Building Java Programs

Chapter 6: File Processing

Lecture 6-2: Advanced file input

reading: 6.3 - 6.5

self-check: #7-11

exercises: #1-4, 8-11

Additional `Scanner` methods

Method	Description
`nextLine()`	returns the next entire line of input
`hasNext()`	returns `true` if there are any more tokens of input to read (always true for console input)
`hasNextInt()`	returns `true` if there is a next token and it can be read as an `int`
`hasNextDouble()`	returns `true` if there is a next token and it can be read as a `double`
`hasNextLine()`	returns `true` if there are any more lines of input to read (always true for console input)

Tokenizing lines of a file

- A String `Scanner` can tokenize each line of a file.

```
Scanner input = new Scanner(new File("<file name>"));
while (input.hasNextLine()) {
    String line = input.nextLine();
    Scanner lineScan = new Scanner(line);

    <process the contents of this line...>;
}
```

Confusion w/ `nextLine`

- Using `nextLine` in conjunction with the token-based methods on the same `Scanner` can cause odd results.

  ```
  23    3.14
  Joe    "Hello world"
          45.2    19
  ```

  - You'd think you could read `23` and `3.14` with `nextInt` and `nextDouble`, then read `Joe "Hello world"` with `nextLine`.

    ```
    System.out.println(input.nextInt());       // 23
    System.out.println(input.nextDouble());     // 3.14
    System.out.println(input.nextLine());      //
    ```

  - But the `nextLine` call produces no output! Why?

Mixing lines and tokens

- Don't read both tokens and lines from the same `Scanner`:

```
23    3.14
Joe    "Hello world"
            45.2    19
```

```
input.nextInt()                                // 23
23\t3.14\nJoe\t"Hello world"\n\t\t45.2  19\n
  ^

input.nextDouble()                             // 3.14
23\t3.14\nJoe\t"Hello world"\n\t\t45.2  19\n
        ^

input.nextLine()                               // "" (empty!)
23\t3.14\nJoe\t"Hello world"\n\t\t45.2  19\n
          ^

input.nextLine()                   // "Joe\t\"Hello world\""
23\t3.14\nJoe\t"Hello world"\n\t\t45.2  19\n
                  ^
```

Line-and-token example

```
Scanner console = new Scanner(System.in);
System.out.print("Enter your age: ");
int age = console.nextInt();
System.out.print("Now enter your name: ");
String name = console.nextLine();
System.out.println(name + " is " + age + " years old.");
```

Log of execution (user input underlined):

```
Enter your age: 12
Now enter your name: Sideshow Bob
 is 12 years old.
```

- Why?
  - User's overall input: 12\nMarty Stepp
  - After `nextInt()`:   **12**\nMarty Stepp
                              ^
  - After `nextLine()`:   12\nMarty Stepp
                               ^

Recall: Hours program

```java
// Processes an employee input file and outputs each employee's hours data.
import java.io.*;      // for File
import java.util.*;    // for Scanner

public class Hours {
    public static void main(String[] args) throws FileNotFoundException {
        Scanner input = new Scanner(new File("hours.txt"));
        while (input.hasNextLine()) {
            String line = input.nextLine();
            Scanner lineScan = new Scanner(line);
            int id = lineScan.nextInt();          // e.g. 456
            String name = lineScan.next();         // e.g. "Brad"
            double sum = 0.0;
            int count = 0;
            while (lineScan.hasNextDouble()) {
                sum = sum + lineScan.nextDouble();
                count++;
            }

            double average = sum / count;
            System.out.println(name + " (ID#" + id + ") worked " +
                    sum + " hours (" + average + " hours/day)");
        }
    }
}
```

Hours v2 question

- Modify the `Hours` program to search for a person by ID:

    - Example:
    ```
    Enter an ID: 456
    Brad (ID#456) worked 36.8 hours (7.36 hours/day)
    ```

    - Example:
    ```
    Enter an ID: 293
    ID#293 not found
    ```

Hours v2 answer 1

```java
// This program searches an input file of employees' hours worked
// for a particular employee and outputs that employee's hours data.

import java.io.*;     // for File
import java.util.*;   // for Scanner

public class HoursWorked {
    public static void main(String[] args) throws FileNotFoundException {
        Scanner console = new Scanner(System.in);
        System.out.print("Enter an ID: ");
        int searchId = console.nextInt();         // e.g. 456

        Scanner input = new Scanner(new File("hours.txt"));
        String line = findPerson(input, searchId);
        if (line.length() > 0) {
            processLine(line);
        } else {
            System.out.println("ID#" + searchId + " was not found");
        }
    }

    ...
```

Hours v2 answer 2

```java
// Locates and returns the line of data about a particular person.
public static String findPerson(Scanner input, int searchId) {
    while (input.hasNextLine()) {
        String line = input.nextLine();
        Scanner lineScan = new Scanner(line);
        int id = lineScan.nextInt();              // e.g. 456
        if (id == searchId) {
            return line;                          // we found them!
        }
    }
    return "";    // not found, so return an empty line
}

// Totals the hours worked by the person and outputs their info.
public static void processLine(String line) {
    Scanner lineScan = new Scanner(line);
    int id = lineScan.nextInt();                  // e.g. 456
    String name = lineScan.next();                // e.g. "Brad"
    double hours = 0.0;
    int days = 0;
    while (lineScan.hasNextDouble()) {
        hours += lineScan.nextDouble();
        days++;
    }

    System.out.println(name + " (ID#" + id + ") worked " + hours + " hours ("
            + (hours / days) + " hours/day)");
}
}
```

10

IMDb movies problem

- Consider the following Internet Movie Database (IMDb) data:

  ```
  1 9.1 196376 The Shawshank Redemption (1994)
  2 9.0 139085 The Godfather: Part II (1974)
  3 8.8 81507 Casablanca (1942)
  ```

- Write a program that displays any movies containing a phrase:

  ```
  Search word? part

  Rank      Votes     Rating  Title
  2         139085    9.0     The Godfather: Part II (1974)
  40        129172    8.5     The Departed (2006)
  95        20401     8.2     The Apartment (1960)
  192       30587     8.0     Spartacus (1960)
  4 matches.
  ```

  - Is this a token or line-based problem?

IMDb answer 1

```java
// Displays IMDB's Top 250 movies that match a search string.
import java.io.*;      // for File
import java.util.*;    // for Scanner

public class Movies {
    public static void main(String[] args) throws FileNotFoundException {
        String searchWord = getWord();
        Scanner input = new Scanner(new File("imdb.txt"));
        String line = search(input, searchWord);

        int matches = 0;
        if (line.length() > 0) {
            System.out.println("Rank\tVotes\tRating\tTitle");
            while (line.length() > 0) {
                matches++;
                display(line, matches);
                line = search(input, searchWord);
            }
        }

        System.out.println(matches + " matches.");
    }

    // Asks the user for their search word and returns it.
    public static String getWord() {
        System.out.print("Search word: ");
        Scanner console = new Scanner(System.in);
        String searchWord = console.next();
        searchWord = searchWord.toLowerCase();
        System.out.println();
        return searchWord;
    }
    ...
```
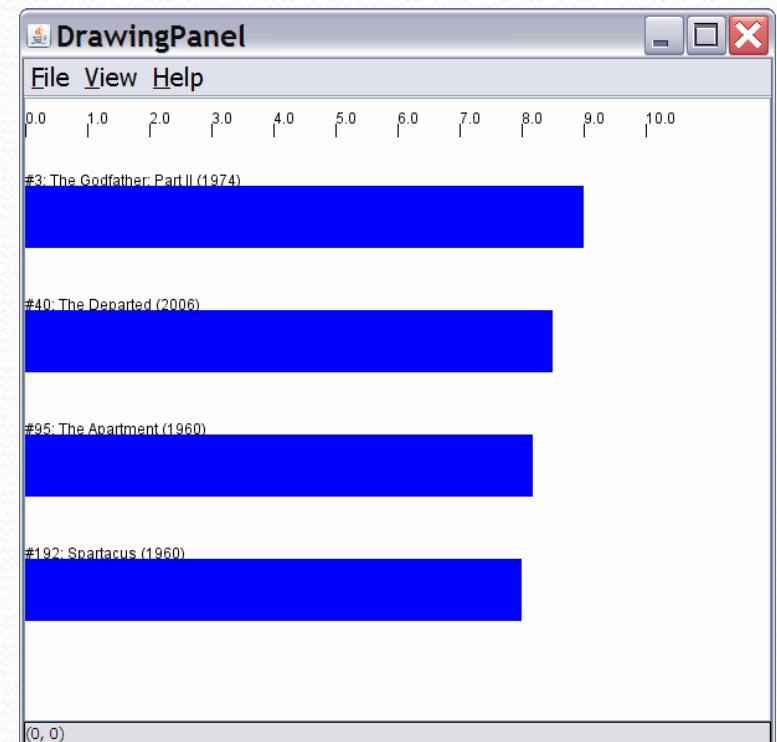
IMDb answer 2

```
...

    // Breaks apart each line, looking for lines that match the search word.
    public static String search(Scanner input, String searchWord) {
        while (input.hasNextLine()) {
            String line = input.nextLine();
            String lineLC = line.toLowerCase();     // case-insensitive match
            if (lineLC.indexOf(searchWord) >= 0) {
                return line;
            }
        }
        return "";    // not found
    }

    // Displays the line in the proper format on the screen.
    public static void display(String line, int matches) {
        Scanner lineScan = new Scanner(line);
        int rank = lineScan.nextInt();
        double rating = lineScan.nextDouble();
        int votes = lineScan.nextInt();
        String title = "";
        while (lineScan.hasNext()) {
            title += lineScan.next() + " ";     // the rest of the line
        }
        System.out.println(rank + "\t" + votes + "\t" + rating + "\t" + title);
    }
}
```

Graphical IMDB problem

- Turn our IMDb code into a graphical program.
  - top-left 0.0 tick mark at (0, 20)
  - ticks 10px tall, 50px apart

  - first blue bar top/left corner at (0, 70)
  - bars 50px tall
  - bars 50px wide per rating point
  - bars 100px apart vertically

Mixing graphics and text

- When mixing text/graphics, solve the problem in pieces.

  Do the text and file I/O first:
  - Display any welcome message and initial console input.
  - Open the input file and print some file data.
    (Perhaps print every line, the first token of each line, etc.)
  - Search the input file for the proper line record(s).

  Lastly, add the graphical output:
  - Draw any fixed graphics that do not depend on the file data.
  - Draw the graphics that do depend on the search result.

Graphical IMDb answer 1

```java
// Displays IMDB's Top 250 movies that match a search string.
import java.awt.*;      // for Graphics
import java.io.*;       // for File
import java.util.*;     // for Scanner

public class Movies2 {
    public static void main(String[] args) throws FileNotFoundException {
        String searchWord = getWord();
        Scanner input = new Scanner(new File("imdb.txt"));
        String line = search(input, searchWord);

        int matches = 0;
        if (line.length() > 0) {
            System.out.println("Rank\tVotes\tRating\tTitle");
            Graphics g = createWindow();
            while (line.length() > 0) {
                matches++;
                display(g, line, matches);
                line = search(input, searchWord);
            }
        }

        System.out.println(matches + " matches.");
    }

    // Asks the user for their search word and returns it.
    public static String getWord() {
        System.out.print("Search word: ");
        Scanner console = new Scanner(System.in);
        String searchWord = console.next();
        searchWord = searchWord.toLowerCase();
        System.out.println();
        return searchWord;
    }
    ...
```

Graphical IMDb answer 2

```
...
    // Breaks apart each line, looking for lines that match the search word.
    public static String search(Scanner input, String searchWord) {
        while (input.hasNextLine()) {
            String line = input.nextLine();
            String lineLC = line.toLowerCase();    // case-insensitive match
            if (lineLC.indexOf(searchWord) >= 0) {
                return line;
            }
        }
        return "";    // not found
    }

    // Displays the line in the proper format on the screen.
    public static void display(Graphics g, String line, int matches) {
        Scanner lineScan = new Scanner(line);
        int rank = lineScan.nextInt();
        double rating = lineScan.nextDouble();
        int votes = lineScan.nextInt();
        String title = "";
        while (lineScan.hasNext()) {
            title += lineScan.next() + " ";    // the rest of the line
        }
        System.out.println(rank + "\t" + votes + "\t" + rating + "\t" + title);
        drawBar(g, matches, title, rank, rating);
    }

...
```

Graphical IMDb answer 3

```
...

    // Creates a drawing panel and draws all fixed graphics.
    public static Graphics createWindow() {
        DrawingPanel panel = new DrawingPanel(600, 500);
        Graphics g = panel.getGraphics();

        for (int i = 0; i <= 10; i++) {          // draw tick marks
            int x = i * 50;
            g.drawLine(x, 20, x, 30);
            g.drawString(i + ".0", x, 20);
        }

        return g;
    }

    // Draws one red bar representing a movie's votes and ranking.
    public static void drawBar(Graphics g, int matches, String title,
                                  int rank, double rating) {
        int y = 70 + 100 * (matches - 1);
        int w = (int) (rating * 50);
        int h = 50;

        g.setColor(Color.BLUE);    // draw the blue bar for that movie
        g.fillRect(0, y, w, h);
        g.setColor(Color.BLACK);
        g.drawString("#" + rank + ": " + title, 0, y);
    }
}
```