

Building Java Programs

Chapter 3

Lecture 3-3: Interactive Programs w/ Scanner

reading: 3.3 - 3.4

self-check: #16-19

exercises: #11

videos: Ch. 3 #4

Interactive programs

- We have written programs that print console output, but it is also possible to read *input* from the console.
 - The user types input into the console. We 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 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);
```

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.

- **Syntax:**

```
// put this at the very top of your program
```

```
import packageName.* ;
```

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

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

- To use `Scanner`, you must place the above line at the top of your program (before the `public class` header).

Scanner methods

Method	Description
<code>nextInt()</code>	reads a token of user input as an <code>int</code>
<code>nextDouble()</code>	reads a token of user input as a <code>double</code>
<code>next()</code>	reads a token of user input as a <code>String</code>
<code>nextLine()</code>	reads a <i>line</i> of user input as a <code>String</code>

- Each method waits until the user presses Enter.
 - The value typed is returned.

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

- **prompt:** A message telling the user what input to type.

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("How old are you? ");
        int age = console.nextInt();

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

- Output (user input underlined):

```
How old are you? 14
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
```

- 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, newlines).
 - 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.

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

Output:

```
What is your age? Timmy  
java.util.InputMismatchException  
    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.

```
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;  
}
```

Cumulative sum

reading: 4.1

self-check: Ch. 4 #1-3

exercises: Ch. 4 #1-6

Adding many numbers

- How would you find the sum of all integers from 1-1000?

```
int sum = 1 + 2 + 3 + 4 + ... ;  
System.out.println("The sum is " + sum);
```

- What if we want the sum from 1 - 1,000,000?
Or the sum up to any maximum?
- We could write a method that accepts the max value as a parameter and prints the sum.
 - How can we generalize code like the above?

A failed attempt

- An incorrect solution for summing 1-1000:

```
for (int i = 1; i <= 1000; i++) {  
    int sum = 0;  
    sum = sum + i;  
}  
  
// sum is undefined here  
System.out.println("The sum is " + sum);
```

- `sum`'s scope is in the `for` loop, so the code does not compile.
- **cumulative sum**: A variable that keeps a sum in progress and is updated repeatedly until summing is finished.
 - The `sum` in the above code is an attempt at a cumulative sum.

Fixed cumulative sum loop

- A corrected version of the sum loop code:

```
int sum = 0;
```

```
for (int i = 1; i <= 1000; i++) {  
    sum = sum + i;  
}  
System.out.println("The sum is " + sum);
```

Key idea:

- Cumulative sum variables must be declared *outside* the loops that update them, so that they will exist after the loop.

Cumulative product

- This cumulative idea can be used with other operators:

```
int product = 1;
for (int i = 1; i <= 20; i++) {
    product = product * 2;
}
System.out.println("2 ^ 20 = " + product);
```

- How would we make the base and exponent adjustable?

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 = sum + console.nextInt();  
}  
System.out.println("The sum is " + sum);
```

User-guided cumulative sum

```
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 = sum + console.nextInt();
}
System.out.println("The sum is " + sum);
```

- Output:

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


Cumulative sum question

- Write a program that reads two employees' hours and displays each employee's total and the overall total hours.
 - The company doesn't pay overtime; cap each 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 (6.3 / day)
```

```
Employee 2: How many days? 2  
Hours? 11  
Hours? 6  
Employee 2's total hours = 14 (7.0 / day)
```

```
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 an employee with the given number.
// Returns 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 = totalHours + Math.min(hours, 8);
    }

    double hoursPerDay = (double) totalHours / days;
    System.out.printf("Employee %d's total hours = %d (%.1f / day)\n",
        number, totalHours, hoursPerDay);

    System.out.println();
    return totalHours;
}
}
```

Cumulative sum question

- Write a modified version of the `Receipt` program from Ch.2 that prompts the user for how many people ate and how much each person's dinner cost.
 - Display results in format below, with \$ and 2 digits after the .
- Example log of execution:

```
How many people ate? 4  
Person #1: How much did your dinner cost? 20.00  
Person #2: How much did your dinner cost? 15  
Person #3: How much did your dinner cost? 25.0  
Person #4: How much did your dinner cost? 10.00
```

```
Subtotal: $70.00  
Tax: $5.60  
Tip: $10.50  
Total: $86.10
```

Cumulative sum answer

```
// This program enhances our Receipt program using a cumulative sum.  
import java.util.*;
```

```
public class Receipt2 {  
    public static void main(String[] args) {  
        Scanner console = new Scanner(System.in);  
        System.out.print("How many people ate? ");  
        int people = console.nextInt();  
        double subtotal = 0.0;           // cumulative sum  
  
        for (int i = 1; i <= people; i++) {  
            System.out.print("Person #" + i +  
                ": How much did your dinner cost? ");  
            double personCost = console.nextDouble();  
            subtotal = subtotal + personCost; // add to sum  
        }  
        results(subtotal);  
    }  
  
    // Calculates total owed, assuming 8% tax and 15% tip  
    public static void results(double subtotal) {  
        double tax = subtotal * .08;  
        double tip = subtotal * .15;  
        double total = subtotal + tax + tip;  
  
        System.out.printf("Subtotal: $%.2f\n", subtotal);  
        System.out.printf("Tax: $%.2f\n", tax);  
        System.out.printf("Tip: $%.2f\n", tip);  
        System.out.printf("Total: $%.2f\n", total);  
    }  
}
```