Building Java Programs

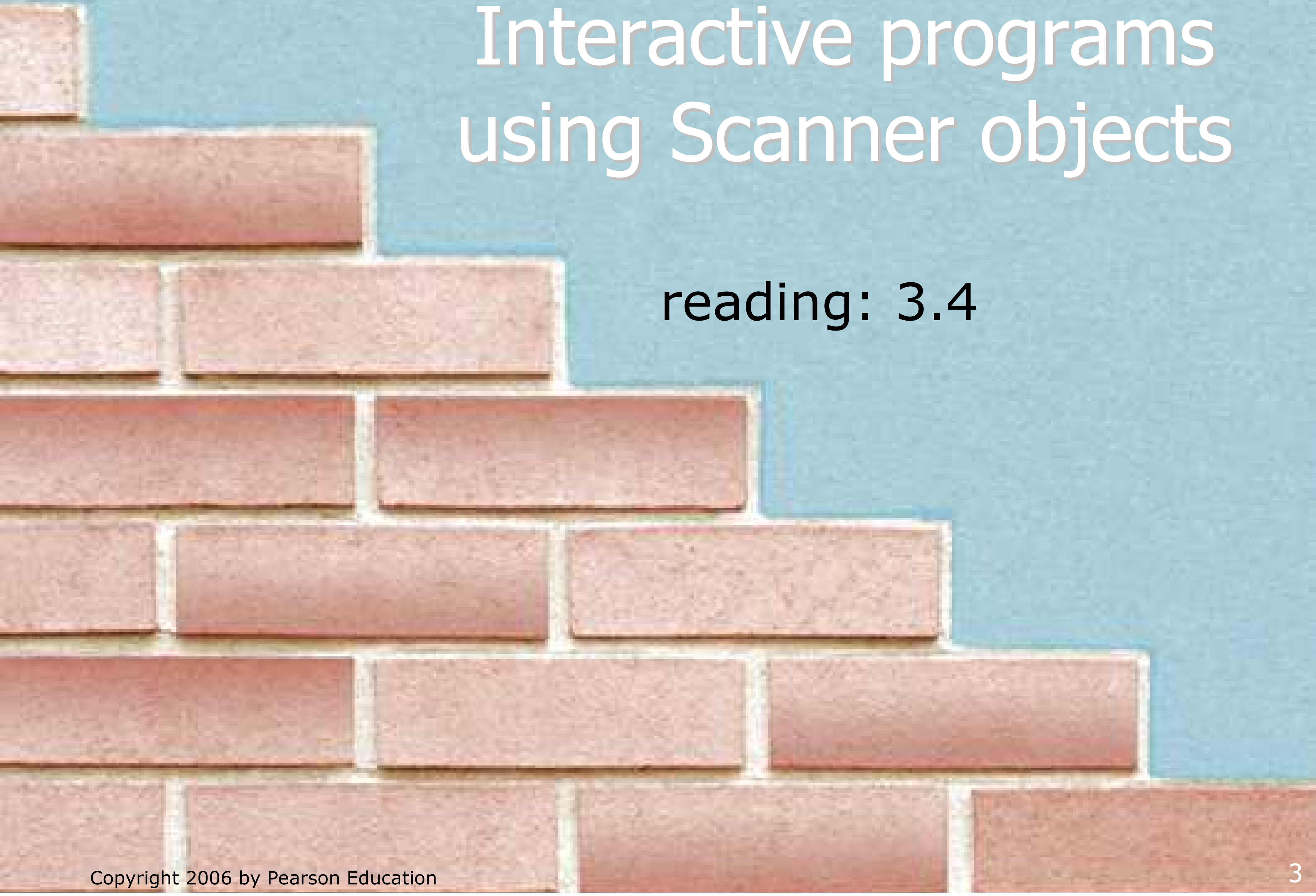
Chapter 3: Parameters, Return, and Interactive Programs with Scanner

Lecture outline

- console input with `Scanner` objects
  - input tokens
  - `Scanner` as a parameter to a method
  - cumulative sums and `Scanner`

Interactive programs using Scanner objects

reading: 3.4



Interactive programs

- We have written programs that print console output.

- It is also possible to read *input* from the console.
  - The user types the input into the console.
  - We can capture the input and use it in our program.
  - Such a program is called an *interactive program*.

- Interactive programs can be challenging:
  - Computers and users think in very different ways.
  - Users tend to misbehave.

Input and System.in

- `System.out`
  - An object with methods named `println` and `print`
- `System.in`
  - not intended to be used directly
  - We use a second object, from a class `Scanner`, to help us.

- Constructing a `Scanner` object to read console input:

  ```
  Scanner <name> = new Scanner(System.in);
  ```

  - Example:

  ```
  Scanner console = new Scanner(System.in);
  ```

Scanner methods

Method	Description
`nextInt()`	reads user input as an `int`
`nextDouble()`	reads user input as a `double`
`next()`	reads user input as a `String`

- Each method waits until the user types input and presses Enter.
  - The value typed is *returned*.
- **prompt**: A message telling the user what input to type.

```
System.out.print("How old are you? ");    // prompt
int age = console.nextInt();
System.out.println("You'll be 40 in " + (40 - age)
                   + " years.");
```

Java class libraries, import

- **Java class libraries**: Classes included with Java's JDK.
  - organized into groups named *packages*
  - To use a package, put an *import declaration* in your program.

- import declaration, general syntax:

```
// put this at the very top of your program
import <package name> .*;
```

- Scanner is in a package named `java.util`

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

Example Scanner usage

```
import java.util.*;   // so that I can use Scanner

public class ReadSomeInput {
    public static void main(String[] args) {
        Scanner console = new Scanner(System.in);

        System.out.print("What is your first name? ");
        String name = console.next();

        System.out.print("And how old are you? ");
        int age = console.nextInt();

        System.out.println(name + " is " + age);
        System.out.println("That's quite old!");
    }
}
```

- Output (user input underlined):

```
What is your first name? Ruth
How old are you? 14
Ruth is 14
That's quite old!
```

Another Scanner example

```
import java.util.*;   // so that I can use Scanner

public class ScannerSum {
    public static void main(String[] args) {
        Scanner console = new Scanner(System.in);

        System.out.print("Please type three numbers: ");
        int num1 = console.nextInt();
        int num2 = console.nextInt();
        int num3 = console.nextInt();

        int sum = num1 + num2 + num3;
        System.out.println("The sum is " + sum);
    }
}
```

- Output (user input underlined):

```
Please type three numbers: 8 6 13
The sum is 27
```

  - Notice that the `Scanner` can read multiple values from one line.

Input tokens

- **token**: A unit of user input, as read by the `Scanner`.
  - Tokens are separated by whitespace (spaces, tabs, new lines).
  - How many tokens appear on the following line of input?

    ```
    23   John Smith    42.0 "Hello world"   $2.50    "19"
    ```

- When a token is not the type you ask for, it crashes.

  Example:

  ```
  System.out.print("What is your age? ");
  int age = console.nextInt();
  ```

  Output (user's input is underlined):

  ```
  What is your age? Timmy
  java.util.InputMismatchException
          at java.util.Scanner.throwFor(Unknown Source)
          at java.util.Scanner.next(Unknown Source)
          at java.util.Scanner.nextInt(Unknown Source)
          ...
  ```

Scanners as parameters

- If many methods read input, declare a `Scanner` in `main` and pass it to the others as a parameter.
  - All the methods share the same `Scanner` object.

```
public static void main(String[] args) {
    Scanner console = new Scanner(System.in);
    int sum = readSum3(console);
    System.out.println("The sum is " + sum);
}

// Prompts for 3 numbers and returns their sum.
public static int readSum3(Scanner console) {
    System.out.print("Type 3 numbers: ");
    int num1 = console.nextInt();
    int num2 = console.nextInt();
    int num3 = console.nextInt();
    return num1 + num2 + num3;
}
```

Scanner BMI question

A person's body mass index (BMI) is computed by the following formula:

$$BMI = \frac{weight}{height^2} \times 703$$

- Write a program that produces the following output:

```
This program reads in data for two people
and computes their body mass index (BMI)
and weight status.

Enter next person's information:
height (in inches)? 62.5
weight (in pounds)? 130.5

Enter next person's information:
height (in inches)? 58.5
weight (in pounds)? 90

Person #1 body mass index = 23.485824
Person #2 body mass index = 18.487836949375414
Difference = 4.997987050624587
```

Scanner BMI solution

```
// This program computes two people's body mass index (BMI)
// and compares them.  The code uses parameters and returns.

import java.util.*;  // so that I can use Scanner

public class BMI {
    public static void main(String[] args) {
        introduction();
        Scanner console = new Scanner(System.in);

        double bmi1 = processPerson(console);
        double bmi2 = processPerson(console);

        // report overall results
        System.out.println("Person #1 body mass index = " + bmi1);
        System.out.println("Person #2 body mass index = " + bmi2);
        double difference = Math.abs(bmi1 - bmi2);
        System.out.println("Difference = " + difference);
    }

    // prints a welcome message explaining the program
    public static void introduction() {
        System.out.println("This program reads in data for two people");
        System.out.println("and computes their body mass index (BMI)");
        System.out.println("and weight status.");
        System.out.println();
    }
    ...
```

Scanner BMI solution, cont.

```
    ...

    // reads information for one person, computes their BMI, and returns it
    public static double processPerson(Scanner console) {
        System.out.println("Enter next person's information:");
        System.out.print("height (in inches)? ");
        double height = console.nextDouble();

        System.out.print("weight (in pounds)? ");
        double weight = console.nextDouble();
        System.out.println();

        double bmi = getBMI(height, weight);
        return bmi;
    }

    // Computes a person's body mass index based on their height and weight
    // and returns the BMI as its result.
    public static double getBMI(double height, double weight) {
        double bmi = weight / (height * height) * 703;
        return bmi;
    }
}
```

Types int and double

- Printing double values can be ugly:

```
double result = 1.0 / 3.0;
System.out.println(result);    // 0.3333333333333
```

- Can we print it with only 2 digits after the decimal?

- Rounding the number doesn't help:

```
double result = 1.0 / 3.0;
System.out.println(Math.round(result));    // 0
```

Rounding real numbers

- To round to N places:
  - multiply by 10^N
  - round
  - divide by 10^N

- Example:

```
double result = 1.0 / 3.0;        // 0.333333333333
result = result * 100;            // 33.333333333
result = Math.round(result);      // 33.0
result = result / 100;            // 0.33
System.out.println(result);
```

System.out.printf

- `System.out.printf` prints formatted text.

  `System.out.printf("`***<format string>***`", `***<parameters>***`);`

  - The format string contains *format placeholders* to specify how to insert the parameters into the string.

%d	an integer
%f	a real number
%s	a string

  - A format placeholder can specify a width:

%8d	an integer, 8 characters wide, right-aligned
%-8d	an integer, 8 characters wide, left-aligned
%12f	a real number, 12 characters wide
%.4f	a real number, 4 characters after decimal
%6.2f	a real number, 6 total characters wide, 2 after decimal

- Example:

```
double d = 1.0 / 3.0;                  // 0.33333333333
System.out.printf("It's %8.2f\n", d);  // It's     0.33
```

printf examples

```
int x = 38, y = 152;
int grade = 86;
double angle = 87.4163;
String veggie = "carrot";

System.out.printf("hello there\n");
System.out.printf("x=%d and y=%d\n", x, y);
System.out.printf("score is %d%%\n", (grade + 5));
System.out.printf("oh my !%d!%6d%6d\n", grade, x, y);
System.out.printf("huh? %.2f %16.5f\n", angle, angle);
System.out.printf("%s%12s!%-8s!\n", veggie, veggie, veggie);
```

Output:

```
hello there
x=38 and y=152
score is 91%
oh my !86!    38   152
huh? 87.42         87.41630
carrot      carrot!carrot  !
```

Scanner and cumulative sum

- We can do a cumulative sum of user input:

```
Scanner console = new Scanner(System.in);
int sum = 0;
for (int i = 1; i <= 100; i++) {
    System.out.print("Type a number: ");
    sum += console.nextInt();
}
System.out.println("The sum is " + sum);
```

User-guided cumulative sum

- User input can control the number of loop repetitions:
  - Desired example output:

```
How many numbers to add? 3
Type a number: 2
Type a number: 6
Type a number: 3
The sum is 11
```

  - Answer:

```
Scanner console = new Scanner(System.in);
System.out.print("How many numbers to add? ");
int count = console.nextInt();

int sum = 0;
for (int i = 1; i <= count; i++) {
    System.out.print("Type a number: ");
    sum += console.nextInt();
}
System.out.println("The sum is " + sum);
```

Cumulative sum question

- Write a program that reads input of the number of hours two employees have worked and displays each employee's total and the overall total hours.
  - The company doesn't pay overtime, so cap any day at 8 hours.
- Example log of execution:

```
Employee 1: How many days? 3
Hours? 6
Hours? 12
Hours? 5
Employee 1's total hours = 19

Employee 2: How many days? 2
Hours? 11
Hours? 6
Employee 2's total hours = 14

Total hours for both = 33
```

Cumulative sum answer

```
// Computes the total paid hours worked by two employees.
// The company does not pay for more than 8 hours per day.
// Uses a "cumulative sum" loop to compute the total hours.

import java.util.*;

public class Hours {
    public static void main(String[] args) {
        Scanner console = new Scanner(System.in);

        int hours1 = processEmployee(console, 1);
        int hours2 = processEmployee(console, 2);

        int total = hours1 + hours2;
        System.out.println("Total hours for both = " + total);
    }

    ...
```

Cumulative sum answer 2

```
    ...

    // Reads hours information about one employee with the given number.
    // Returns the total hours worked by the employee.
    public static int processEmployee(Scanner console, int number) {
        System.out.print("Employee " + number + ": How many days? ");
        int days = console.nextInt();

        // totalHours is a cumulative sum of all days' hours worked.
        int totalHours = 0;
        for (int i = 1; i <= days; i++) {
            System.out.print("Hours? ");
            int hours = console.nextInt();
            totalHours += Math.min(hours, 8);   // cap at 8 hours/day
        }

        System.out.println("Employee " + number + "'s total hours = "
                           + totalHours);
        System.out.println();
        return totalHours;
    }
}
```