Recurring problems and patterns to solve them

Learning Objectives

- You recognize that certain sub-problems, such as reading input or calculations, recur in programs.
- You're aware of solution models to certain sub-problems.
- You practice combining solution patterns used on sub-problems to solve broader ones.

The same small problems, or "sub-problems", reappear in programs time after time: "Read input from the user", "Calculate the sum of values", and so forth.

Let's look at a few sub-problems and patterns for solving them.

Reading User Input

The solution pattern for programming tasks involving reading user input is straightforward. If the program needs to read from the user, a Scanner helper tool is created for the task. The Scanner is created in the main method after the line `public static void main(String[] args) {`. To use the Scanner, it needs to be made available in the program through the statement `import java.util.Scanner;`, which comes before the class definition (`public class ...`). Importing the Scanner tool makes it available for the program.

```
// Making the scanner available in the program
import java.util.Scanner;

public class Program {
    public static void main(String[] main) {
        // Creating the scanner
        Scanner reader = new Scanner(System.in);

        // Examples of reading different types of user input
        String text = reader.nextLine();
        int number = Integer.valueOf(reader.nextLine());
        double numberWithDecimals = Double.valueOf(reader.nextLine());
        boolean trueOrFalse = Boolean.valueOf(reader.nextLine());

    }
}
```

Quiz:

Reacting to input

Points:

1/1

What should the user give as input for the program below to print 12?

```
import java.util.Scanner;

public class Program {
    public static void main(String[] main) {
        Scanner scanner = new Scanner(System.in);

        int number = Integer.valueOf(scanner.nextLine());
        int result = number * 2;
        System.out.println(result);

    }
}
```

Select the correct answer

1

12

✓ 6

8

5

nothing

2

55

7

The answer is correct

Answered Tries remaining: 1

Calculating

We quite often need to calculate something in a program, such as an average or a sum. The solution pattern to solve such problems is as follows.

1. Define the inputs required for the calculation and declare variables for them. Input refers to the values used in the calculation. You can typically identify the type of inputs from the problem description.
2. Identify the operation needed, and declare a variable for the result of the calculation. Perform the calculation using the inputs, and assign the result to the variable that was reserved for it. The type of the result can also usually be identified from the problem description.
3. Once the calculation is done, do something with its result. This can mean printing the result of a computation, or, for example, using it in calculating an average by dividing a sum of the inputs by their count.

For example, the solution pattern for the problem *Create a program to calculate the sum of two integers* is the following.

```java
// Identifying the input values and declaring the variables for them
int first = 1;
int second = 2;

// Identifying the operation and declaring a variable for the result
int sum = first + second;

// printing the result of the calculation
System.out.println("The sum of " + first + " and " + second + " is " + sum);
```

A program that both reads and calculates combines both of these patterns. One that calculates the product of two integers provided by the user looks like this:

```java
// Making the scanner available in the program
import java.util.Scanner;

public class Program {
    public static void main(String[] main) {
        // Creating the scanner
        Scanner reader = new Scanner(System.in);

        // Identifying the input values and declaring the variables for them
        int first = 1;
        int second = 2;

        // Assigning the user input to the variables
        first = Integer.valueOf(reader.nextLine());
        second = Integer.valueOf(reader.nextLine());

        // Identifying the operation and declaring a variable for the result
        int product = first * second;

        // Printing the result of the operation
        System.out.println("The product of " + first + " and " + second + " is

    }
}
```

In the example above, the program has been implemented so that the variables are declared first after which values are read into them. Variable declaration and the reading of values into them can also be combined into one.

```java
// Making the Scanner available to the program
import java.util.Scanner;

public class Program {
    public static void main(String[] main) {
        // Creating the Scanner
        Scanner reader = new Scanner(System.in);

        // Declaring the variables and assigning user input to them
        int first = Integer.valueOf(reader.nextLine());
        int second = Integer.valueOf(reader.nextLine());

        // Identifying the operation and declaring a variable for the result
        int product = first * second;

        // Printing the result of the operation
        System.out.println("The product of " + first + " and " + second + " is

    }
}
```

Programming exercise:

Squared

Points

1/1

Write a program that reads an integer from the user and prints the square of the given integer, i.e. the integer multiplied by itself.

Sample output

```
4
16
```

Sample output

```
-3
9
```

Sample output

```
5
25
```

Exercise submission instructions

How to see the solution

Programming exercise:

Square root of sum

Points

1/1

Write a program that reads two integers from the user and prints the square root of the sum of these integers. The program does not need to work with negative values.

You can calculate the square root of an integer with the command `Math.sqrt` like this:

```
int number = 42;
double squareRoot = Math.sqrt(number);
System.out.println(squareRoot);
```

Here are a few examples:

Sample output

```
1
0
1
```

Sample output

```
5
4
3
```

Sample output

```
1
35
6
```

Exercise submission instructions

How to see the solution

Conditional Logic

Problems often contain some conditional logic. In these instances we use conditional statements. A conditional statement starts with an `if` command followed by an expression in parentheses. The expression evaluates to either true or false. If it evaluates true, the following block delimited by curly brackets gets executed.

```
// if the value is greater than five
if (value > 5) {
    // then...
}
```

A program that prints "ok" if the value of the variable is greater than `42`, and otherwise prints "not ok" looks like this:

```
int value = 15;
if (value > 42) {
    System.out.println("ok");
} else {
    System.out.println("not ok")
}
```

You can also chain together multiple conditions. In such a case, the problem takes the form "if a, then b; else if c, then d; else if e, then f; otherwise g". The chain consists of an `if`-statement followed by `else if`-statements, each containing its own expression and a block.

```
// if the value is greater than five
if (value > 5) {
    // functionality when value is greater than five
} else if (value < 0) { //
    // functionality when value is less than zero
    // and the value IS NOT larger than five
} else {  // otherwise
    // functionality otherwise
}
```

System.out.print

The quiz below uses `System.out.print` for printing. It works exactly like `System.out.println`, but it doesn't add a line break to the end of the output.

Quiz:
Executing conditional statements

Points:
1/1

What does the program below print?

```
public static void main(String[] args) {
    int number = 12;

    if (number > 10) {
        System.out.print("Hello ");
    }  else if (number < 20) {
        System.out.print("world");
    } else {
        System.out.print("!");
    }
}
```

Select the correct answer

[✓ Hello]

world

!

Hello !

world!

Hello world!

The answer is correct

Answered Tries remaining: 1

Conditional logic can be combined with other patterns used for problem solving. Let's look into a problem "Read two integers from the user. If the sum of the integers is over 100, print `too much`. If the sum is less than 0, print `too little`. Otherwise, print `ok`. The program below combines reading, calculating and conditional functionality.

```
// Bringing Scanner to the program's use
import java.util.Scanner;
```

```java
public class Program {
    public static void main(String[] main) {
        // Creating the scanner
        Scanner reader = new Scanner(System.in);

        // Declaring the variables and assigning user input to them
        int first = Integer.valueOf(reader.nextLine());
        int second = Integer.valueOf(reader.nextLine());

        // Identifying the operation and declaring variable for the result
        int sum = first + second;

        // Doing something with the result. In this case
        // the result is used in the conditional operations.

        if (sum > 100) { // if the sum is over 100
            System.out.println("too much");
        } else if (sum < 0) { // if the sum is less than 0
            System.out.println("too little");
        } else { // otherwise
            System.out.println("ok");
        }
    }
}
```

Programming exercise:

Absolute Value

Points

1/1

Write a program that reads an integer from the user. If the number is less than 0, the program prints the given integer multiplied by -1. In all other cases, the program prints the number itself. A few examples of how the program's expected to function are shown below:

Sample output

```
-3
3
```

Sample output

```
2
```

```
2
```

Sample output

```
-5
5
```

Exercise submission instructions

How to see the solution

Programming exercise:

Comparing Numbers

Points

1/1

Write a program that reads two integers from the user. If the first number is greater than the second, the program prints "(first) is greater than (second)." If the first number is less than the second, the program prints "(first) is smaller than (second)." Otherwise, the program prints "(first) is equal to (second)." The (first) and (second) should always be replaced with the actual numbers that were provided by the user.

A few examples of the expected behaviour:

Sample output

```
8
4
8 is greater than 4.
```

Sample output

```
-3
5
-3 is smaller than 5.
```

Sample output

```
1
1
1 is equal to 1.
```

Exercise submission instructions

How to see the solution

You have reached the end of this section! Continue to the next section:

→ 2. Repeating functionality

Remember to check your points from the ball on the bottom-right corner of the material!

In this part: