תוכנה 1 שיעור 2

כמה אותיות מרכיבות את הספר "תום סוייר"?
(ועוד סוגיות שברומו של עולם)

```
import java.io.*;
import java.net.*;
import java.util.*;

public class Lecture2Program1 {

  public static
  void main(String[] args) throws IOException {
    URL         url     = new URL("http://www."
       + "gutenberg.org/cache/epub/74/pg74.txt");
    InputStream stream  = url.openStream();
    Scanner     scanner = new Scanner(stream);
    scanner.useDelimiter("\\A");
    String      book    = scanner.next();

    System.out.println("The text contains "
            + book.length() + " characters");
  }
}
```

ספריה לעבודה עם קבצים

ספריה לתקשורת

ספריה של כל מיני

```
import java.io.*;
import java.net.*;
import java.util.*;

public class Lecture2Program1 {

  public static
  void main(String[] args) throws IOException {
    URL         url     = new URL("http://www."
       + "gutenberg.org/cache/epub/74/pg74.txt");
    InputStream stream  = url.openStream();
    Scanner     scanner = new Scanner(stream);
    scanner.useDelimiter("\\A");
    String      book    = scanner.next();

    System.out.println("The text contains "
            + book.length() + " characters");
  }
}
```

הצהרה שיתכן שהפונקציה תעוף בגלל בעיות תקשורת או בעיות עם קבצים

```
import java.io.*;
import java.net.*;
import java.util.*;

public class Lecture2Program1 {

  public static
  void main(String[] args) throws IOException {
    URL          url     = new URL("http://www."
       + "gutenberg.org/cache/epub/74/pg74.txt");
    InputStream stream  = url.openStream();
    Scanner      scanner = new Scanner(stream);
    scanner.useDelimiter("\\A");
    String       book    = scanner.next();

    System.out.println("The text contains "
            + book.length() + " characters");
  }
}
```

עצם שמייצג כתובת אינטרנט, עם אתחול

```
import java.io.*;
import java.net.*;
import java.util.*;

public class Lecture2Program1 {

  public static
  void main(String[] args) throws IOException {
    URL         url     = new URL("http://www."
       + "gutenberg.org/cache/epub/74/pg74.txt");
    InputStream stream  = url.openStream();
    Scanner     scanner = new Scanner(stream);
    scanner.useDelimiter("\\A");
    String      book    = scanner.next();

    System.out.println("The text contains "
            + book.length() + " characters");
  }
}
```

"חיבור" של מחרוזות משרשר אותן (המחרוזת גלשה מהשקף...)

```
import java.io.*;
import java.net.*;
import java.util.*;

public class Lecture2Program1 {

  public static
  void main(String[] args) throws IOException {
    URL         url     = new URL("http://www."
     +  "gutenberg.org/cache/epub/74/pg74.txt");
    InputStream stream  = url.openStream();
    Scanner     scanner = new Scanner(stream);
    scanner.useDelimiter("\\A");
    String      book    = scanner.next();

    System.out.println("The text contains "
            + book.length() + " characters");
  }
}
```

טיפוס שמייצג סדרת קלט של בתים (למשל מקובץ או משרת אינטרנט)

```
import java.io.*;
import java.net.*;
import java.util.*;

public class Lecture2Program1 {

  public static
  void main(String[] args) throws IOException {
    URL          url     = new URL("http://www."
        + "gutenberg.org/cache/epub/74/pg74.txt");
    InputStream  stream  = url.openStream();
    Scanner      scanner = new Scanner(stream);
    scanner.useDelimiter("\\A");
    String       book    = scanner.next();

    System.out.println("The text contains "
            + book.length() + " characters");
  }
}
```

שירות של העצם url
שמחזיר את סדרת הבתים
שהשרת ישלח

```
import java.io.*;
import java.net.*;
import java.util.*;

public class Lecture2Program1 {

  public static
  void main(String[] args) throws IOException {
    URL         url     = new URL("http://www."
       + "gutenberg.org/cache/epub/74/pg74.txt");
    InputStream stream  = url.openStream();
    Scanner     scanner = new Scanner(stream);
    scanner.useDelimiter("\\A");
    String      book    = scanner.next();

    System.out.println("The text contains "
            + book.length() + " characters");
  }
}
```

עצם שמייצג אלגוריתם שמפרק סדרת אותיות לחלקים

```
import java.io.*;
import java.net.*;
import java.util.*;

public class Lecture2Program1 {

  public static
  void main(String[] args) throws IOException {
    URL          url     = new URL("http://www."
       + "gutenberg.org/cache/epub/74/pg74.txt");
    InputStream stream  = url.openStream();
    Scanner      scanner = new Scanner(stream);
    scanner.useDelimiter("\\A");
    String       book    = scanner.next();

    System.out.println("The text contains "
            + book.length() + " characters");
  }
}
```

```
import java.io.*;
import java.net.*;
import java.util.*;

public class Lecture2Program1 {

  public static
  void main(String[] args) throws IOException {
    URL          url     = new URL("http://www."
       + "gutenberg.org/cache/epub/74/pg74.txt");
    InputStream stream  = url.openStream();
    Scanner      scanner = new Scanner(stream);
    scanner.useDelimiter("\\A");
    String       book    = scanner.next();

    System.out.println("The text contains "
            + book.length() + " characters");
  }
}
```

שירות של scanner
שאומר לו מה מפריד
בין חלקים של המחרוזת

המפריד \A
הוא תחילת
המחרוזת

שירות של scanner שמחזיר את החלק הבא (פה הכל)

```
import java.io.*;
import java.net.*;
import java.util.*;

public class Lecture2Program1 {

  public static
  void main(String[] args) throws IOException {
    URL         url     = new URL("http://www."
       + "gutenberg.org/cache/epub/74/pg74.txt");
    InputStream stream  = url.openStream();
    Scanner     scanner = new Scanner(stream);
    scanner.useDelimiter("\\A");
    String      book    = scanner.next();

    System.out.println("The text contains "
            + book.length() + " characters");
  }
}
```

אם אחד הארגומנטים של + הוא מחרוזת, גם השני חייב להיות, אז השפה ממירה את השלם למחרוזת

```
import java.io.*;
import java.net.*;
import java.util.*;

public class Lecture2Program1 {

  public static
  void main(String[] args) throws IOException {
    URL          url     = new URL("http://www."
       + "gutenberg.org/cache/epub/74/pg74.txt");
    InputStream stream  = url.openStream();
    Scanner     scanner = new Scanner(stream);
    scanner.useDelimiter("\\A");
    String      book    = scanner.next();

    System.out.println("The text contains "
          + book.length() + " characters");
  }
}
```

כמה אותיות מרכיבות את הספר "תום סוייר"?

421884

וריאציות

יבוא יותר ספיציפי מהספריות; מועיל בעיקר אם יש אותו טיפוס בשתי ספריות

```
import java.io.InputStream;
import java.io.IOException;
import java.net.URL;
import java.util.Scanner;

public class Lecture2Program1 {

  public static
  void main(String[] args) throws IOException {
    URL         url     = new URL("http://www."
       + "gutenberg.org/cache/epub/74/pg74.txt");
    InputStream stream  = url.openStream();
    Scanner     scanner = new Scanner(stream);
    scanner.useDelimiter("\\A");
    String      book    = scanner.next();

    System.out.println("The text contains "
            + book.length() + " characters");
  }
}
```

שימוש בטיפוס מספריה בלי ליבא את הספריה או את הטיפוס

```
import java.io.InputStream;
import java.io.IOException;
// import java.net.URL;
import java.util.Scanner;

public class Lecture2Program1 {

  public static
  void main(String[] args) throws IOException {
    java.net.URL url = new java.net.URL("http://www."
       + "gutenberg.org/cache/epub/74/pg74.txt");
    InputStream stream  = url.openStream();
    Scanner     scanner = new Scanner(stream);
    scanner.useDelimiter("\\A");
    String      book    = scanner.next();

    System.out.println("The text contains "
            + book.length() + " characters");
  }
}
```

יבוא חבילות

- ספריות תוכנה בג'אווה נקראות חבילות (packages)
- השם המלא (fully qualified name) של טיפוס URL מהחבילה java.net הוא java.net.URL

1. אפשר ליבא (import) את כל הטיפוסים מהחבילה ולהשתמש בשם הטיפוס הקצר, URL
2. אפשר ליבא את הטיפוס הספיציפי
3. או שאפשר לציין את השם המלא של הטיפוס בלי ליבא אותו לתוכנית

חבילות משלנו

- גם אנחנו יכולים לארגן את התוכניות וקבצי קוד המקור שלנו בחבילות
- עוזר להבין כמויות גדולות של קוד וספריות קיימות
- (יש גם הבדל קטן בין היחס בין שני קבצי קוד מקור באותה חבילה ובין קבצים מחבילות אחרות, אבל זה לא הבדל חשוב ועדיף לא להשתמש בו)
- שמות החבילות ארוכים בדרך כלל כדי למנוע התנגשויות בין חבילות של ארגונים/פרוייקטים שונים; מאפשר להשתמש "בכל הקוד בעולם"

```
package tau.software1.lecture2;

import java.io.*;
...

public class Program1 {

  public static void main(String[] args) throws IOE
    URL         url     = new URL("http://www.guten
    InputStream stream  = url.openStream();
    Scanner     scanner = new Scanner(stream);
    scanner.useDelimiter("\\A");
    String      book    = scanner.next();

    System.out.println("The text contains " + book.
  }
}
```

```
package tau.software1.lecture2;

import java.io.*;
...
public class Program1 {

  public static void main(String[] args) throws IOE
    URL         url     = new URL("http://www.guten
    InputStream stream  = url.openStream();
    Scanner     scanner = new Scanner(stream);
    scanner.useDelimiter("\\A");
    String      book    = scanner.next();

    System.out.println("The text contains " + book.
  }
}
```

תיאורטית כנראה ששם החבילה
צריך להיות
il.ac.tau.cs.software1.lecture2
אבל לא נגזים...

חבילות וספריות קבצים

- קבצי קוד המקור של חבילה בשם tau.software1.lecture2 צריכים להישמר בספריה tau/software1/lecture2
- הספריות מקוננות, אבל החבילות לא; אין קשר בין החבילה הזאת וחבילה בשם tau.software1
- הקונבציה הרשמית היא ששמות חבילות צריכות להתאים ל-URL של הארגון, למשל il.ac.tau.cs.software1.lecture1; מיועד לאפשר להשתמש בכל שני טיפוסים בעולם ביחד
- אבל זו רק קונבנציה; מותר להיות לא קונבנציונלי

```
URL          url      = new URL("http...txt");
InputStream stream   = url.openStream();
Scanner      scanner = new Scanner(stream);
```

```
InputStream stream  =
  new URL("http...txt").openStream();
Scanner     scanner = new Scanner(stream);
```

```
Scanner scanner =
  new Scanner(new URL("http...txt").openStream());
```

```
URL          url     = new URL("http...txt");
InputStream stream  = url.openStream();
Scanner      scanner = new Scanner(stream);
```

```
InputStream stream  =
  new URL("http...txt").openStream();
Scanner      scanner = new Scanner(stream);
```

```
Scanner scanner =
  new Scanner(new URL("http...txt").openStream());
```

אם משתמשים במשתנה פעם אחת, לעיתים
קרובות לא צריך לתת לו שם (הוא עדיין קיים)

```
URL          url     = new URL("http...txt");
InputStream stream  = url.openStream();
Scanner      scanner = new Scanner(stream);
```

```
InputStream stream  =
  new URL("http...txt").openStream();
Scanner      scanner = new Scanner(stream);
```

```
Scanner scanner =
  new Scanner(new URL("http...txt").openStream());
```

מה עדיף? עניין של סגנון

הגרסה האחרונה קומפקטית אבל קשה להבנה

מה ראינו עד כה (1)

- מרחב שמות הטיפוסים (חבילות ומחלקות) ואיך להשתמש בו (יבוא, שימוש בלי יבוא, הגדרת חבילות ומחלקות)
- הטיפוסים URL, InputStream, Scanner
- שרשור מחרוזות (אופרטור +)
- הכרזה על שגרה (main במקרה שלנו) שעלולה לזרוק חריג כי שירות שהיא קוראת לו עלול לזרוק חריג (על שימוש בחריגים נדבר הרבה בהמשך הקורס)

מה ראינו עד כה (2)

- שירות מחלקה (השגרה main) עם שלושה משתנים שכל אחד מהם מתייחס לעצם אחד במהלך הריצה
- הפעלנו שירותי מופע על כל אחד מהעצמים
  - url.openStream()
  - scanner.useDelimiter(...), scanner.next()
  - book.length()
- ברור גם שהשירותים של Scanner (הבנאי Scanner ואולי גם next) קראו לשירותי מופע של stream כדי לקרוא את תוכן הדף מהאינטרנט

כמה **מילים** מרכיבות את
הספר "תום סוייר"?

```
public class Program2 {

  public static int wordcount(String text) {
    Scanner words = new Scanner(text);
    words.useDelimiter("\\s+");

    int counter = 0;
    while (words.hasNext()) {
      String word = words.next();
      counter++;
    }
    return counter;
  }

  public static void main(String[] args) ... {
    ...
    System.out.println("The text contains "
      + wordcount(book) + " words");
  }
}
```

השמטנו את
ה-pacakge,
import מהשקף

```
public class Program2 {

  public static int wordcount(String text) {
    Scanner words = new Scanner(text);
    words.useDelimiter("\\s+");

    int counter = 0;
    while (words.hasNext()) {
      String word = words.next();
      counter++;
    }
    return counter;
  }

  public static void main(String[] args) ... {
    ...
    System.out.println("The text contains "
      + wordcount(book) + " words");
  }
}
```

התבנית \s מתאימה
לכל תווי הרווח הלבן;
ה-+ מתאים למופע
אחד או יותר (של רווח)

```
public class Program2 {

  public static int wordcount(String text) {
    Scanner words = new Scanner(text);
    words.useDelimiter("\\s+");

    int counter = 0;
    while (words.hasNext()) {
      String word = words.next();
      counter++;
    }
    return counter;
  }

  public static void main(String[] args) ... {
    ...
    System.out.println("The text contains "
      + wordcount(book) + " words");
  }
}
```

בנאי

שרות (פקודה)

שרות (שאילתה)

שרות (פקודה+שאילתה)

עצמים, מחלקות, שירותים

- המשתנה words מתייחס לעצם (object) ממחלקה Scanner (class)
- העצם הוא מבנה נתונים בזיכרון ושירותי מופע שפועלים עליו
- בנאי (constructor) מאתחל את מבנה הנתונים
- פקודה משנה את המצב של מבנה הנתונים
- שאילתה מחזירה מידע על מצבו אבל לא משנה אותו מהותית
- יש שירותים שהם גם פקודה וגם שאילה; נפוץ אבל לא רעיון כל כך טוב

```
public class Program2 {

  public static int wordcount(String text) {
    Scanner words = new Scanner(text);
    words.useDelimiter("\\s+");

    int counter;
    for (counter=0; words.hasNext(); counter++) {
      String word = words.next();
    }
    return counter;
  }

  public static void main(String[] args) ... {
    ...
    System.out.println("The text contains "
      + wordcount(book) + " words");
  }
}
```

עוד וריאציה ללולאה

שני Scanners

- כאשר הפונקציה wordcount רצה, יש בזיכרון שני עצמים מהמחלקה Scanner
- scanner ב-main מתייחס לאחד
- words ב-workcount מתייחס לשני
- שירותי מופע שמופעלים על words משנים רק את המצב שלו, לא את המצב של ה-Scanner השני
- זה טיפוסי למחלקות שמתוכננות נכון

מה למדנו

- להגדיר שירותי מחלקה (פונקציות או פרוצדורות שאינן משתמשות במבנה הנתונים שמייצג עצם מהמחלקה שהן שייכות אליו), כאן wordcount; מוגדרים ככאלה בעזרת מילת המפתח static
- המחלקה Program2 משמשת לנו כמארז להגדרת שגרות (שירותי מחלקה)
- שימוש בשני עצמים מאותה מחלקה (Scanner) והפעלת שירותי מופע על כל אחד מהם
- וריאציות על לולאות

כמה מילים מרכיבות את הספר "תום סוייר"?

73837

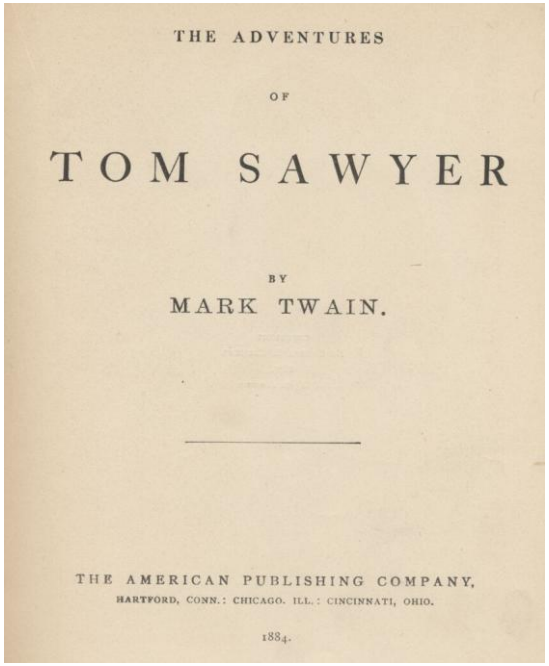
הערה על יעילות

- התוכניות שראינו קראו את כל הספר למשתנה בזיכרון
- עבור המשימות שרצינו לבצע, זה לא הכרחי; אפשר לקרוא כל פעם שורה לזיכרון ולספור את האותיות והמילים שיש בה, ולסכום
- יש פה חוסר יעילות בשימוש בזיכרון
- אבל התוכנית קצת יותר פשוטה מתוכנית יותר יעילה
- המחרוזת שמייצגת את הספר תופסת רק מגה-בית או חצי מגה; זה לא הרבה במחשב מודרני

מה המילה הנפוצה ביותר בספר "תום סוייר"?
(משתני מחלקה)

```
public class Program3 {

  public static Map<String,Integer> wordcounts =
    new TreeMap<String,Integer>();

  public static void wordHistogram(String text) {
    ... // fill wordcounts

  public static String mostFrequent() {
    ... // find most frequent word in wordcounts

  public static void main(String[] args) throws IOE
    ... // read the text into book
    wordHistogram(book);
    System.out.println(
      "The most frequent word is \""
      +mostFrequent() + "\"");
  }
```

"TOM!"
No answer.
"TOM!"
No answer.
"What's gone with that boy, I wonder?
You TOM!"
No answer.
...



wordcounts →

```
TOM     → 2
No      → 2
answer  → 2
```

```
public static Map<String,Integer> wordcounts
  = new TreeMap<String,Integer>();

public static void wordHistogram(String text) {
  Scanner words = new Scanner(text);
  words.useDelimiter("\\s+");

  while (words.hasNext()) {
    String word = words.next();
    if (wordcounts.containsKey(word))
      wordcounts.put(word,
                     wordcounts.get(word)+1);
    else wordcounts.put(word, 1);
  }
}

public static String mostFrequent() ...
```

```
public static Map<String,Integer> wordcounts
  = new TreeMap<String,Integer>();

public static void wordHistogram(String text) ...

public static String mostFrequent() {
  int max = -1;
  String most_frequent = null;

  for (String word: wordcounts.keySet()) {
    int count = wordcounts.get(word);
    if (count > max) {
      max = count;
      most_frequent = word;
    }
  }
  return most_frequent;
}
```

```
public static Map<String,Integer> wordcounts
  = new TreeMap<String,Integer>();
```

משתנה מחלקה (או שדה מחלקה)

```
public static void wordHistogram(String text) ...

public static String mostFrequent() {
  int max = -1;
  String most_frequent = null;
```

משתנים של שגרה

```
  for (String word: wordcounts.keySet()) {
    int count = wordcounts.get(word);
    if (count > max) {
      max = count;
      most_frequent = word;
    }
  }
  return most_frequent;
}
```

משתנים מקומיים ומשתני מחלקה

- משתנים שמוגדרים בשגרה (שירות) נוצרים כשקוראים לה ונעלמים מהזיכרון כשהיא חוזרת
- הזיכרון של משתנים כאלה מוקצה על מחסנית
- משתנים שמוגדרים מחוץ לשגרה (אבל תמיד בתוך מחלקה) עם מילת המפתח static הם משתני מחלקה (או שדות מחלקה)
- הזיכרון למשתני מחלקה מוקצה בתחילת הריצה של התוכנית והם זמינים כל זמן הריצה שלה

שימושים למשתני מחלקה

- השימוש שראינו כאן, להעברת מידע משגרה אחת (wordHistogram) לאחרת (mostFrequent) לא מצדיק בדרך כלל שימוש במשתנה מחלקה
  - אפשר היה להחזיר את wordcounts מ-wordHistogram ולהעביר אותו כארגומנט ל-mostFrequent
  - אבל זה מדגים את משך החיים של משתנים כאלה
- בדרך כלל משתמשים במשתנים כאלה לייצג ישות שיש רק אחד ממנה (singleton)
  - קובץ שמכיל את כל הודעות האזהרה של תוכנית
  - כתובת של שרת שיודע מה השעה
  - ...

מה המילה הנפוצה ביותר בספר "תום סוייר"?

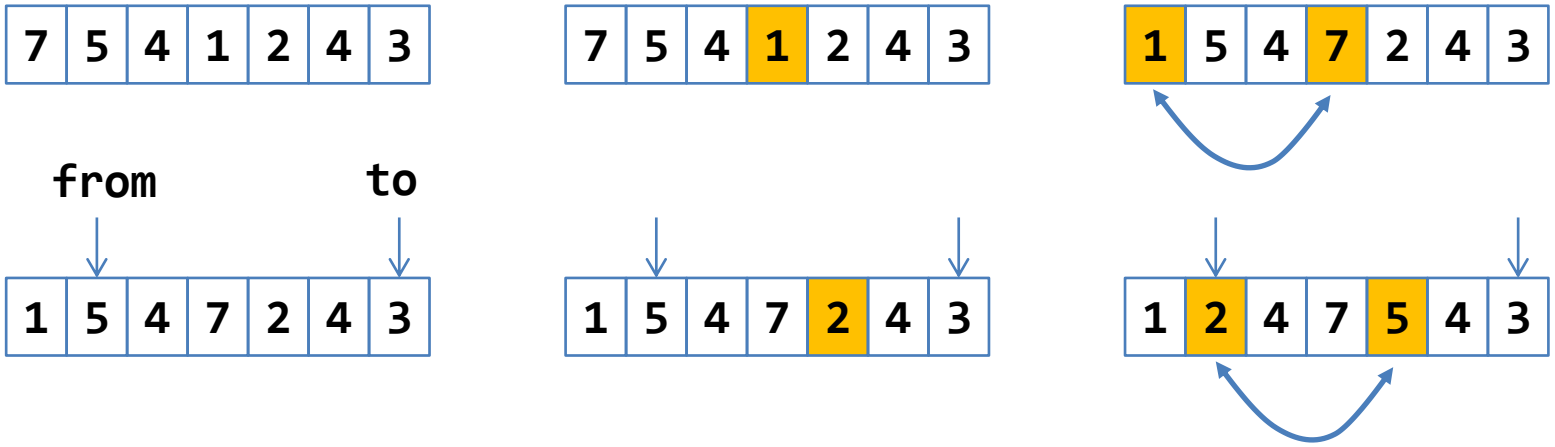
the

נמיין קצת...
(מערכים, switch)

```
public class Program4 {

  ...

  public static void main(String[] args) {

    int[] array = { 7, 5, 4, 1, 2, 4, 3 };

    sort(array);

    for (int i: array) System.out.println(i);
  }

}
```

אסטרטגיה למיון
(לא יעילה אבל פשוטה)

- נגדיר שגרה למיון של חלק של מערך בין שני מיקומים
- השגרה תמצא את האיבר המינימלי במערך (בעזרת שגרת עזר) תחליף את האיבר הזה עם הראשון, ותמיין את שארית המערך בין המקום השני והסוף, ברקורסיה



```
public class Program4 {

  public static int minIndex(int[] a,
    int from, int to) { ... }

  public static void sortRange(int[] a,
    int from, int to) { ... }

  public static void sort(int[] a) {
    sortRange(a, 0, a.length-1);
  }

  public static void main(String[] args) {
    int[] array = { 7, 5, 4, 1, 2, 4, 3 };
    sort(array);
    for (int i: array) System.out.println(i);
  }
}
```

```
public static int minIndex(int[] a,
                           int   from,
                           int   to) {
  int min = Integer.MAX_VALUE;
  int idx = -1;

  for (int i=from; i<=to; i++)
    if (a[i] <= min) {
      min = a[i];
      idx = i;
    }

  return idx;
}
```

```
public static void sortRange(int[] a,
                             int   from,
                             int   to) {
  if (from == to) return; // already sorted

  int i = minIndex(a, from, to);

  int min = a[i];
  a[i] = a[from];
  a[from] = min;

  sort(a, from+1, to); // recursion
}
```

```
public static void sort(int[] a) {
  sort(a, 0, a.length-1);
}
```

עוד גרסה, אולי טיפה יותר יעילה

(ומדגימה משפט switch)

```
public static void sort(int[] a) {
  switch (a.length){
    case 0:
    case 1:
      return;
    default:
      sort(a, 0, a.length-1);
  }
}
```

עוד על switch

- בחירה בין אפשרויות על פי שויון בין משתנה לקבועים
- הריצה קופצת ל-case של הקבוע המתאים (אם יש) וממשיכה משם
- אפשר לעצור את הריצה ולצאת מהבלוק של ה-switch בעזרת break
- אם אין case עם קבוע ששוה לערך המשתנה אבל יש תווית default, התוכנית קופצת אליה

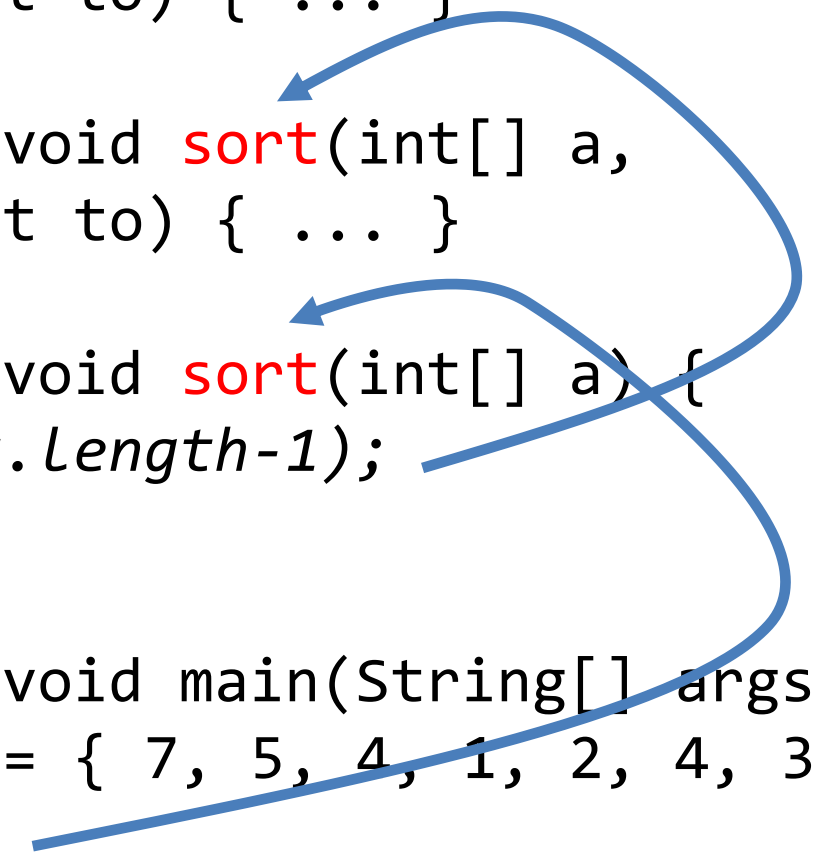
למשל,

```
public static void sort(int[] a) {
  switch (a.length){
    case 0:
      System.out.println("empty");
      break;
    case 1:
      return;
    default:
      sort(a, 0, a.length-1);
      break; // strictly optional
  }
}
```

העמסה (overloading)

```
public class Program4 {

  public static int minIndex(int[] a,
    int from, int to) { ... }

  public static void sortRange(int[] a,
    int from, int to) { ... }

  public static void sort(int[] a) {
    sortRange(a, 0, a.length-1);
  }

  public static void main(String[] args) {
    int[] array = { 7, 5, 4, 1, 2, 4, 3 };
    sort(array);
    for (int i: array) System.out.println(i);
  }
}
```

```
public class Program4 {

  public static int minIndex(int[] a,
    int from, int to) { ... }

  public static void sort(int[] a,
    int from, int to) { ... }

  public static void sort(int[] a) {
    sort(a, 0, a.length-1);
  }

  public static void main(String[] args) {
    int[] array = { 7, 5, 4, 1, 2, 4, 3 };
    sort(array);
    for (int i: array) System.out.println(i);
  }
}
```

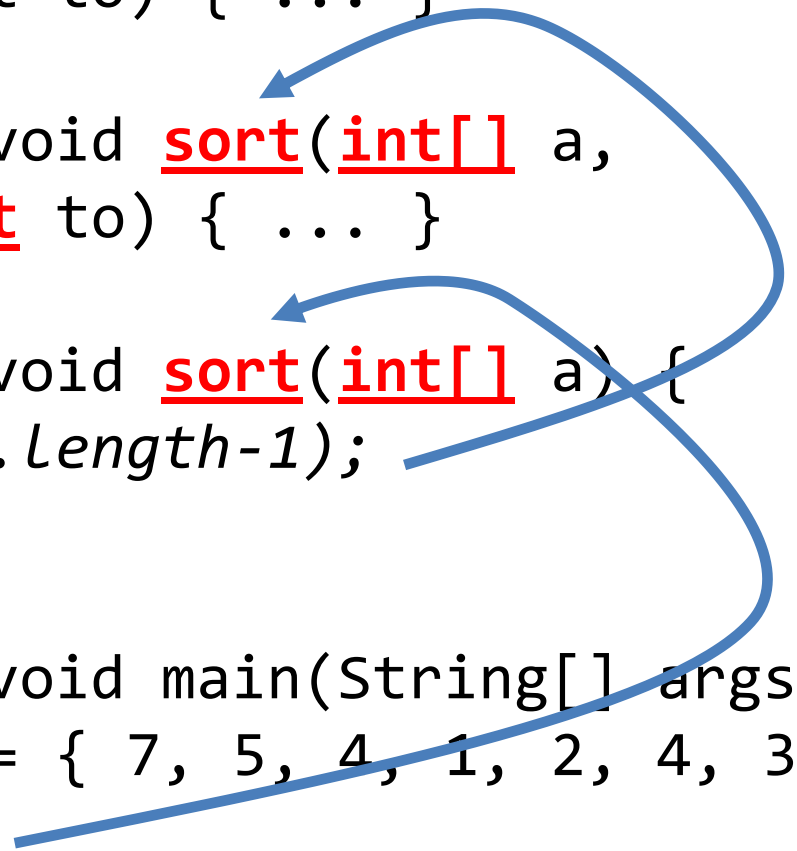


אבל אפשר גם

איך זה קורה?

- השם המלא של שירות (שגרה) כולל לא רק את השם שלה (sort) אלא גם את מספר וטיפוס הארגומנטים שלה

```
public class Program4 {

  public static int minIndex(int[] a,
    int from, int to) { ... }

  public static void sort(int[] a,
    int from, int to) { ... }

  public static void sort(int[] a) {
    sort(a, 0, a.length-1);
  }

  public static void main(String[] args) {
    int[] array = { 7, 5, 4, 1, 2, 4, 3 };
    sort(array);
    for (int i: array) System.out.println(i);
  }
}
```



שתי שגרות שונות!

איך הקומפיילר יודע לאיזה לקרוא

- לפי מספר הארגומנטים והטיפוס שלהם
- הטיפוסים לא חייבים להיות זהים
- הם צריכים להתאים; זה כולל תתי טיפוסים (נראה בהמשך) והמרות (כמעט) ללא איבוד מידע של פרימיטיבים
- לפעמים ברור מה יקרה ולפעמים לא (כדאי להמיר מפורשות)
- לפעמים הקומפיילר לא יודע מה לעשות ומרים ידיים

```java
package tau.software1.lecture2;

public class Program5 {

 public static void overloaded(int ix, double dy) {}
 public static void overloaded(double dx, int iy) {}
 public static int  overloaded(double dx, int iy) {}

 public static void main(String[] args) {
   overloaded(5, 3.14);
   overloaded(3.14, 5);
   overloaded(5,    5);
 }
}
```

איזו שגרה תקרא בכל קריאה?

```
package tau.software1.lecture2;

public class Program5 {

 public static void overloaded(int ix, double dy) {}
 public static void overloaded(double dx, int iy) {}
 public static int  overloaded(double dx, int iy) {}

 public static void main(String[] args) {
   overloaded(5, 3.14);
   overloaded(3.14, 5);
   overloaded(5,    5);
 }
}
```

טיפוס הערך המוחזר
אינו חלק מהשם המלא

שתי אפשרויות המרה
טובות באותה מידה

פרטים טכניים ללימוד עצמי

- break, continue בלולאות (דומה ל-Python)
- יש גם break, continue עם תווית (label) אבל צורך בזה מראה בדרך כלל על מבני בקרה מסובכים מדי; עדיף כנראה לפשט
- לולאות do {...} while () (כמו while אבל האיטרציה הראשונה תמיד מתבצעת)
- הגוף של לולאות ומשפטי תנאי יכול להיות משפט בודד (בלי סוגריים מסולסלים) או בלוק (עם)

סיכום השיעור

- חבילות
- שירותי מחלקה (static)
- משתני מחלקה (static)
- שימוש בשירותי מופע של עצמים מהספריה
- העמסה (overloading)
- עוד על מבני בקרה: לולאות, switch, וכו׳