Building Java Programs

Chapter 7
Arrays for Tallying; Text Processing

reading: 4.3, 7.6

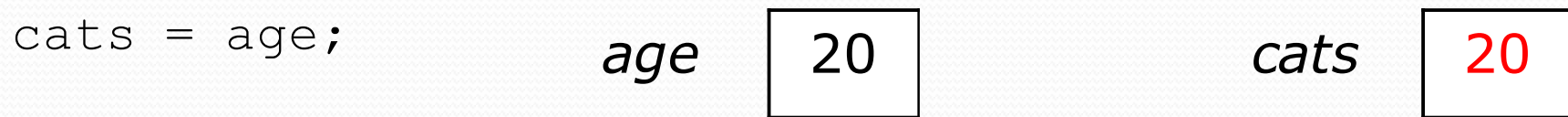# [“Hip”, “Hip”]

Hip Hip Array

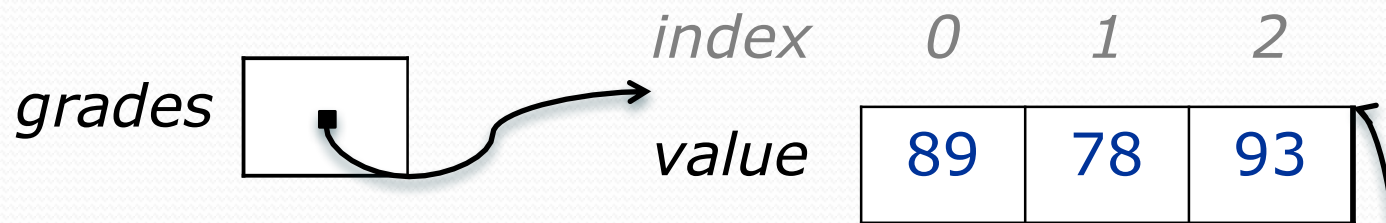# Value/Reference Semantics

- Variables of primitive types store values directly:

  age | 20 | *cats* | 3 |

- Values are copied from one variable to another:

  `cats = age;`   *age* | 20 | *cats* | 20 |

- Variables of object types store references to memory:

  index   0   1   2

  grades | ■ | ⟶   *value* | 89 | 78 | 93 |

- References are copied from one variable to another:

  `scores = grades;`   *scores* | ■ |

3

Text processing

reading: 7.2, 4.3

String traversals

- The `char`s in a `String` can be accessed using the `charAt` method.
    - accepts an `int` index parameter and returns the `char` at that index

    ```
    String food = "cookie";
    char firstLetter = food.charAt(0);     // 'c'

    System.out.println(firstLetter + " is for " + food);
    ```

- You can use a `for` loop to print or examine each character.

    ```
    String major = "CSE";
    for (int i = 0; i < major.length(); i++) {     // output:
        char c = major.charAt(i);                  // C
        System.out.println(c);                     // S
    }                                              // E
    ```

A counting problem

- Problem: Write a method `mostFrequentDigit` that returns the digit value that occurs most frequently in a number.

  - Example: The number 669260267 contains:
    one 0, two 2s, four 6es, one 7, and one 9.

    `mostFrequentDigit(669260267)` returns 6.

  - If there is a tie, return the digit with the lower value.

    `mostFrequentDigit(57135203)` returns 3.

A multi-counter problem

- We could declare 10 counter variables ...

  ```
  int counter0, counter1, counter2, counter3, counter4,
      counter5, counter6, counter7, counter8, counter9;
  ```

- But a better solution is to use an array of size 10.
  - The element at index i will store the counter for digit value i.
  - Example for 669260267:

index	0	1	2	3	4	5	6	7	8	9
value	1	0	2	0	0	0	4	1	0	0

  - How do we build such an array? And how does it help?

Creating an array of tallies

```
// assume n = 669260267
int[] counts = new int[10];
while (n > 0) {
    // pluck off a digit and add to proper counter
    int digit = n % 10;
    counts[digit]++;
    n = n / 10;
}
```

index	0	1	2	3	4	5	6	7	8	9
value	1	0	2	0	0	0	4	1	0	0

Tally solution

```java
// Returns the digit value that occurs most frequently in n.
// Breaks ties by choosing the smaller value.
public static int mostFrequentDigit(int n) {
    int[] counts = new int[10];
    while (n > 0) {
        int digit = n % 10;   // pluck off a digit and tally it
        counts[digit]++;
        n = n / 10;
    }

    // find the most frequently occurring digit
    int bestIndex = 0;
    for (int i = 1; i < counts.length; i++) {
        if (counts[i] > counts[bestIndex]) {
            bestIndex = i;
        }
    }

    return bestIndex;
}
```

Section attendance question

- Read a file of section attendance (*see next slide*):

```
yynyyynnynnyynyyynyyynynnnyyynynnyynyyynnynynn
nyynnyyyynynnnyyynnnyyynnnnnynynynynynyynynyn
yynynynyynyynyynnyyynyyynnnynynnynnyynyynynyny
```

- And produce the following output:

```
Section 1
Student points: [30, 25, 25, 20, 15]
Student grades: [100.0, 83.3, 83.3, 66.67, 50.0]

Section 2
Student points: [25, 30, 20, 20, 15]
Student grades: [83.3, 100.0, 66.67, 66.67, 50.0]

Section 3
Student points: [25, 25, 25, 30, 20]
Student grades: [83.3, 83.3, 83.3, 100.0, 66.67]
```

- Students earn 5 points for each section attended up to 30.

Section input file

student	1234512345123451234512345123451234512345
week	1 2 3 4 5 6 7 8 9
section 1	yynyyynnnynyyynyyyynynnyyynynnyyynyyynnnyyyynnynynn
section 2	nyynnyyyynynnnyyynnnyyynnnnnynynynynynyynynyn
section 3	yynynynyynyynyynnyyynyyynnnynynnynnyynyyynynyny

- Each line represents a section.
- A line consists of 9 weeks' worth of data.
  - Each week has 5 characters because there are 5 students.
- Within each week, each character represents one student.
  - `n` means the student was absent   (+0 points)
  - `y` means they attended and did the problems   (+5 points)

Section attendance question

- Read a file of section attendance:

```
yynyyynnynnyynyyynynnyyynynnyynyyynnnyyynnynynn
nyynnyyyynynnnyyynnnyyynnnnnynynynynynyynynyn
yynynynyynyynyynnyyynyynnnynynnynnyynyynynyny
```

- And produce the following output:

```
Section 1
Student points: [30, 25, 25, 20, 15]
Student grades: [100.0, 83.3, 83.3, 66.67, 50.0]

Section 2
Student points: [25, 30, 20, 20, 15]
Student grades: [83.3, 100.0, 66.67, 66.67, 50.0]

Section 3
Student points: [25, 25, 25, 30, 20]
Student grades: [83.3, 83.3, 83.3, 100.0, 66.67]
```

- Students earn 5 points for each section attended up to 30.

Section attendance answer

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

public class Sections {
    public static void main(String[] args) throws FileNotFoundException {
        Scanner input = new Scanner(new File("sections.txt"));
        int section = 1;
        while (input.hasNextLine()) {
            String line = input.nextLine();        // process one section
            int[] points = new int[5];
            for (int i = 0; i < line.length(); i++) {
                int student = i % 5;
                if (line.charAt(i) == 'y') {        // c == 'y' or 'n'
                    points[student] += 5;
                    points[student] = Math.min(30, points[student]);
                }
            }

            double[] grades = new double[5];
            for (int i = 0; i < points.length; i++) {
                grades[i] = 100.0 * points[i] / 30.0;
            }

            System.out.println("Section " + section);
            System.out.println("Student points: " + Arrays.toString(points));
            System.out.println("Student grades: " + Arrays.toString(grades));
            System.out.println();
            section++;
        }
    }
}
```

Data transformations

- In many problems we transform data between forms.
  - Example: digits ☻ count of each digit ☻ most frequent digit
  - Often each transformation is computed/stored as an array.
  - For structure, a transformation is often put in its own method.

- Sometimes we map between data and array indexes.

  - by position   (store the ith value we read at index i)
  - tally   (if input value is i, store it at array index i)
  - explicit mapping (count 'J' at index 0, count 'X' at index 1)

Array param/return answer

```java
// This program reads a file representing which students attended
// which discussion sections and produces output of the students'
// section attendance and scores.

import java.io.*;
import java.util.*;

public class Sections2 {
    public static void main(String[] args) throws FileNotFoundException {
        Scanner input = new Scanner(new File("sections.txt"));
        int section = 1;
        while (input.hasNextLine()) {
            // process one section
            String line = input.nextLine();
            int[] points = countPoints(line);
            double[] grades = computeGrades(points);
            results(section, points, grades);
            section++;
        }
    }

    // Produces all output about a particular section.
    public static void results(int section, int[] points, double[] grades) {
        System.out.println("Section " + section);
        System.out.println("Student scores: " + Arrays.toString(points));
        System.out.println("Student grades: " + Arrays.toString(grades));
        System.out.println();
    }

    ...
```

Array param/return answer

```
...
    // Computes the points earned for each student for a particular section.
    public static int[] countPoints(String line) {
        int[] points = new int[5];
        for (int i = 0; i < line.length(); i++) {
            int student = i % 5;
            if (line.charAt(i) == 'y') {        // c == 'y'  or  c == 'n'
                points[student] += 5;
                points[student] = Math.min(30, points[student]);
            }
        }
        return points;
    }

    // Computes the percentage for each student for a particular section.
    public static double[] computeGrades(int[] points) {
        double[] grades = new double[5];
        for (int i = 0; i < points.length; i++) {
            grades[i] = 100.0 * points[i] / 30.0;
        }
        return grades;
    }
}
```

PrintStream question

- **Modify our previous Sections program to use a** `PrintStream` **to output to the file** `sections_out.txt.`

```
Section 1
Student points: [30, 25, 25, 20, 15]
Student grades: [100.0, 83.3, 83.3, 66.67, 50.0]

Section 2
Student points: [25, 30, 20, 20, 15]
Student grades: [83.3, 100.0, 66.67, 66.67, 50.0]

Section 3
Student points: [25, 25, 25, 30, 20]
Student grades: [83.3, 83.3, 83.3, 100.0, 66.67]
```

System.out and PrintStream

- The console output object, `System.out`, is a `PrintStream`.

```
PrintStream out1 = System.out;
PrintStream out2 = new PrintStream(new File("data.txt"));
out1.println("Hello, console!");    // goes to console
out2.println("Hello, file!");       // goes to file
```

- A reference to it can be stored in a `PrintStream` variable.
  - Printing to that variable causes console output to appear.

- You can pass `System.out` as a parameter to a method expecting a `PrintStream`.
  - Allows methods that can send output to the console or a file.

PrintStream answer

```java
// Section attendance program
// This version uses a PrintStream for output.
import java.io.*;
import java.util.*;

public class Sections {
    public static void main(String[] args) throws FileNotFoundException {
        Scanner input = new Scanner(new File("sections.txt"));
        PrintStream out = new PrintStream(new File("sections_out.txt"));
        while (input.hasNextLine()) {    // process one section
            String line = input.nextLine();
            int[] attended = countAttended(line);
            int[] points = computePoints(attended);
            double[] grades = computeGrades(points);
            results(attended, points, grades, out);
        }
    }

    // Produces all output about a particular section.
    public static void results(int[] attended, int[] points,
            double[] grades, PrintStream out) {
        out.println("Sections attended: " + Arrays.toString(attended));
        out.println("Sections scores: " + Arrays.toString(points));
        out.println("Sections grades: " + Arrays.toString(grades));
        out.println();
    }
    ...
```

Prompting for a file name

- We can ask the user to tell us the file to read.
  - The file name might have spaces; use `nextLine()`, not `next()`

```
// prompt for input file name
Scanner console = new Scanner(System.in);
System.out.print("Type a file name to use: ");
String filename = console.nextLine();
Scanner input = new Scanner(new File(filename));
```

- What if the user types a file name that does not exist?