

תרגול מס' 4: המתרגם

שימוש במחלקות קיימות
מחרוזות, קבצים, וקבלת קלט מהמשתמש

המתרגם

Language Translation



• משימה:

- תכנית המתרגמת קטעי טקסט לשפה אחרת
- הקלט: קובץ המכיל את קטעי הטקסט וכן את השפה אליה רוצים לתרגם

• שאלות:

- האם כבר יש שירות תרגום שאנחנו יכולים להשתמש בו?
- כיצד קוראים מקבצים?
- מה הפורמט של הקלט?
- נצטרך להחליט

פתרון צעד אחר צעד

- כצעד ראשון נפתור בעיה הרבה יותר פשוטה
- תכנית שמתרגמת את המילה "Hello" מאנגלית לצרפתית
 - יש: שימוש בשירות תרגום
 - אין: קלט, טקסט, עבודה עם קבצים, פורמט



API – Application Programming Interface

- ממשק המאפשר לאפליקציה לתקשר עם תוכנה אחרת
- בג'אווה קיימים כלים רבים הזמינים ברשת בקוד פתוח
- בתרגול זה נשתמש ב-API תרגום כללי Translate
- ברשת קיימים כלים שונים של Google, Microsoft

ועוד

שלב א'

```
public class TranslatorEngine1 {  
  
    public static void main(String[] args) {  
  
        String TranslatedText = Translate.execute("Hello", "English",  
        "French");  
  
        System.out.println(TranslatedText);  
    }  
}
```

מתודה סטטית, שנקראת
מן המחלקה (Translate)

שלב ב' - אינטראקציה עם המשתמש

- קלט מהמשתמש יינתן בשורת הפקודה
 - פרמטר ראשון: המילה לתרגום
 - פרמטר שני: שפת המקור
 - פרמטר שלישי: שפת היעד

```
public class TranslatorEngine2 {  
  
    public static void main(String[] args) {  
        String TranslatedText =  
            Translate.execute(args[0], args[1], args[2]);  
        System.out.println(TranslatedText);  
    }  
}
```

קלט אינטרקטיבי

- מה עם נרצה להעביר קלט במהלך הריצת התוכנית?

```
>java TranslatorEngine  
Enter your input:  
Hello English French  
Your translation is: Bonjour
```

המחלקה Scanner

- נשתמש בסורק טקסט פשוט לקרוא קלט מהשתמש
- "שובר" את הקלט לרכיביו השונים (מילה, מספר וכדומה)
- בעת יצירה מקבל כפרמטר מהיכן לקרוא את הקלט
- בתור התחלה, נקרא מה-console (הקלט הסטנדרטי של התכנית) - `system.in`

```
Scanner scanner = new Scanner(System.in);  
int anInt = scanner.nextInt();  
float aFloat = scanner.nextFloat();  
String aString = scanner.next();  
String aLine = scanner.nextLine();
```

מתודות מופע (לא סטטיות),
אשר נקראות ממופע
(אובייקט) של המחלקה

<https://docs.oracle.com/javase/7/docs/api/index.html?java/util/Scanner.html>

דוגמא

קרא מ- standard input

```
Scanner s = new Scanner(System.in);  
System.out.println("enter line:");  
while (s.hasNext())  
    System.out.println(s.next());  
  
s.close();
```

קרא את ה- Token הבא

שלב ג' – שימוש בסיסי ב-Scanner

נבחר את פורמט הקלט: `<word> <source-lang> <target-lang>`
לדוגמא,

הקלט: `hello English French` ■

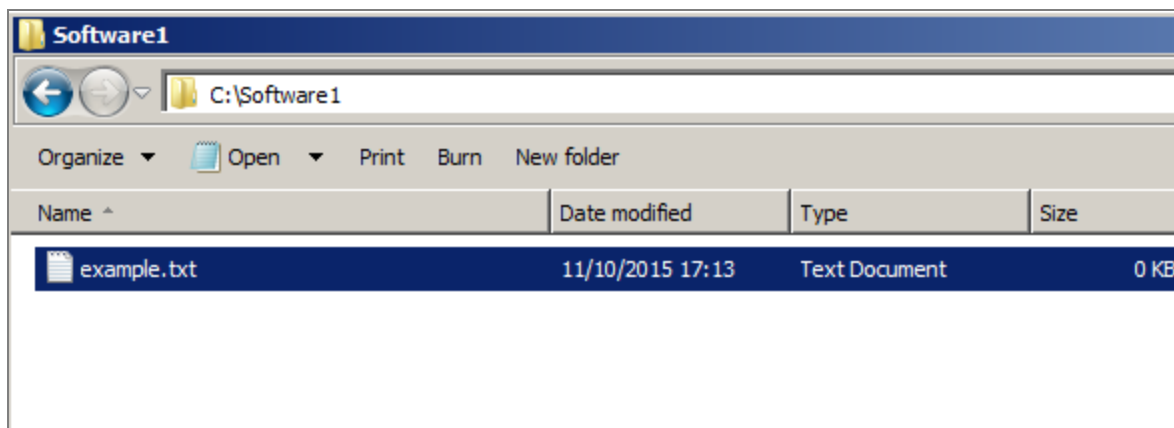
הפלט: `bonjour` ■

```
public class TranslatorEngine3 {  
    public static void main(String[] args) {  
        Scanner s = new Scanner(System.in);  
        String[] fragments = s.nextLine().split(" ");  
        String TranslatedText =  
            Translate.execute(fragments[0], fragments[1], fragments[2]);  
        System.out.println(TranslatedText);  
        s.close();  
    }  
}
```

קבצים

- במקום לקרוא את שורת הקלט מהמשתמש נקרא אותה מקובץ
- קובץ מיוצג ע"י המחלקה `java.io.File`
- נאתחל את האובייקט עם המסלול (`path`) לקובץ

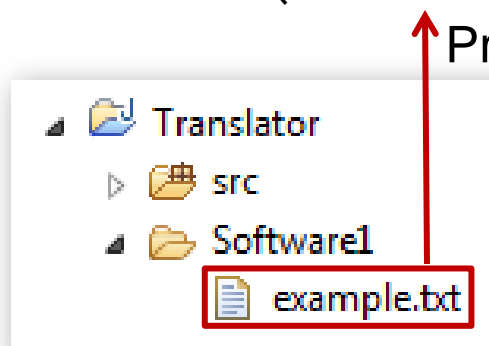
```
String filePath = "C:\\Software1\\example.txt";  
File exampleFile = new File(filePath);
```



מסלול (Path) לקובץ

• מסלול יחסי – Relative path

```
new File("Software1\\example.txt")
```



- ב- eclipse המיקום ה"נוכחי" במהלך ריצה הוא ה-Project root
- דרך טובה לבדוק את המיקום הנוכחי של הפרוייקט הוא לייצר קובץ במיקום היחסי, ואז לבדוק היכן הוא נוצר.

• מסלול מלא – Absolute path

```
new File("C:\\Software1\\example.txt")
```

- יתרון – ניתן להריץ את התוכנית מכל מקום והיא תמיד תוכל למצוא את הקובץ.
- חסרון- הרבה פעמים הקובץ ממוקם יחסית לתוכנית, ואז אם היא מועתקת, גם הקובץ מועתק והקוד לא ירוץ.
- טעות נפוצה בתרגילי הבית: הגשת קוד שמכיל מסלול מלא לקובץ:

```
new File("C:\\users\\lenadank\\software1\\ex4\\my_file.txt")
```

מסלול (Path) לקובץ

- כיצד נדאג שהתכנית תתאים לכל מערכת הפעלה? (Windows, Linux...)
- פתרון א':

```
new File("Software1/example.txt")
```

- פתרון ב':
- ```
new File("Software1" + File.separator + "example.txt")
```

- פתרון ג': נקבל את המסלול כקלט מהמשתמש.

# שלב ד' – Scanner וקריאה מקובץ

המסלול לקובץ יהיה (שדה) קבוע של המחלקה

```
public class TranslatorEngine4 {

 private static final String FILE_NAME = "Software1" + File.separator +
 "example.txt";

 public static void main(String[] args) throws Exception {

 Scanner s = new Scanner(new File(FILE_NAME));
 String[] fragments = s.nextLine().split(" ");
 String TranslatedText = Translate.execute(fragments[0], fragments[1],
 fragments[2]);
 System.out.println(TranslatedText);

 s.close();
 }
}
```

## זריקת חריגים – הצהרת throws

- בעת חיבור ה-Scanner לקובץ עלולה להיזרק שגיאה (חריג, Exception) מוג FileNotFoundException
- במקרה שהקובץ ממנו ניסינו לקרוא לא קיים, ריצת המתודה תעצור
- החריג מכיל הסבר על מקור השגיאה

### • שתי אפשרויות להתמודדות: זרקו הלאה או טפלו

- נדבר על טיפול בחריגים ועוד בהמשך הקורס.
- כרגע נטפל בחריג באופן הבא:
- נצהיר על זריקת חריג בחתימת המתודה באמצעות המילה השמורה throws.
- החריג עליו נצהיר יהיה חריג מטיפוס Exception, שהוא החריג הכללי ביותר שיש. כלומר, המתודה שלנו מצהירה שהיא יכולה לזרוק חריג, ומי שקורא לה צריך להיות מודע לזה ולטפל בזה במידת הצורך.

# שלב ה' – קלטים מרובים

- מספר שורות קלט מקובץ
- נקרא מספר קלטים עד לסוף הקובץ, שימוש ב- `hasNextLine`

```
public class TranslatorEngine5 {
 private static final String FILE_NAME = "Software1" + File.separator
 + "example5.txt";

 public static void main(String[] args) throws Exception {
 Scanner s = new Scanner(new File(FILE_NAME));
 while (s.hasNextLine()) {
 String[] fragments = s.nextLine().split(" ");
 System.out.println(Translate.execute(fragments[0], fragments[1],
 fragments[2]));
 }
 s.close();
 }
}
```



# נרחיב את המחלקה שלנו לטיפול בפיסקאות

- נרצה לקרוא פיסקה, להמיר לשורה אחת, ולתרגם
- צריך להגדיר את פורמט הקלט מחדש.

נגדיר:

`<source-lang>#<target-lang>#<paragraph>`

• למשל:

English#French#Hello world!

This program works.

Bye.

# שימוש ב delimiter ב scanner

```
String input = "1 fish 2 fish red fish blue fish ";
Scanner s = new Scanner(input).useDelimiter(" fish ");
while (s.hasNext())
 System.out.println(s.next());
s.close();
```

Output:

```
1
2
red
blue
```

# שלב ו' – תרגום פסקה

```

public class TranslatorEngine6 {
 private static final String FILE_NAME = "Software1" + File.separator + "example6.txt";

 public static void main(String[] args) throws Exception {
 Scanner s = new Scanner(new File(FILE_NAME));
 s.useDelimiter("#");
 String srcLanguage = s.next();
 String destLanguage = s.next();
 s.skip("#");
 String text = "";
 while (s.hasNextLine()) {
 text += s.nextLine() + ' ';
 }
 System.out.println(Translate.execute(text, srcLanguage, destLanguage));
 s.close();
 }
}

```

English#French#Hello world!  
 This program works.  
 Bye.

# לאן עכשיו?

- טיפול בשגיאות
  - פורמט לא תקין, כשלון בזיהוי השפות או בתרגום
  - ניתן לבדוק בקוד או להגדיר בחוזה
- הרחבת התכנית
  - תרגום מספר קבצים
  - מספר פסקאות בקובץ יחיד
  - זיהוי אוטומטי של שפת הקלט
  - ...

# המחלקה StringBuilder

- מייצגת מחרוזות ניתנת לשנוי (mutable)
- מאפשרת לבצע שינוי במחרוזת קיימת מבלי ליצור עצמים חדשים
- שירותים חשובים: append ו- insert

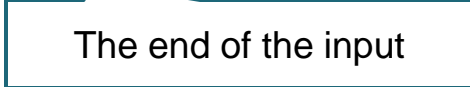
```
StringBuilder sb = new StringBuilder("abc");
sb.append("d");
```

למה לא לשרשר מחרוזות באמצעות חיבור מחרוזות?

# קריאת קובץ טקסט לתוך מחרוזת

```
//Option #1
```

```
Scanner scanner = new Scanner(new File("example.txt"));
String result = scanner.useDelimiter("\\Z").next();
scanner.close();
```



The end of the input

```
//option #2
```

```
BufferedReader bufferedReader = new BufferedReader(
 new FileReader("example.txt"));

String line;
StringBuilder stringBuilder = new StringBuilder();
while ((line = bufferedReader.readLine()) != null) {
 stringBuilder.append(line + "\n");
}
String result = stringBuilder.toString();
bufferedReader.close();
```

# כתיבת מחרוזת לקובץ טקסט

```
public static void main(String args[]) throws IOException {
 String content = "Hello! Java-Buddy :)";
 File newTextFile = new File("output.txt");
 FileWriter fileWriter = new FileWriter(newTextFile);
 fileWriter.write(content);
 fileWriter.close();
}
```