

תרגול מס' 4: המתרגם

שימוש במחלקות קיימות
מחרוזות, קבצים, וקבלת קלט מהמשתמש

המתרגם

משימה:

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



תכנות מתקדם בשפת Java
אוניברסיטת תל אביב

שאלות

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

■ 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
- במציאות, קיימים ברשת כלים שונים של Google,

Microsoft ועוד

שלב א'

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

אינטראקציה עם המשתמש

- נתחיל להתקדם עקב בצד אגודל אל היעד שלנו
- קלט מהמשתמש יינתן בשורת הפקודה
 - פרמטר ראשון: המילה לתרגום
 - פרמטר שני: שפת המקור
 - פרמטר שלישי: שפת היעד

שלב ב'

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


קריאת קלט

■ נקרא קלט מהמשתמש (console)

■ עדיין מילה אחת

■ אין שימוש בקבצים

■ נשתמש במחלקה Scanner

■ מה הפורמט של הקלט?

המחלקה Scanner

- סורק טקסט פשוט
- "שובר" את הקלט לרכיביו השונים (מילה, מספר וכדומה)
- בעת יצירה מקבל כפרמטר מהיכן לקרוא את הקלט

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

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

פורמט הקלט

מהו הפרוטוקול המשותף שחולקים האפליקציה והמשתמש לצורך התקשורת ביניהם

■ איזה מידע דרוש

■ כיצד הוא מקודד (מספר, מחרוזת, ...)

■ מה סדר הפרמטרים

■ נבחר: `<word> <source-lang> <target-lang>`

■ לדוגמא,

■ הקלט: hello English French

■ הפלט: bonjour

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'

```
public class TranslatorEngine3 {  
  
    public static void main(String[] args) throws Exception {  
  
        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();  
    }  
}
```

מפצל את המחרוזת
לפי רווחים

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

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

מסלול (Path) לקובץ

מסלול יחסי – Relative path ■

`new File("example.txt")` ■

ב- eclipse המיקום הנוכחי במהלך ריצה הוא ה-
Project root ■

מסלול מלא – Absolute path ■

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

תלות בסביבה

■ ג'אווה היא שפת תכנות חוצת סביבות, אבל מערכת הקבצים תלויה בסביבה!

■ למשל, המפריד בסביבת Unix הוא / (slash)

`/usr/local/software1/example.txt`

■ ובסביבת Windows הוא \ (backslash)

`C:\Software1\example.txt`

אך היא תומכת גם ב- / כמפריד.

■ **פתרון א':** נשתמש תמיד ב- /

■ בעיה – מה לגבי סביבות אחרות שאולי לא תומכות בו?

תלות בסביבה - המשך

■ פתרון ב': שימוש ב- `File.separator` המוגדר בהתאם לסביבה

■ מתאים בעיקר לקבצים במיקום יחסי לפרוייקט

■ לדוגמא:

```
new File("Software1" + File.separator +  
"example.txt")
```

■ פתרון ג': נקבל את המסלול לקובץ כקלט מהמשתמש

שלב ד'

```
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();  
    }  
}
```

קלטים מרובים

■ מספר שורות קלט מקובץ

■ נקרא מספר קלטים עד לסוף הקובץ

■ שימוש ב**nextLine** ו**hasNextLine**

שלב ה'

```
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();  
    }  
}
```

פיסקה

■ פיסקה ולא רק מילה אחת

■ מה יהיה הפורמט החדש?

■ נבחר:

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

Scanner – Set Delimiter Example

```
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();
```

קריאת פיסקה מהקובץ

- פיסקה יכולה להכיל מספר שורות (נוותר בינתיים על קלטים מרובים).
- נרצה לקרוא ולצרף אותן למחרוזת אחת.
- ניתן להשתמש באופרטור +, שיוצר בכל פעם מחרוזת חדשה
- אנו נשתמש במחלקה `StringBuilder`

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");
```



```
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("#");
        StringBuilder text = new StringBuilder();
        while (s.hasNextLine()) {
            text.append(s.nextLine());
            text.append(' ');
        }
        System.out.println(Translate.execute(text.toString(), srcLanguage,
            destLanguage));
        s.close();
    }
}
```

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

...