פרק 8
תרגיל מסכם : מפה

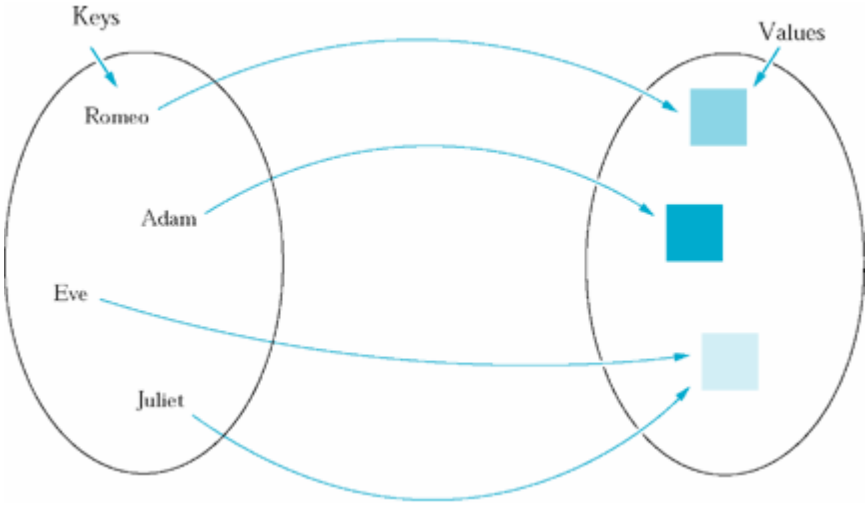
תרגיל מעשי: הגדרת טיפוס נתונים מופשט Map וייצוגו בעזרת רשימה

בפרק זה נלמד על צורך נפוץ לניהול אוסף נתונים באופן מסוים. ננתח את הבעיה, נגדיר את הפעולות הדרושות על אוסף הנתונים ועל פיהן נגדיר את ממשק טיפוס הנתונים שיגדיר את האוסף. נבחן ייצוגים שונים האפשריים עבור האוסף ופעולותיו ונתמקד בייצוג המתרגל את השימוש ברשימה כמבנה נתונים. כך נכיר טיפוס נתונים חשוב ושימושי שילווה אותנו בהמשך היחידה, ונתרגל את כל הנושאים שנלמדו ביחידה עד כה.

נניח כי אנו מעוניינים לנהל מאגר המכיל שמות של אנשים ומספרי הטלפון שלהם. לשם פשטות, נניח כי לכל שם במאגר יש מספר טלפון יחיד, כלומר שם אינו מופיע יותר מפעם אחת באוסף. כמובן, יתכן בהחלט שלגב' כהן ומר כהן יש אותו מספר טלפון.
אנו מעוניינים לבצע על מאגר זה (לפחות) את הפעולות הבאות: לאחזר מספר טלפון כאשר נתון שם; להוסיף למאגר זוג חדש של שם ומספר טלפון ולעדכן מספר טלפון של אדם. למעשה, זהו תיאור של ספר טלפונים פשוט.

א. טיפוס הנתונים מפה

בניהול אוסף הנתונים המתואר לעיל, מקובל להתייחס לשם כ**מפתח** (**key**) ולמספר הטלפון כ**ערך** (**value**). כיון ששם אינו מופיע יותר מפעם אחת, האוסף הוא למעשה **מיפוי** בין המפתחות לערכים. משום כך טיפוס הנתונים המתואר קרוי בשם **מפה** (**Map**).
מפה היא אוסף נתונים דינמי, של קשרי מפתח-ערך. בהגדרה זו אנו אומרים למעשה שהמפה אינה מוגבלת במספר הקשרים שהיא יכולה להכיל. במפה לא מוגדר סדר כלשהו.
המיפוי מתואר באיור הבא:



אוסף נתונים רבים הם למעשה מפות, נדגים:

1. "מילון" הוא מפה שבה המפתח הוא מילה, והערך הוא פירוש של המילה.
2. "יומן" או "לוח פגישות" הוא מפה שבה המפתח הוא זמן (תאריך ושעה) והערך הוא תיאור פגישה.
3. "אינדקס של ספר" הוא מפה שבה המפתח הוא מילה והערך הוא סדרת מספרי העמודים שבהם מופיעה המילה.
4. "רשימת ציונים" היא מפה שבה המפתח הוא שם תלמיד והערך הוא ציונו בקורס.

ב. האוסף מפה בשפת ג'אווה

בחיי יום יום יש לנו דרכים מגוונות לאחזקת האוסף והטיפול בו: פנקס כתובות וטלפונים, רשימות ממוחשבות בתוכנות מתאימות, מפות נתונים השמורות בטלפון הנייד ועוד. לצורך כתיבת יישומים המשתמשים במפות, נציג את המחלקה Map המגדירה את האוסף והפעולות עליו.

ב.1. הפעולות

בדיון הבא בנושא הפעולות, נסמן את טיפוס המפתחות ב-K (מלשון key), ואת טיפוס הערכים ב-V (מלשון value). לפני שנסכם את הדיון, עם הצגת הממשק המלא, נדון בקצרה בשאלה האם במחלקה Map טיפוסים אלו יכולים להיות גנריים.

אחדות מהפעולות הבסיסיות כבר הזכרנו: אחזור ערך הקשור למפתח, תוספת זוג מפתח-ערך ושינוי הערך הקשור למפתח נתון.

- הפעולה: **void** insert (K key, V value)

  הפעולה insert מאחדת בתוכה למעשה את שתי הפעולות שהזכרנו: את הפעולה המוסיפה זוג מפתח-ערך ואת פעולת העדכון של ערך הקשור למפתח נתון. הפעולה מוסיפה למפה את הזוג: key, value. אם המפתח כבר קיים, הערך שלו יעודכן ל-value.

- הפעולה: V getValue (K key)

  מחפשת במפה את הערך הקשור למפתח key ומחזירה אותו. ההנחה היא שהמפתח קיים במפה. כדי שהמשתמש לא יזמן את הפעולה כאשר המפתח לא קיים הוא זקוק לפעולה הבאה.

- כדי לאפשר למתכנת לדעת האם מפתח מסוים קיים במפה, נוסיף לממשק את הפעולה exists המחזירה 'אמת' אם המפתח המסוים קיים ו-'שקר' אחרת: **boolean** exists (K key)

קל למדי לראות כי הפעולות הנזכרות אינן מאפשרות לבצע *מחוץ* למחלקה Map פעולה כגון מציאת המפתח המינימלי הקיים במפה, או אחזור כל הזוגות שבמפה. כמו כן, ניתן כרגע לשנות ערך הקשור למפתח להיות **null**, אך אי אפשר למחוק את המפתח מהמפה.

? נמקו את הטענה המובעת במשפט הקודם (על חוסר האפשרות לבצע מחוץ למחלקה מפה, את הפעולות הנזכרות).

לכן נוסיף לממשק המחלקה את הפעולות הבאות:

- הפעולה: `K[] getAllKeys()`

מחזירה את המפתחות הקיימים במפה בתור מערך שערכיו ממוינים בסדר עולה. ניתן היה בהחלט להחזיר את אוסף המפתחות כרשימה (List<K>), אך לנוחיותכם כמתכנתי המחלקה, נבחר להשתמש במערך המחזיק את המפתחות כערך החזרה של הפעולה.

- הפעולה: `V remove (K key)`

מוחקת מהמפה את המפתח key ואת הערך המצורף אליו. גם בפעולה זו ההנחה היא שהמפתח קיים במפה, והמשתמש צריך לבדוק זאת בעזרת הפעולה exists לפני הזימון של remove.

? הסבירו כיצד פעולות אלו מאפשרות לבצע את הפעולות החיצוניות שלא התאפשרו קודם לכן.

ב.2. גנריות של מפתח וערך

נדון עתה בשאלה האם הטיפוסים K, V יכולים להיות טיפוסים גנריים בהגדרת המחלקה. מעיון בתיאורי הפעולות ברור כי אין צורך לבצע במחלקה שום פעולה ייחודית 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, exists, remove דורשות השוואת מפתח נתון עם מפתח המאוחסן במפה. פעולת ההשוואה כפי שכבר ציינו, היא פעולה המשתנה מטיפוס עצם אחד למשנהו. בהיקף הידע הנדרש ביחידה זו, מחייב צורך ההשוואה הזה שטיפוס המפתח K לא יהיה גנרי.

להלן סיכום ממשק המחלקה מפה, כאשר המפתח הוא מטיפוס מחרוזת ואילו הערך – גנרי. מכיון שהפעולה getAllKeys() מחזירה מערך ממוין של המפתחות, ומכיון שהמפתחות מחרוזתיים, ברור שהמיון של המערך יהיה בסדר אלפביתי עולה.

ג. ממשק המחלקה Map<V>

Map()	הפעולה בונה מפה ריקה
boolean exists (**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 key קיים במפה הנוכחית
void insert (String key, V value)	הפעולה מוסיפה למפה הנוכחית מפתח key שערכו value. אם המפתח key קיים במפה זוהי פעולת עדכון של ערך המפתח
V remove (String key)	הפעולה מוציאה את המפתח key (יחד עם ערכו) מהמפה הנוכחית, ומחזירה את ערכו. **הנחה**: המפתח key קיים במפה הנוכחית
String[] getAllKeys()	הפעולה מחזירה את אוסף המפתחות שקיימים במפה הנוכחית ממוין בסדר אלפביתי עולה
String toString()	הפעולה מחזירה מחרוזת המתארת את המפה בפורמט הבא: {key1=value1, key2=value2, key3=value3,.. }

ד. ייצוג המחלקה מפה

כדי להשלים את הגדרת המחלקה, עלינו לבחור את התכונות, שבהן ישמרו הנתונים ולממש את הפעולות. המפה מזכירה את מבנה הנתונים מערך הקיים בשפה. מערך הוא אוסף של איברים המאפשר לגשת לאיבר על פי אינדקס (מספר שלם) המציין את מקומו הסידורי באוסף. אם i הוא אינדקס במערך, אז 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 לערכיהם?

התשובה שלילית. האינדקסים בשפת ג'אווה הם תמיד סדרת המספרים השלמים 0, 1, 2...n-1, כאשר n הוא גודל המערך. אנו זקוקים לאחסון מפתחות מטיפוס עצם כלשהו. גם כאשר טיפוס המפתח הוא **int**, ההגבלה במערך על תחום המפתחות מונעת שימוש במערך כייצוג למפה. לכן, ייצוג פנימי של מפה כמערך אינו אפשרי.

ניתן להציע ייצוג מתוחכם יותר המשתמש בשני מערכים.

? הציעו ייצוג למפה המשתמש בשני מערכים.

לייצוג זה יש מגבלה והיא העובדה שגודלו של מערך נקבע בזמן יצירתו. דבר זה מקשה על שימוש במערך לייצוג אוסף דינמי, וזו סיבה טובה שלא לבחור בו.

אם כך עלינו לבחור במבנה נתונים אחר לייצוג הנתונים באוסף. בחירה טבעית היא הרשימה, שבה לא קיימת הגבלת המקום. ניתן אם כן לייצג מפה על ידי זוג רשימות.

? הציעו ייצוג למפה המשתמש בשתי רשימות.

מכיון שאנו מגדירים את אוסף הנתונים השמורים במפה, כאוסף של מפתח-ערך, ניתן לחשוב על הרעיון של ייצוג ה**זוג** מפתח-ערך בעזרת טיפוס נתונים עצמאי. המפה תפעל על טיפוס הנתונים של זוגות אלו.

ה. המשימה – שלב א

1. כתבו את המחלקה Pair המגדירה זוג מפתח-ערך .
2. הציעו ייצוג למפה המשתמש ברעיון הזוגות השמורים באוסף. למעשה עליכם לבחור במבנה נתונים המתאים לשמירת הזוגות שהגדרתם (המחלקה Pair). הסבירו את בחירתכם!
3. ממשו את הפעולות exists ו-remove המוגדרות בממשק המחלקה על פי הייצוג שבחרתם.

ו. פעולה פרטית במחלקה

נשים לב כי כמעט לכל הפעולות המוגדרות בממשק המפה אותו מבנה: לפעולה מועבר מפתח, והפעולה מחפשת חוליה המכילה זוג שבו קיים המפתח (אם הוא קיים ברשימה). ההבדלים בין הפעולות הם במה שהן עושות אחרי שנמצאה חוליה שכזו. מבנה זה משותף לפעולות: getValue, insert, remove, exists. ניתן אם כך לחשוב על האפשרות להגדיר פעולה פרטית המבצעת את חיפוש החוליה. הפעולה הפרטית תקרא search (String key).

על פי הייצוג האחרון של המפה בעזרת רשימה של זוגות, סביר היה להניח כי ערך החזרה של הפעולה הוא הפניה לחוליה במפה המכילה את הזוג שבו קיים המפתח key או **null** אם המפתח המבוקש לא קיים באוסף.

כיוון שערך חזרה זה מועבר אל פעולות הממשק שזימנו את search, ו-search אמורה להקל את מימושן, יש לבדוק אם ערכי החזרה שתוארו לשני המקרים (מפתח קיים, ומפתח לא קיים) אמנם משרתים מטרה זו.

נתמקד בפעולה remove. כאשר המפתח קיים במפה, פעולה זו זקוקה להפניה שונה. הפעולה זקוקה להפניה לחוליה *הקודמת* לחוליה שבה מופיע המפתח key (אם אינכם זוכרים מדוע, חיזרו לעיין בהגדרת הפעולה remove של רשימה...).

כיצד אם כן לממש את הפעולה search? לגבי ערך החזרה כאשר המפתח קיים ברשימה, הדבר יוכרע על ידי שיקולי יעילות. אם לצורך היישום נדרשות הרבה פעולות מחיקה מהמפה, כדאי לממש את search כך שתחזיר הפניה לחוליה הקודמת לחוליה המכילה את המפתח!

את השיקולים לגבי ערך החזרה כאשר המפתח אינו קיים במפה, אנו משאירים לקורא.

? כתבו את פעולת החיפוש הפרטית search המוצאת זוג ברשימה על פי מפתח נתון. הסבירו מהו ערך החזרה המדויק של הפעולה שכתבתם?

ז. המשימה – שלב ב

לאחר שנכתבה הפעולה הפנימית ניתן לחסוך בכתיבת הקוד הדרוש למימוש הפעולות התלויות במציאת מפתח מסוים.

כתבו את כל המחלקה Map, תוך שימוש בפעולה הפנימית search שהגדרתם. שנו את מימושי הפעולות exists ו-remove בהתאמה.

ח. שימוש במחלקה Map

כפי שציינו בתחילת הפרק, יש שימושים רבים באוסף Map: מילון, רשימת ציונים, יומן פגישות ועוד. אנו נציג כאן דוגמה אחת, מעט יותר מורכבת. בתרגילים המצורפים לפרק תתבקשו לטפל בדוגמאות נוספות: מילון וספר טלפונים.

דוגמה: **"בחירות" (או "כוכב נולד")**

כדי לבחור זמר מועדף מתוך קבוצת זמרים, נערכת הצבעה. הזמר הזוכה הוא זה שיזכה במספר הקולות המקסימלי. לשם הפשטות, נניח שקיים מנצח יחיד.

לצורך הפתרון נשתמש במפה שהמפתחות בה יהיו שמות הזמרים והערכים הקשורים למפתחות אלו, יהיו מספרי הקולות בהם זכה כל זמר. כלומר, במפה המפתחות הם מחרוזתיים והערכים כבר אינם גניריים, אלא הם מספרים שלמים:

```
Map<Integer> votes = new Map<Integer>();
```

מכיון שלא מוגדר סדר בין הזמרים ואין לנו סיבה להגביל את מספרם, המפה היא טיפוס מתאים לשמירת ההצבעות. שמו של כל מועמד מזהה אותו באופן ייחודי, כך שהשם משמש כמפתח. למפתח זה נקשר ערך מספרי שהוא מונה הקולות של אותו מועמד.
בתוכנית יהיו שני שלבים. בשלב הראשון תתבצע הצבעה למועמד. בשלב האחר ימצאו המועמד שניצח ומספר הקולות שקיבל, ויודפסו נתונים אלו. שלב זה מניח שהמפה `votes` אינה ריקה.

```
public static void main(String[] args)
{
	Map<Integer> votes = new Map<Integer>();

	IO.hebrew();
	IO.print("ENTER :הקש את שם הזמר/ת הנבחר/ת, לסיום הקש ");
	String singerName = IO.readString();

	while(singerName.compareTo("") != 0)
	{
		if(votes.exists(singerName) == false)
			votes.insert (singerName,1); // הוספת זמר חדש
		else    // עדכון כמות הקולות
			votes.insert(singerName,votes.getValue(singerName)+1);
		IO.print("ENTER :הקש את שם הזמר/ת הנבחר/ת, לסיום הקש ");
		singerName = IO.readString();
	}

	int numVotes;
	String[] allNames = votes.getAllKeys();
	if(allNames.length == 0)
		IO.println(".בתחרות כוכב נולד לא היו כלל הצבעות");

	else
	{
		int max = votes.getValue (allNames[0]);
		String winner = allNames[0];

		for(int i=1; i<allNames.length; i++)
		{
			numVotes = votes.getValue(allNames[i]);
			if (numVotes > max)
			{
				max = numVotes;
				winner = allNames[i];
			}
		}
		IO.println("הכוכב הנולד הוא"+ winner +", זכה ב " +
                                                    max+ " קולות.");
	}
}
```

ט. סיבוכיות

הפעולות insert, getValue, exists, remove כולן משתמשות בפעולה הפנימית search(…) המבצעת חיפוש של המפתח בסדר גודל של O(n), כאשר n הוא מספר המפתחות במפה. לכן, הסיבוך של כל אחת מהפעולות ליניארי בגודל המפה.

י. סיכום

- הגדרנו בפרק זה את טיפוס הנתונים 'מפה' המאפשר לשמור אוסף דינמי של קשרי מפתח-ערך.
- נשים לב כי ממשקי הפעולות אינם מתייחסים למימוש. אמנם ייצגנו את המפה כרשימת זוגות, אך ממשקי הפעולות אינם מתייחסים כלל למושג זוג, למחלקה Pair ולרשימה. נוכל בקלות להחליף את ייצוג המחלקה בלי לשנות את ממשקי הפעולות, ואמנם הזכרנו בפרק מימושים אפשריים אחרים (בעזרת שני מערכים או שתי רשימות).
- המפה כפי שהגדרנו אותה היא טיפוס נתונים מופשט: המימוש אינו חשוף דרך הממשק, ואף אין אפשרות לכתוב פעולות שיחליפו את הפעולות המוצעות, או יגרמו לקלקול האוסף.

ממשק המחלקה Map<V>

הפעולה בונה מפה ריקה	Map()
הפעולה מחזירה 'אמת' אם המפתח key קיים במפה הנוכחית, ו-'שקר' אחרת	**boolean** exists (**String** key)
הפעולה מחזירה את הערך הקשור למפתח key. **הנחה**: המפתח key קיים במפה הנוכחית	**V** getValue (String key)
הפעולה מוסיפה למפה הנוכחית מפתח key שערכו value. אם המפתח key קיים במפה זוהי פעולת עדכון של ערך המפתח	**void** insert (String key, V value)
הפעולה מוציאה את המפתח key (יחד עם ערכו) מהמפה הנוכחית, ומחזירה את ערכו. **הנחה**: המפתח key קיים במפה הנוכחית	**V** remove (String key)
הפעולה מחזירה את אוסף המפתחות שקיימים במפה הנוכחית ממוין בסדר אלפביתי עולה	String[] getAllKeys()
הפעולה מחזירה מחרוזת המתארת את המפה בפורמט הבא: {key1=value1, key2=value2, key3=value3,.. }	String toString()

תרגילים

פעולות נוספות

1. הפעולה **ספור-מפתחות** numOfKeys, מחזירה את מספר המפתחות במפה נתונה.

i. הוסיפו למחלקה Map את הפעולה numOfKeys() כך שסיבוכיות זמן הריצה שלה תהיה O(1).

ii. האם בצעתם שינוי כלשהו בייצוג המחלקה Map בעקבות הדרישה בסעיף i? פרטו והסבירו.

אם ביצעתם שינוי שכזה, האם הוא מחייב שינוי המימוש של פעולות כלשהן או משפיע על יעילותן של פעולות בממשק? פרטו והסבירו.

שימוש במפה

בשאלות הבאות נשתמש בטיפוס הנתונים המופשט 'מפה' לצורך ייצוג אוספים בבעיות קונקרטיות.

פרק 8 דף עבודה 1
ספר טלפונים

מטרות

1. תרגול של טיפול באוספים ספציפיים תוך שימוש בטיפוסי אוסף קיימים (שימוש ב-Map)

המחלקה PhoneBook

ספר טלפונים הוא אוסף, לא מוגבל באורכו, של אנשי קשר, כך שכל איש קשר מזוהה על ידי שם ייחודי. הערך המזהה נקרא **מפתח** (אין שמות שחוזרים על עצמם). לכל איש קשר קיימים בספר: מספר טלפון וכתובת דוא"ל (ניתן להוסיף פרטים נוספים).

לפניכם ממשק המחלקה:

PhoneBook()	הפעולה בונה ספר טלפונים ריק
boolean exists (String name)	הפעולה מחזירה 'אמת' אם איש הקשר קיים בספר הטלפונים ו-'שקר' אחרת
String getPhone (String name)	הפעולה מחזירה את מספר הטלפון השייך לאיש הקשר. **הנחה**: איש הקשר מופיע בספר הטלפונים
String getEMail (String name)	הפעולה מחזירה את כתובת הדוא"ל השייכת לאיש הקשר. **הנחה**: איש הקשר מופיע בספר הטלפונים
void addContact (String name, String phone, String eMail)	הפעולה מוסיפה איש קשר לספר הטלפונים הנוכחי. אם קיים איש קשר באותו שם תתבצע פעולת עדכון של נתוניו האחרים
void deleteContact (String name)	הפעולה מוחקת איש קשר ששמו name מספר הטלפונים
String[] getAllNames()	הפעולה מחזירה אוסף ממוין בסדר אלפביתי עולה של כל השמות של אנשי הקשר המופיעים בספר הטלפונים
String toString()	הפעולה מחזירה מחרוזת המתארת את ספר הטלפונים בפורמט הבא: name1: phone number, eMail address name2: phone number, eMail address :

מה עליכם לעשות?

1. כדי לכתוב יישום ממוחשב עליכם למדל את ספר הטלפונים ולהגדיר לאילו מחלקות תזדקקו, מהן תכונותיהן ומה ממשקן של המחלקות. הגדירו את המידול אליו הגעתם בעזרת איורי UML.
2. כתבו את המחלקות שהגדרתן בסעיף א, תוך תיעודן המלא. ראשית, חשבו על ייצוג מתאים למחלקה.
3. בדקו את המחלקה שכתבתם בעזרת תוכנית הבדיקה TestPhoneBook הנמצאת ב: C:\eclipse\Unit 4\HelpFiles.

פרק 8 דף עבודה 2

מילון Dictionary

מטרות

תרגול של מידול ושימוש ב-UML (בעיית מידול)

תרגול של טיפול באוספים ספציפיים תוך שימוש בטיפוסי אוסף קיימים (Map)

כתיבת טיפוס נתונים חדש

המחלקה Dictionary

ברצוננו לפתח מילון **אנגלי-עברי** ממוחשב המכיל אוסף של מילים באנגלית ותרגומן לעברית. לכל מילה (word) במילון יש כמה תרגומים (translations). המילון יכיל את הפעולות הבאות:

Dictionary()	הפעולה בונה מילון ריק
boolean exists (String word)	הפעולה מחזירה 'אמת' אם המילה קיימת במילון ו-'שקר' אחרת
String[] getTranslations (String word)	הפעולה מחזירה את אוסף התרגומים של מילה. התרגומים מסודרים לפי סדר הוספתם **הנחה**: המילה קיימת במילון
void addWord (String word, String translation)	הפעולה מוסיפה מילה ואת תרגומה למילון. אם המילה קיימת במילון, פעולה זו מוסיפה תרגום נוסף למילה. אם התרגום למילה כבר קיים, לא יתבצע כל שינוי
void removeWord (String word)	הפעולה מוציאה מילה (כולל תרגומיה) מתוך המילון. **הנחה**: המילה קיימת במילון
String[] getAllWords()	הפעולה מחזירה את אוסף כל המילים המופיעות במילון, ממוין בסדר אלפביתי עולה של המילים
String toString()	הפעולה מחזירה מחרוזת המתארת את המילון בפורמט הבא: word1: תרגום-1, תרגום-2,... word2: תרגום-1, תרגום-2,... המילים מופיעות ממוינות בסדר אלפביתי עולה, והתרגומים לכל מילה מופיעים לפי סדר הוספתם למילון (הופעת העברית והאנגלית על פי כיווני הכתיבה המקובלים – נעשית באופן אוטומטי!!)

מה עליכם לעשות?

חלק א

1. על מנת לכתוב את המחלקה Dictionary עליכם לבחור ייצוג מתאים.

הדרכה:

ייצוג מתאים צריך לקחת בחשבון את הנתונים שלכם ואת הפעולות המוגדרות על נתונים אלו.

i. אוסף נתונים אחד הוא אוסף המילים במילון עם תרגומיהן, והפעולות המופיעות בממשק שלעיל. איזה טיפוס נתונים עולה בדעתכם כאשר אתם מסכמים את המידע הזה?

ii. אוסף נתונים נוסף שעליכם להתייחס אליו הוא אוסף התרגומים עבור כל מילה. כל תרגום הוא מחרוזת ולכן הוא מוגדר היטב ואין צורך להגדיר עבורו טיפוס חדש. אוסף המחרוזות, הוא אוסף שהסדר המחייב בו הוא סדר הכנסת התרגומים למילון! הפעולות הנדרשות על האוסף פרט לבנייתו הן: הוספה, חיפוש והחזרה. האם עולה בדעתכם טיפוס נתונים המאפשר בדיוק את הדרישות הללו, או שמא תגדירו טיפוס חדש לצרכים אלו?

עכשיו ביכולתכם לכתוב את המחלקה כולה ולתעד אותה במלואה.

<u>הערה חשובה</u>: לצורך שאלה זו, כל הפעילות במילון ומחוצה לו אינה מבחינה בין אותיות גדולות וקטנות (כלומר winter, Winter הן אותה מילה).

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

השלבים:

i) צרו מילון ובו מילים ותרגומן כרצונכם (לפחות עשר מילים).

ii) קלטו משפט באנגלית ותרגמו אותו לעברית בעזרת המילון. במידה ויש למילה יותר מתרגום אחד, הדפיסו את הראשון. במידה ואין למילה תרגום, הדפיסו '?'.

הנחיה: לצורך הקלט אתם יכולים:

i. להיעזר במחלקה StringTokenizer שתאפשר לכם לקלוט משפט שלם ולפרקו למילים (ראו בJavaAPI).

ii. או, לבצע קלט של מילים בודדות.

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

אם במילון האנגלי-עברי הופיעו: **indenture**: אמנה, חוזה, הסכם

אזי במילון העברי-אנגלי יופיעו:

אמנה: indenture

הסכם : indenture

חוזה : indenture

הוסיפו לתוכנית שכתבת בסעיף 2 פעולה המקבלת מילון אנגלי-עברי ומייצרת ממנו מילון עברי-אנגלי מתאים. הפעולה תדפיס את תכולת המילונים.

חלק ב

1. הוסיפו למחלקה Dictionary את הפעולה:

public List<String> sameTranslation (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 אך אין להן תרגום במילון.

3. כתבו פעולה המקבלת מילון ומדפיסה את כל המילים המופיעות בו ותרגומיהן. המילים יופיעו בסדר אלפביתי עולה וגם אוסף התרגומים של כל מילה יופיע בסדר אלפביתי עולה.
שימו לב: שהדפסת התרגומים העבריים, מימין לשמאל, היא תוצאה טבעית של השימוש במחלקת IO (המשתמשת בשירותי System.out) ואין לכם צורך לטפל בנושא סדר ההדפסה הנכון.
רמז: חשבו היכן תוגדר הפעולה?

4. מהי סיבוכיות זמן הריצה של הפעולה שהגדרתם בסעיף 2?