תווים ומחרוזות

ראינו שאפשר לשמור במחשב נתונים מספריים ע"י משתנים. מצד שני, מכיוון שבני האדם עובדים דווקא יותר עם קלט טקסטואלי (מילים, משפטים, אותיות וכו') היינו רוצים למצוא דרך לשמור גם קלטים כאלה בזכרון המחשב.

1. תווים

תווים הם כל הסימנים שהמחשב מציג ושיש (למשל) על המקלדת – אותיות, מספרים, סימנים כמו %, *, $, רווח, enter ועוד. בג'אווה ניתן לשמור תו בודד בתא זכרון מסוג char. למשל:

```
char c = 'a';
```

שימו לב שכדי לציין שמדובר בתו a ולא במשתנה שקוראים לו ככה, מסמנים גרש מסביב לתו. תא זכרון מסוג char יודע לשמור תו **אחד** בלבד! למשל, השורה הבאה לא תעבור קומפילציה:

```
char c = 'abc';
```

כידוע, המחשב יודע להתמודד רק עם קלט מספרי, ולכן גם את התווים הוא "מתרגם" לערך מספרי. כל תו מקבל מספר יחודי משלו (מתוך טבלת ערכים הנקראת ASCII), ולפיכך, לפעמים נוח להתייחס לתווים לפי סדר הופעתם בטבלה זו. למשל, כל אותיות האלפבית האנגלי מסודרות במספרים עוקבים אחת אחרי השניה, כל האותיות הגדולות מ-A עד Z ברצף, וכל האותיות הקטנות מ-a עד z ברצף. לכן, למשל, נוכל לבצע את השורות הבאות:

```
char c1 = 'a', c2 = 'b';
if(c1 < c2)
     System.out.println("OK");
```

והפלט יהיה "OK", מאחר ושתי האותיות באמת מסודרות אחת אחרי השניה גם באלפבית, וגם במספור שלהם במחשב.

הקוד הבא מדפיס על המסך את כל אותיות האלפבית הקטנות:

```
for(char c = 'a'; c <= 'z'; c++)
     System.out.print(c + " ");
```

2. מחרוזות

העבודה עם תווים היא לרוב לא מספיקה עבור רוב האפליקציות – בדר"כ אנחנו משתמשים ביותר מתו אחד כדי לייצג קלט (למשל, שם של בן אדם מורכב מכמה תווים ברצף). אוסף תווים נקרא **מחרוזת (String)** וכל שפה מייצגת מחרוזות בדרך שונה.

שפת ג'אווה מגדירה מחרוזות באמצעות אובייקט סטנדרטי שנקרא String. אובייקט זה קיים בחבילה java.lang ומוכר אוטומטית בכל תוכנית ג'אווה. למשל, נוכל להגדיר מחרוזת כך:

String str = "hello world!"

המשתנה str **מצביע** על אובייקט מסוג String שמכיל את המחרוזת "Hello World!" (כולל הרווח).

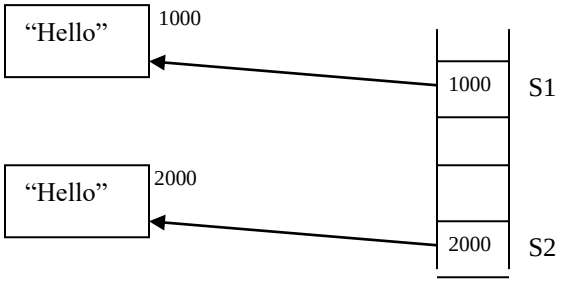
3. מחרוזות כאובייקטים

צורת הרישום של מחרוזות בג'אווה מבלבלת, וגורמת למחשבה (מוטעית) שמחרוזת היא משתנה פרימיטיבי (כלומר פשוט, מכיל תא זכרון אחד) כמו int, double, long וכו', אולם אסור לטעות – מחרוזת (למרות דרך ההגדרה שלה בקוד) היא עדיין **אובייקט** ולכן גם חלים עליה הכללים הרגילים של אובייקט. אמנם לא כתבנו את המילה new ביצירת המחרוזת, אולם בפועל, עדיין נוצר אובייקט בזכרון הערימה, שהמשתנה str מצביע עליו.

כיוון שמחרוזות הן אובייקטים, אין מניעה ליצור אותן בצורה הסטנדרטית של אובייקטים, כך למשל:

```
String s1 = new String("Hello");
String s2 = new String("Hello");
```

ובזכרון יראו האובייקטים כך:



כלומר, במחסנית ישנם שני תאי זכרון, s1, s2 שמכילים כתובות של אובייקטים מסוג String מהערימה. שימו לב, למרות שבשתי האובייקטים מאוחסן אותו ערך, עדיין מדובר בשני אובייקטים **שונים**! (יש להם כתובות שונות).

מה יקרה אם ננסה להשוות עכשיו בין המחרוזות?

```
if(s1 == s2)
```

כיוון ששני האובייקטים מכילים את המחרוזת "Hello" היינו מצפים שהתשובה על התנאי תהיה true, אולם בפועל ההשוואה מתבצעת על ערכי s1 ו-s2, כלומר מתבצעת השוואה של כתובות. כיוון שב-s1 יש 1000, וב-s2 יש 2000, הם לא שווים, והתנאי יחזיר false. העניין הזה הוא מכשלה אמיתית למתכנתים מתחילים מאחר ומצד אחד ניתן להתייחס למחרוזת כאל משתנה "כמו" פרימיטיבי (בדרך ההגדרה שלה) אולם מצד שני בפועל המחרוזות הן אובייקטים לכל דבר. חשוב לזכור, אם כן, שיצירת מחרוזת יוצרת **תמיד** אובייקט בערימה.

אז איך בכל זאת נשווה בין ערכי המחרוזות ולא בין הכתובות? נשתמש בשיטה equals שמוגדרת במחלקה String ושמשווה ערכים ולא כתובות, כך:

```
if(s1.equals(s2))
```

ועכשיו נקבל באמת true כיוון שהשיטה משווה בין הערכים הפנימיים של המחרוזות ולא בין הכתובות של האובייקטים.

האם התשובה תשתנה אם נכתוב s2.equals(s1)?

4. מחרוזות כאובייקטים בלתי ניתנים לשינוי

בג'אווה המחרוזות הן אובייקטים שאינם ניתנים לשינוי (immutable) – כלומר, לאחר יצירת מחרוזת והצבת ערך בתוכה, לא ניתן לשנות את הערך הזה. חשוב לזכור עובדה זו – כל פעולה שמבצעת שינויים על מחרוזות למעשה יוצרת אובייקט חדש של מחרוזת עם השינויים שהתבצעו, השינויים לא מתבצעים על המחרוזת המקורית! למשל, מוכרת לנו כבר פעולת שרשור המחרוזות +:

```
String str = "Hello";

str = str + " World";
```

גם פה יש "טעות אופטית" – המשתנה str הצביע בהתחלה על אובייקט שהכיל את המחרוזת Hello. בשורה השניה מתבצע שרשור, כלומר "מדביקים" ל-"Hello" את המחרוזת "World ". אולם למרות צורת ההצבה שגורמת לנו לחשוב כאילו נעשתה הוספה אמיתית של שתי המחרוזות, מה

שקורה בפועל הוא שנוצר אובייקט חדש לחלוטין (אפילו שלא השתמשנו ב-new) שמכיל את המחרוזת "Hello World", וכתובתו של אובייקט חדש זה מוצבת לתוך str.

מה גורלו של האובייקט הישן, זה שהכיל את המחרוזת "Hello"? כרגיל כמו בכל אובייקט בג'אווה, ברגע שהאובייקט מאבד את ההצבעה שלו, הוא מתפוגג בזכרון.

5. שיטות נוספות של מחלקת String

המחלקה String מגדירה שיטות מועילות לעבודה עם מחרוזות. נביא כמה מהן כאן.

- לפעמים נרצה לדעת כמה תווים מכילה המחרוזת (כלומר, מה אורכה של המחרוזת). ניתן לקבל את המידע הזה באמצעות השיטה length(), כך:

```
String s = "Hello World";

System.out.println(s.length());
```

ועל המסך נקבל את המספר 11 (שימו לב שהרווח גם נחשב כחלק מתווי המחרוזת).

- אם נרצה לדעת איזה תו מאוחסן במיקום מסויים במחרוזת, נוכל להשתמש בשיטה charAt שמקבלת כפרמטר מספר שמציין את המיקום המבוקש. שימו לב שמיקומו של התו הראשון במחרוזת הוא 0 ולא 1. למשל, עבור המחרוזת שהוגדרה קודם, נוכל לבצע את השורות הבאות:

```
System.out.println(s.charAt(0)); // 'H'

System.out.println(s.charAt(6)); // 'W'

char c = s.charAt(10);   // c = 'd'
```

דוגמא: ברצונינו לכתוב שיטה שמקבלת כפרמטר מחרוזת, ומחזירה כמה פעמים מופיע התו '?' במחרוזת:

```
public int countQuestionMark(String str)
{
     int count = 0;
     for(int i=0; i<str.length(); i++)
          if(str.charAt(i) == '?')
               count++;
     return count;
}
```

מה יקרה אם נעביר לשיטה charAt מספר שהוא מחוץ לגבולות המחרוזת? למשל:

```
String s = "abcd";

char c = s.charAt(10);
```

במקרה כזה נקבל **שגיאת ריצה** שאומרת שניסינו לגשת לתא לא חוקי במחרוזת. שאלה למחשבה – למה הקומפיילר לא מזהה את הבעיה בעצמו, אלא נותן לתוכנית לרוץ ולהיתקל בבעיה בזמן ריצה?

מה יקרה אם ננסה לשנות את המחרוזת ע"י הצבת תווים בתוכה? למשל:

```
String s = "abcd";

s.charAt(0) = "F";
```

השורה השניה בכלל לא תעבור קומפילציה. וזה גם הגיוני לפי מה שהגדרנו קודם – מחרוזות שנוצרו אינן ניתנות לשינוי!

- ניתן לקבל תת מחרוזת ממחרוזת נתונה ע"י שימוש בשיטה substring. שיטה זו שמוגדרת גם היא במחלקה String מקבלת כפרמטר מספר שמציין את המיקום ממנו רוצים לחתוך את המחרוזת המקורית, ומחזירה מחרוזת **חדשה** שמכילה את התווים מנקודת החיתוך ועד סוף המחרוזת המקורית. למשל:

```
String s1 = "abcde";

String s2 = s1.substring(2);

System.out.println(s2);
```

ועל המסך נקבל "cde" (שוב, זיכרו שמתחילים לספור את התווים מתו מספר 0, ולכן התו 'c' נמצא במיקום 2).

שיטות נוספות ניתן למצוא ב-API של ג'אווה, וגם בספר באנגלית בעמוד 147. כדאי לנסות לשחק עם המחרוזות כדי לוודא שדרך הפעולה של ג'אווה עליהן מובנת לחלוטין.