגים אלה ישתמשו במבני נתונים ובטיפוסי נתונים מופשטים מתוך ארגז הכלים שבנינו לאורך יחידת הלימוד. נדון במימושי הפעולות עבור הייצוגים השונים ונשווה את יעילותם. לסיכום תתבקשו לכתוב מימוש מלא של המשימה המקורית בעזרת טיפוס האוסף החדש.

הגדרת המשימה :

אנו מעוניינים לנהל מאגר המכיל שמות של אנשים ומספרי הטלפון שלהם.

לשם הפשטות, נניח כי לכל שם במאגר יש מספר טלפון יחיד, כלומר שם אינו מופיע יותר מפעם אחת באוסף. כמובן, שייתכן כי לגבי כהן ולמר כהן יש אותו מספר טלפון, אך לא נאפשר מצב בו לאדם אחד יהיה יותר ממספר טלפון אחד.

אנו מעוניינים לבצע על המאגר (לפחות) את הפעולות האלה :

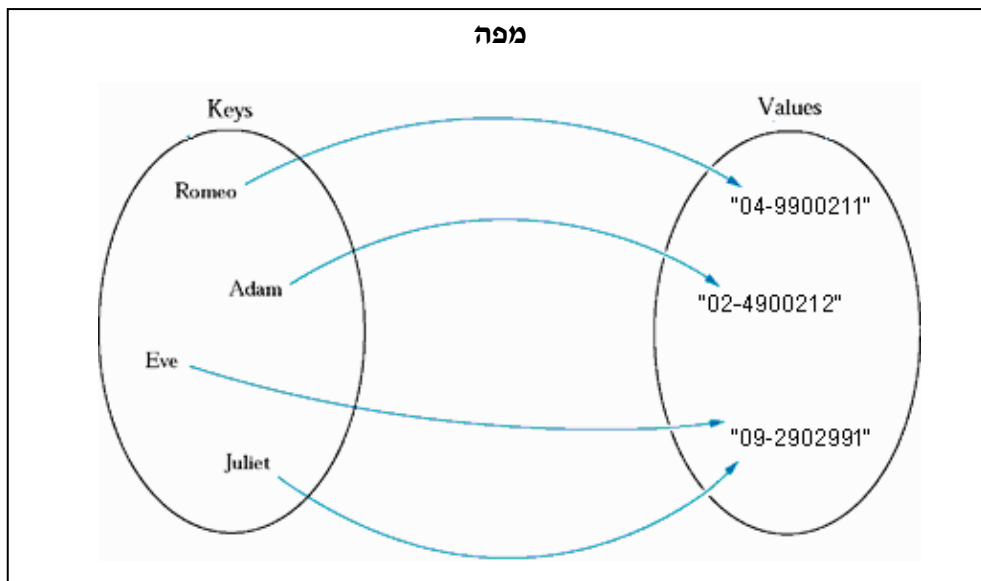
1. לאחזר מספר טלפון כאשר נתון שם.
2. להוסיף למאגר זוג חדש של שם ומספר טלפון.
3. לעדכן מספר טלפון של אדם.
4. למחוק מהמאגר זוג קיים של שם ומספר טלפון.

למעשה, זהו תיאור של ספר טלפונים פשוט.

א. המפה

בניהול אוסף נתונים כמו זה המתואר לעיל, מקובל להתייחס לשם האדם ברשימה כ**מפתח** (key) ולמספר הטלפון כ**ערך** (value). כיוון ששם אינו מופיע יותר מפעם אחת, האוסף הוא למעשה **מיפוי** בין המפתחות לערכים.

מיפוי ניתן לתיאור כך :



טיפוס הנתונים המתאר מיפויים כאלה קרוי **מפה (map)**. הטיפוס מפה הוא אם כן סוג אוסף כללי ודינמי, שמופע שלו הוא קבוצה של קשרי מפתח-ערך. מפה אינה מוגבלת במספר הקשרים שהיא יכולה להכיל, ולא מוגדר בה סדר כלשהו של הנתונים. באמצעות מפתח ניתן להגיע ישירות לערך הקשור אליו במפה.

אוספי נתונים רבים הם למעשה מפות, נדגים :

1. "מילון" הוא מפה שבה המפתח הוא מילה, והערך הוא הפירוש של המילה.
 2. "יומן" או "לוח פגישות" הוא מפה שבה המפתח הוא זמן (תאריך ושעה), והערך הוא תיאור הפגישה.
 3. "אינדקס של ספר" הוא מפה שבה המפתח הוא מילה, והערך הוא סדרת מספרי העמודים שבהם מופיעה המילה.
 4. "יומן ציונים של מורה" היא מפה שבה המפתח הוא שם תלמיד, והערך הוא ציונו במקצוע שמלמד המורה.
- כדי לממש את הטיפוס מפה בשפת התכנות עלינו לדון תחילה בפעולות הנחוצות, ולקבץ אותן לממשק הטיפוס.

ב. דיון בפעולות הממשק

בדיון על הפעולות הנחוצות לנו, נסמן את טיפוס המפתחות ב-K (מלשון key), ואת טיפוס הערכים ב-V (מלשון value).

מרבית הפעולות שנגדיר עבור המפה יקבלו כפרמטר את המפתח או את הערך הקשור אליו. אנו מניחים שהן המפתח והן הערך הקשור אליו אינם null.

כבר הזכרנו אֶחָדוֹת מהפעולות שאנו רוצים לבצע על מפה: אחזור ערך הקשור למפתח, תוספת זוג מפתח-ערך ושינוי הערך הקשור למפתח נתון. בממשק נוח יותר לאחד את שתי הפעולות האחרונות לפעולה אחת.

אם כך, בממשק מופיעות הפעולות האלה:

אחזור ערך הקשור למפתח

```
public V GetValue(K key)
```

הפעולה מחפשת במפה את הערך הקשור למפתח key ומחזירה אותו. הפעולה מניחה שהמפתח key קיים במפה.

בדיקה האם קיים מפתח

כדי לאפשר למשתמש להימנע מזימון הפעולה GetValue(...) כאשר המפתח אינו קיים, יש להוסיף לממשק הטיפוס מפה את הפעולה:

```
public bool ContainsKey(K key)
```

פעולה זו מחזירה 'אמת' אם המפתח המסוים קיים במפה ו-'שקר' אחרת.

הוספה או עדכון מפתח-ערך

```
public void Insert(K key, V value)
```

הפעולה מבצעת הוספת זוג או עדכון קשר קיים, לפי מצב המפה (נזכיר כי ערכי שני הפרמטרים אינם null):

אם מפתח זה אינו קיים במפה – הפעולה מוסיפה למפה את הזוג: key-value.

אם המפתח קיים במפה, הערך הקשור אליו יעודכן לערך value.

לכאורה, די בפעולות אלה, אך שימו לב: באמצעות פעולות אלה ניתן לשנות ערך הקשור למפתח, אך אי אפשר להוציא את המפתח מהמפה. כמו כן, נניח כי אנו זוכרים שהכנסנו למפה לפני זמן רב מספר טלפון של חבר רחוק, אך איננו זוכרים כרגע את שמו, או את האיות המדויק של השם. במקרה זה נרצה לחפש את מספר הטלפון, אך ניתן לראות כי הפעולות שהגדרנו מאפשרות למצוא ערך רק כאשר המפתח ידוע לנו. אם נסתפק בפעולות שהגדרנו, הדרך היחידה לגלות את מספר הטלפון של החבר תהיה לנסות להכניס בזה אחר זה כל שם אפשרי, עד שנתקל בשמו. זה אינו פתרון סביר, שכן מספר השמות האפשריים גדול מאוד. באופן דומה, אפשר לראות כי הפעולות הנזכרות אינן מאפשרות לבצע מחוץ למחלקה Map פעולות כגון מציאת המפתח המינימלי הקיים במפה או אחזור כל הזוגות שבה.

? נמקו את הטענה האחרונה: "הפעולות הנזכרות אינן מאפשרות לבצע מחוץ למחלקה Map פעולות כגון מציאת המפתח המינימלי הקיים במפה או אחזור כל הזוגות שבה".

כדי שנוכל לבצע משימות מורכבות יותר, כפי שתיארנו, יש להוסיף פעולות לממשק:

הוצאת מפתח מהמפה

```
public V Remove(K key)
```

הפעולה מוציאה מהמפה את המפתח `key` ואת הערך המצורף אליו. נוח לנו שהפעולה תחזיר את הערך שהיה בזוג, שהוצא מהמפה, כפי שעשינו בפעולת ההוצאה מרשימה. הפעולה מניחה שהמפתח `key` קיים במפה.

אחזור כל המפתחות שבמפה

```
public K[] GetAllKeys()
```

הפעולה מחזירה מערך ממוין בסדר עולה ובו כל המפתחות הקיימים במפה. אם המפה ריקה, גודל המערך שיוחזר יהיה 0.

? הסבירו כיצד הפעולה `GetAllKeys()` מאפשרת לבצע את כל פעולות החיפוש שלא התאפשרו קודם לכן.

ב.1. גרירות של מפתח וערך

לפני שנציג את הממשק המתקבל, עלינו עוד לדון בשאלה: האם הטיפוסים `V`, `K` יכולים להיות טיפוסים גנריים בהגדרת המחלקה `Map`?

מעיון בתיאורי הפעולות ברור כי אין צורך לבצע במחלקה שום פעולה הייחודית לטיפוס `V`. אנו מכניסים ערכים מטיפוס זה למפה, ואחר כך מוציאים אותם כערכי החזרה. לכן הטיפוס `V` יכול להיות טיפוס גנרי. לעומת זאת, פעולות כגון `GetValue(...)`, `Remove(...)` דורשות השוואת מפתח נתון למפתח המאוחסן במפה, והפעולה `GetAllKeys()` מחייבת מיון של המפתחות בסדר עולה. פעולת השוואה המאפשרת מיון של קבוצת ערכים אינה קיימת לכל טיפוס העצמים בשפה. זוהי פעולה ייחודית הקיימת במחלקה רק אם הוגדרה בה בפירוש. כיוון שאנו נדרשים לפעולה ייחודית על המפתחות, טיפוס המפתח אינו יכול להיות גנרי. לצורך המשימה שלנו, נקבע שטיפוס המפתח יהיה מחרוזת.

הסעיף הבא הוא סיכום של ממשק טיפוס הנתונים מפה. המפתח הוא מטיפוס מחרוזת ואילו הערך – גנרי.

2.ב. ממשק המחלקה Map<V>

המחלקה מגדירה אוסף דינמי הממפה מפתחות וערכים. המפתחות במפה יהיו מטיפוס מחרוזת בעוד הערכים יהיו גנריים.

Map()	הפעולה בונה מפה ריקה
bool ContainsKey (string key)	הפעולה מחזירה 'אמת' אם המפתח key קיים במפה הנוכחית ו-'שקר' אחרת
V GetValue (string key)	הפעולה מחזירה את הערך הקשור למפתח key. הנחה : המפתח קיים במפה
void Insert (string key, V value)	הפעולה מוסיפה למפה הנוכחית את המפתח key ואת הערך value הקשור אליו. אם המפתח key קיים במפה, הפעולה מעדכנת את הערך הקשור אליו בערך value שהתקבל
V Remove (string key)	הפעולה מוציאה מהמפה הנוכחית את המפתח key ואת הערך הקשור אליו. הפעולה מחזירה את הערך הקשור למפתח שהוצא מהמפה. הנחה : המפתח קיים במפה
string[] GetAllKeys()	הפעולה מחזירה את אוסף המפתחות שקיימים במפה הנוכחית ממוין בסדר אלפביתי עולה. אם המפה ריקה, יוחזר מערך בגודל אפס
string ToString()	הפעולה מחזירה מחרוזת המתארת את המפה כך : [key1:value1, key2:value2, key3:value3,...]

ג. שימוש במפה

כפי שצינו בתחילת הפרק, לאוסף Map יש שימושים רבים: יומן פגישות, אינדקס של ספר, רשימת ציונים ועוד. אנו נציג כאן שימוש פשוט. בתרגילים בסוף הפרק תכירו דוגמאות נוספות מורכבות יותר, כגון מילון וספר טלפונים.

דוגמה: "זמר השנה"

קבוצת זמרים משתתפת בתחרות שירה. כדי לקבוע מיהו הזמר הזוכה נערכת הצבעה. על המשתתפים בהצבעה לבחור בשם הזמר המועדף עליהם. שם הזמר מזהה אותו באופן מוחלט (כלומר אין שני זמרים בעלי אותו שם). מערכת ממוחשבת צוברת עבור כל זמר את מספר הקולות שקיבל. הזמר שיקבל את מספר הקולות הרב ביותר יזכה בתואר "זמר השנה".

מאחר שמספר הזמרים אינו ידוע, נצטרך להגדיר טיפוס אוסף דינמי של זוגות, שם ומספר. כיוון שאנו מעוניינים להגיע בכל שלב בתהליך ישירות אל הזמרים על פי שמם, תהיה המפה טיפוס האוסף המתאים לשמירת הנתונים. המפה אינה שומרת על סדר בין הזמרים ואינה מגבילה את

מספרם באוסף. המפה, אם כן, מייצגת את האוסף שבידינו כראוי. שם הזמר ישמש כמפתח, ומספר הקולות שבהם זכה הזמר יהיה הערך הקשור אליו.

הבה נניח כי לצורך ניהול תהליך ההצבעה הוגדרה מחלקה כלהלן:

```
public class Elections
{
    private Map<int> votes; // מפה לשמירת ההצבעות

    public Elections()
    {
        this.votes = new Map<int>();
    }

    public void VoteFor(string name)
    {
        if (!this.votes.ContainsKey(name)) // הוספת זמר חדש למפה
            this.votes.Insert(name, 1);

        else // הוסף הצבעה לזמר קיים
            this.votes.Insert(name, this.votes.GetValue(name)+1);
    }

    // הנחה: התבצעה הצבעה אחת לפחות
    public string FindWinner()
    {
        string[] allNames = this.votes.GetAllKeys();
        string winnerName = allNames[0];

        for(int i=1; i < allNames.Length; i++)
            if(this.votes.GetValue(allNames[i]) >
                this.votes.GetValue(winnerName))
                winnerName = allNames[i];

        return winnerName;
    }

    public override string ToString()
    {
        string str = "";
        string[] allNames = this.votes.GetAllKeys();

        for (int i = 0; i < allNames.Length; i++)
            str = str + allNames[i] + "\t" +
                this.votes.GetValue(allNames[i]) + "\n";

        return str;
    }
}
```

כאשר נוצר עצם מהמחלקה, הוא מכיל מפה ריקה. הפעולה VoteFor(...) מאפשרת להצביע עבור מועמד. הפעולה FindWinner(...) מאפשרת בסיום תהליך ההצבעה, למצוא את שם המנצח. ההנחה שקיים מנצח יחיד תפשט את הקוד. במציאות, יתכן מצב שבו שני מתמודדים או יותר

זוכים באותו מספר קולות, ועל התוכנה להתמודד עם מצב זה. בהינתן המחלקה Elections ניתן לדמות את תהליך ההצבעה שבסיומה מוכרז שמו של "זמר השנה":

```
public static void Main (string[] args)
{
    Elections singerElections = new Elections();

    Console.Write ("Enter your favorite singer name: ");
    string name = Console.ReadLine();

    while (!name.Equals(""))
    {
        singerElections.VoteFor (name);
        Console.Write ("Enter your favorite singer name
                        (to stop press enter): ");
        name = Console.ReadLine();
    }

    Console.WriteLine();

    // הדפסת כל ההצבעות בתחרות
    Console.WriteLine (singerElections);

    // הכרזה על הזוכה בתחרות "זמר השנה". הפלט: שם הזמר הזוכה
    Console.WriteLine ("The winner is: " +
                      singerElections.FindWinner());
}
```

ד. ייצוגים אפשריים למפה

כדי לכתוב את המחלקה מפה עלינו לבחור ייצוג לנתונים ולממש את הפעולות בהתאם. נבחן כמה ייצוגים אפשריים.

1.ד. ייצוג בעזרת מערך

נתחיל במבנה הנתונים הוותיק מערך. מערך הוא אוסף של איברים המאפשר לגשת לאיבר על פי אינדקס (מספר שלם) המציין את מקומו הסידורי באוסף. אם i הוא אינדקס במערך, ניתן לאחזר את האיבר שנמצא במקום i או להציב איבר חדש במקומו. אם כן, מערך אף הוא מפה, שבה המפתחות הם מספרים (אינדקסים). האם נוכל להשתמש במערך לאחסון הנתונים במחלקה Map כאשר המערך מייצג את המיפוי של מפתחות וערכיהם?

התשובה שלילית. האינדקסים במערך הם תמיד ערכים שלמים $0, 1, 2, \dots, n-1$ (הוא גודל המערך). כלומר המפתחות שיעמדו לרשותנו בזמן שימוש במערך יוכלו להיות מפתחות מספריים בלבד בתחום הנתון. סביר שלא נרצה להיות מוגבלים בערכי המפתחות, ולכן המסקנה היא שייצוג ישיר של מפה בעזרת מערך אינו אפשרי.

ניתן להציע ייצוג מתוחכם יותר של מפה המשתמש בשני מערכים.

? הציעו ייצוג למפה המשתמש בשני מערכים.

לייצוג זה יש מגבלה, והיא העובדה שגודלו של מערך נקבע בזמן יצירתו, בעוד מפה היא אוסף דינמי. אפשר להתמודד עם מגבלה זו על ידי הקצאת זוג מערכים גדולים יותר והעתקת הנתונים אליהם בכל פעם שהמערכים מתמלאים, אך כפי שנהגנו לאורך היחידה, אנו נעדיף ייצוג דינמי "אמיתי".

2.ד. ייצוג בעזרת רשימה

בחירה טבעית לייצוג מבנים דינמיים היא שרשרת חוליות. כיוון שאנו זקוקים לפעולות הכנסה והוצאה, נוכל להשתמש ברשימה ה"עוטפת" שרשרת חוליות יחד עם פעולות הכנסה והוצאה. כיוון שכל מיפוי הוא של זוג (מפתח וערך), ניתן לייצג מפה על ידי שתי רשימות (כשם שהצענו קודם בדיון בייצוג בעזרת מערך).

? הציעו ייצוג למפה המשתמש בשתי רשימות.

לייצוג זה חיסרון – הוא מפריד את המפתח מהערך הקשור אליו. טעות קטנה בתכנות יכולה לגרום לכך שפעולה תפעל על מפתח, ועל ערך שאינו הערך הקשור אליו. עדיף לחשוב על הרעיון של ייצוג הזוג מפתח-ערך בעזרת טיפוס נתונים עצמאי – Pair. המפה עצמה תיוצג בעזרת רשימה של זוגות כאלה.

1.2.ד. המחלקה Pair

המחלקה הגנרית Pair<K, V> מגדירה זוג: מפתח גנרי וערך גנרי. לפניכם תרשים UML של המחלקה:

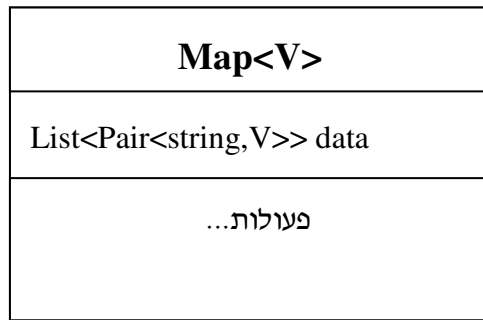
Pair<K, V>
K key V value
Pair(K key, V value) K GetKey() V GetValue() void SetKey(K key) void SetValue(V value) string ToString()

כיוון שאין במחלקה שימוש במפתחות ובערכים, פרט לשליפה ועדכון שלהם, שני הטיפוסים הם גנריים.

ד.2.2. ייצוג מפה בעזרת רשימת זוגות

לאחר שיש בידינו את הטיפוס זוג, נוכל לייצג את המפה עצמה בעזרת רשימה של זוגות כאלה. תכונת המחלקה מפה תהיה רשימה קונקרטית, שטיפוס הערך המאוחסן בחוליותיה יהיה זוג. המפתח בזוג יהיה מחרוזתי, והערך הקשור אליו גנרי. ההגדרה מורכבת מעט אך הבה נראה כיצד היא נראית.

לפניכם תרשים UML של המחלקה מפה בייצוג זה:



לפני שנתחיל בכתיבת המחלקה Map, נשים לב שעל פי הגדרתה, הפעולה GetAllKeys() מחזירה את אוסף המפתחות שבמפה כשהוא ממיון בסדר עולה. ברשימה הכיתתית נתקלנו במקרה דומה, וכמו שם, גם כאן עומדות לפנינו שתי חלופות למימוש הפעולה: רשימת הזוגות במפה תישמר בלתי ממוינת, והמיון יתבצע כחלק ממימוש הפעולה GetAllKeys(); או שרשימת הזוגות במפה תישמר ממוינת כל הזמן (המיון יישמר בפרט בעת ביצוע הכנסת זוגות לרשימה). באפשרות השנייה מימוש הפעולה GetAllKeys() פשוט יותר. ההבדל בין החלופות יתבטא בעיקר ביעילות של הפעולות.

הערה: המחלקות Map1, Map2 שתכתבו במשימות הבאות יממשו כולן את ממשק המחלקה Map שהוצג בראשית הפרק. ממשימה למשימה ישתנו רק הייצוגים ומימושי הפעולות. בכל זאת, כדי שתוכלו לשוב ולעיין במחלקות השונות, נשמור כל משימה תחת שם מחלקה שונה.

משימה 1: כתיבת המחלקה Map1 המיוצגת בעזרת רשימת זוגות

- א. כתבו את המחלקה Pair<K, V> על פי תרשים ה-UML המופיע לעיל.
- ב. כתבו את המחלקה Map1<V> שתיוצג בעזרת רשימת זוגות.
- ג. נתחו את יעילות הפעולות של המחלקה Map1.
- ד. אילו שינויים יש לעשות במחלקה Map1, שכתבתם בסעיף ב, כדי שיעילות הפעולה GetAllKeys() תהיה לינארית בגודל המפה? נמקו.

3.4. ייצוג בעזרת עץ-חיפוש-בינרי

בפרק הקודם הצגנו את עץ-החיפוש-הבינרי, וציינו שרעיון השימוש בעץ שכזה מעניין יותר אם כל צומת בו מכיל מפתח ומידע נוסף, לדוגמה: תעודות זהות שאליהן מקושרים נתוני אנשים במרשם התושבים, מספרי סטודנט המקושרים לרשומות הסטודנטים באוניברסיטה, או רשימת שמות שאליהם קשורים מספרי טלפון. הפעם נשמור בעץ-החיפוש-הבינרי חוליות שבהן שדה המידע הוא זוג מפתח-ערך. ארגון העץ וביצוע הפעולות ייקבעו על ידי רכיב המפתח שבזוג.

ייצוג המפה באמצעות עץ-חיפוש-בינרי יאפשר לנו לשפר את יעילות הפעולות השונות של המפה. דיון מפורט ביעילות הפעולות מופיע להלן.

התכונה במחלקה Map2, בייצוג החדש, תהיה תכונה פרטית מטיפוס זה:

```
BinTreeNode<Pair<string, V>>
```

כאשר ערך התכונה יהיה **null**, פירוש הדבר הוא כי המפה ריקה. זה המצב, למשל, מיד אחרי יצירת המפה. בכל מקרה אחר, התכונה תחזיק הפניה לשורשו של עץ חוליות בינרי המאורגן כעץ-חיפוש-בינרי. פעולות החיפוש, ההכנסה, ההוצאה והאחזור מעץ-החיפוש ישתמשו ברכיב המפתח מתוך הזוג השמור בכל חוליה.

את פעולות המפה: `Remove(...)`, `Insert(...)`, `GetAllKeys(...)` ניתן לממש על ידי שימוש בפעולות המוכרות של עץ-חיפוש-בינרי, תוך ביצוע התאמות קטנות. ההתאמות נובעות מכך שכאן אנו מתייחסים לזוגות, ומקומו של זוג נקבע לפי ערך הרכיב הראשון בזוג - המפתח, ולכן גם החיפוש משתמש בערכי המפתחות בזוגות. הפעולה `GetValue(...)` תמומש באופן דומה לפעולה `ExistsInBST(...)` לחיפוש בעץ-חיפוש-בינרי, בהבדל קטן - הפעולה תחזיר את ערך המפתח במקום ערך בוליאני.

משימה 2: כתיבת המחלקה Map2 המיוצגת בעזרת עץ-חיפוש-בינרי

א. כתבו את המחלקה `Map<V>` מחדש. הפעם יצגו אותה בעזרת עץ חוליות בינרי המאורגן

כעץ-חיפוש-בינרי מטיפוס `Pair<string, V>`, שבו מיקום זוג נקבע לפי ערך המפתח.

ב. נתחו את יעילות הפעולות של המחלקה `Map2`.

1.3.4. יעילות המימוש בעזרת עץ-חיפוש-בינרי

בניתוח היעילות שלפניכם, n הוא מספר הצמתים בעץ.

יעילותה של הפעולה הבונה את המפה היא $O(1)$, שהרי כל שעליה לעשות הוא להציב את הערך

null בתכונה הפרטית של המחלקה `Map2`.

יעילות הפעולות `ToString()` ו-`GetAllKeys()` תהיה מסדר גודל $O(n)$. `GetAllKeys()` צריכה לסרוק את העץ בסריקה בסדר תזכי המבטיח שהערכים יישלפו על פי סדר ממיון של המפתחות השמורים בעץ. הפעולה `ToString()` תבצע סריקה כלשהי של העץ או תשתמש במערך שהחזירה `GetAllKeys()`.

שאר הפעולות יהיו תלויות בגובהו של העץ. אם עץ-חיפוש יישאר מאוזן בכל השלבים, אזי פעולות הממשק `Insert(...)`, `Remove(...)`, `GetValue(...)`, `ContainsKey(...)` יהיו מסדר גודל $O(\log n)$ (זהו המקרה הממוצע). אחרת, במקרה הגרוע ביותר (של עץ בלתי מאוזן לגמרי שבו צומת אחד בכל רמה), יהיו הפעולות מסדר גודל לינארי.

ה. השוואת היעילות של הייצוגים השונים

האם שיפרנו במשהו את יעילות הפעולות של המפה כאשר במקום לייצגה בעזרת רשימה של זוגות, ייצגנו אותה בעזרת עץ בינרי של זוגות המאורגן כעץ-חיפוש-בינרי?

כפי שראינו בדיוננו הקודם במפה המיוצגת בעזרת רשימה, יעילות פעולת החיפוש ברשימה היא מסדר גודל $O(n)$. בעץ-חיפוש-בינרי, אם הוא מאוזן, סדר הגודל של פעולת החיפוש נמוך יותר, והוא $O(\log n)$. ואולם, הכנסה לרשימה, כאשר מקום ההכנסה אינו חשוב, נעשית בסדר גודל $O(1)$, בעוד בעץ-חיפוש-בינרי מקום ההכנסה אינו נתון לבחירתנו, ולכן יעילות ההכנסה אליו היא $O(\log n)$, אם הוא מאוזן. נניח שבונים עץ בינרי מאוזן, שרוב הפעולות עליו לאחר הבנייה הן חיפושים ורק מדי פעם יש לבצע גם פעולת הכנסה או הוצאה, הרי הרווחנו, כי המחיר הממוצע של הפעולות ירד משמעותית.

נסכם את כל חישובי היעילות של הייצוגים השונים :

פעולות המפה	עץ חוליות בינרי המאורגן כעץ-חיפוש	רשימה ממוינת	רשימה לא ממוינת
Map()	O(1)	O(1)	O(1)
bool ContainsKey (string key)	O(log n) במקרה הממוצע O(n) במקרה הגרוע	O(n)	O(n)
V GetValue (string key)	O(log n) במקרה הממוצע O(n) במקרה הגרוע	O(n)	O(n)
void Insert (string key, V value)	O(log n) במקרה הממוצע O(n) במקרה הגרוע	O(n)	O(1)
V Remove (string key)	O(log n) במקרה הממוצע O(n) במקרה הגרוע	O(n)	O(n)
string[] GetAllKeys()	O(n)	O(n)	O(n log n)
string ToString()	O(n)	O(n)	O(n)

1. מפה – טיפוס נתונים מופשט

המפה, כפי שהוגדרה בתרגיל זה, היא טיפוס אוסף כללי. כיוון שהמחלקה Map מסתירה את אופן הייצוג שנבחר עבור המפה וניתן להחליף את הייצוג שלה בלי לשנות את ממשק המחלקה, וכיוון שפעולות הממשק, עבור כל אחד מהייצוגים, שומרות על מבנה המפה (לא ניתן להכניס למפה זוגות בעלי מפתח זהה), המפה מהווה טיפוס נתונים מופשט.

את המפה נוסף לארגז הכלים, שאליו כבר אספנו לאורך הלימוד מבני נתונים ומחלקות המגדירות אוספים כלליים שימושיים, היכולים לשמש לייצוג אוספים ספציפיים שונים.

בארגז הכלים יהיו כעת :

- טיפוס האוסף המופשטים : מחסנית, תור ומפה והטיפוס רשימה ;
- מבני הנתונים : שרשרת חוליות, עץ חוליות בינרי ועץ-חיפוש-בינרי ;
- המחלקות : BinTreeNode, Node.

בסוף פרק עצים הוצג הסבר, שעוזר לנו להבין מדוע גם כאן הצלחנו להסתיר את המימוש של המפה. המפה עוסקת בערכים השמורים בה בלבד, והידע של המשתמש מוגבל לקשרים בין הפעולות. המשתמש אינו מתייחס כלל לארגון הפנימי של האוסף, ואין לו כל השפעה או ידע על ייצוג האוסף ועל מקום הערכים בו.

בשפת סישרפ יש ספריה המציעה ממשק של מפה, וכן כמה מימושים של ממשק זה, במחלקות שונות. אחד המימושים האלה משתמש בעצי-חיפוש-בינריים ובאלגוריתמים המבצעים איזון מחדש במקרה הצורך, וכך מבטיחים יעילות שהיא לוגריתמית במקרה הגרוע ביותר. מימוש זה תומך בתפיסת קבוצת המפתחות במפה כקבוצה ממוינת, ויש לו כמובן יתרון ברור על ייצוג מפה בעזרת רשימה ממוינת. ייצוג אחר של מפה, יעיל אף יותר מכך, אך כזה שאינו תומך במיון של המפתחות, הוא ייצוג על ידי מבנה הקרוי "טבלת ערבול" (hash table). מכאן ברור שמפה בסישרפ היא טיפוס נתונים מופשט: היא מגדירה טיפוס על ידי פעולות ממשק בלבד, אינה חושפת אף פרט מהייצוג ומהמימוש ומאפשרת למעשה ייצוגים ומימושים שונים מאוד זה מזה.

ז. סיכום

- טיפוס האוסף 'מפה' מאפשר לשמור אוסף דינמי של קשרי מפתח-ערך.
- ממשקי הפעולות של מפה אינם מתייחסים לייצוג ולמימוש. ייצגנו את המפה כרשימת זוגות, אך ממשקי הפעולות אינם מתייחסים כלל למושג זה, למחלקה Pair ולרשימה. ניתן בקלות להחליף את ייצוג המחלקה בלי לשנות את ממשקי הפעולות, ואכן בחנו מימושים אפשריים אחרים למפה.
- המפה היא טיפוס נתונים מופשט: אופן הייצוג והמימוש שלה אינם חשופים דרך הממשק, ניתן להחליף את הייצוג ואת המימוש הבנוי עליו באחר ואין דרך לקלקל את מבנה המפה באמצעות פעולות הממשק (הכנסת זוגות בעלי מפתח זהה).

מושגים

Map	מפה
Value	ערך
Key	מפתח

תרגילים

שאלה 1

נתונה המפה studentsPool שהוגדרה כך :

```
Map<Student> studentsPool = new Map<Student> ();
```

המפה מייצגת מאגר תלמידים הממפה מספרי תעודת זהות לתלמידים. כלומר, המפתחות במפה הם מספרי תעודות הזהות של התלמידים והערכים הקשורים למפתחות הם עצמים מטיפוס Student (ראו את המחלקה Student בפרק 6).

בצעו את המשימות שלפניכם, בהנחה שמאגר התלמידים המיוצג על ידי המפה studentsPool אינו ריק :

א. כתבו קטע תוכנית המדפיס את שמות כל התלמידים הגרים בירושלים ובסביבתה (קידומת מספר הטלפון שלהם היא "02").

ב. כתבו קטע תוכנית המדפיס את מספרי הזהות של כל התלמידים ששם ההורה האחראי עליהם הוא "משה כהן".

ג. כתבו קטע תוכנית המעדכן את מספר הטלפון הנייד של ההורה של התלמיד שמספר הזהות שלו הוא : "07845602". המספר החדש יהיה "050-1234567". הניחו שהתלמיד קיים במאגר.

שאלה 2

המשימה שהוצגה בתחילת הפרק הייתה כתיבת ספר טלפונים, עתה הגיע הזמן לבצע את המשימה.

חזרו ובצעו מחדש את כל הסעיפים שבדף עבודה 5 מפרק 6 – "ספר טלפונים". השתמשו במחלקה Map לצורך ייצוג אוסף אנשי הקשר בספר הטלפונים.

פרק 11 דף עבודה 1

מילון

ברצוננו לפתח מילון (Dictionary) אנגלי-עברי ממוחשב המכיל אוסף לא מוגבל של מילים באנגלית ותרגומן לעברית. לכל מילה (word) במילון יש תרגום (translation) אחד לפחות.

ממשק המחלקה Dictionary

המחלקה מגדירה מילון, שהוא אוסף של מילים ותרגומיהן.

Dictionary()	הפעולה בונה מילון ריק
void AddWord (string word, string translation)	הפעולה מוסיפה למילון מילה ותרגומה. אם המילה קיימת במילון, פעולה זו מוסיפה תרגום נוסף למילה. אם התרגום למילה קיים, לא יתבצע כל שינוי
void DelWord (string word)	הפעולה מוחקת מהמילון את המילה word יחד עם כל תרגומיה.
string[] GetAllWords()	הפעולה מחזירה את אוסף המילים המופיעות במילון, ממוין בסדר אלפביתי עולה. אם המילון ריק, יוחזר מערך בגודל אפס
string[] GetTranslations(string word)	הפעולה מחזירה את אוסף התרגומים של המילה word, ממוין בסדר אלפביתי עולה. אם המילה אינה מופיעה במילון, יוחזר מערך בגודל אפס
string ToString()	הפעולה מחזירה מחרוזת המתארת את המילון כך: word1: translation1, translation2, ... word2: translation1, translation2, ... : המילים יופיעו בשורות נפרדות בסדר אלפביתי עולה, והתרגומים של כל מילה יופיעו בסדר אלפביתי עולה לאחר המילה באותה שורה.

מה עליכם לעשות?

חלק א

1. כדי לכתוב את המחלקה Dictionary עליכם לבחור ייצוג מתאים.

שימו לב לכללים האלה:

- אין כפילות של מילים במילון.
- אין כפילות של תרגומים עבור אותה המילה.
- הגישה לתרגומים היא דרך המילה.
- מספר המילים במילון אינו מוגבל.
- מספר התרגומים לכל מילה אינו מוגבל.

הערות חשובות:

- א. בשל מגבלת השפה, התרגומים ייכתבו בעברית אך באותיות אנגליות.
- ב. לצורך שאלה זו, כל הפעילות במילון ומחוצה לו אינה מבחינה בין אותיות גדולות וקטנות (כלומר winter ו-Winter הן אותה המילה).

2. כתבו תוכנית ראשית הקולטת מחרוזת המייצגת משפט תקני באנגלית, ומדפיסה את תרגומו המילולי לעברית (תרגמו מילה במילה ללא תלות בהקשר).

השלבים:

- א. צרו מילון ובו מילים ותרגומן כרצונכם (לפחות עשר מילים).
 - ב. קלטו משפט באנגלית ותרגמו אותו לעברית בעזרת המילון. אם יש לאחת המילים יותר מתרגום אחד, הדפיסו את הראשון. אם אין למילה תרגום, הדפיסו "!".
- הנחיה:** כדי לקלוט משפט ולפרקו למילים, אתם יכולים:
- להיעזר בפעולה Split של המחלקה string שתאפשר לכם לפרק משפט שלם למילים.
 - או, לבצע קלט של מילים בודדות.

3. ברצוננו לייצר מילון **עברי-אנגלי** מתוך מילון **אנגלי-עברי** קיים. במילון זה ישמשו התרגומים שהכנסתם בסעיף כמילים השמורות במילון החדש. עבור כל תרגום שכזה תופיע המילה המקורית כתרגום. לדוגמה:

אם במילון האנגלי-עברי הופיעו: **indenture: heskem, hoze, amana**

(התרגומים הם כמובן: הסכם, חוזה, אמנה).

אזי במילון העברי-אנגלי יופיעו: **amana : indenture**

heskem : indenture

hoze : indenture

הוסיפו לתוכנית הראשית, פעולה המקבלת את המילון האנגלי-עברי שיצרתם בסעיף א.1. ומייצרת ממנו מילון עברי-אנגלי מתאים. הפעולה תדפיס את תכולת המילונים.

חלק ב

1. הוסיפו למחלקה Dictionary את הפעולה:

```
public List<string> TranslateTo (string translation)
```

הפעולה מחזירה את אוסף המילים במילון שאחד מתרגומיהן הוא translation.

2. הוסיפו למחלקה Dictionary את הפעולה:

```
public List<string> SpellCheck (string sentence)
```

הפעולה מקבלת משפט ומחזירה את אוסף המילים המופיעות במשפט שאין להן תרגום במילון. נתחו את יעילות הפעולה.

בהצלחה!