Chapter 6
Methods II
COSC1046

Ping Luo

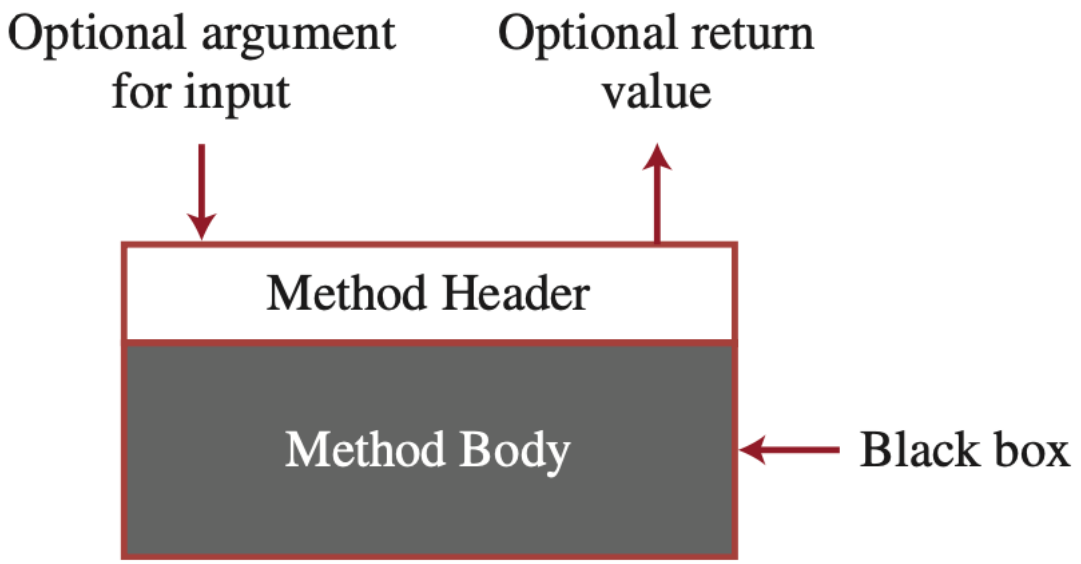
Content

1. Method II
2. Review for Chapters 1- 6
    1. key concepts
    2. example quiz questions
3. Exam start at 8:30pm

Method Abstraction and Stepwise Refinement

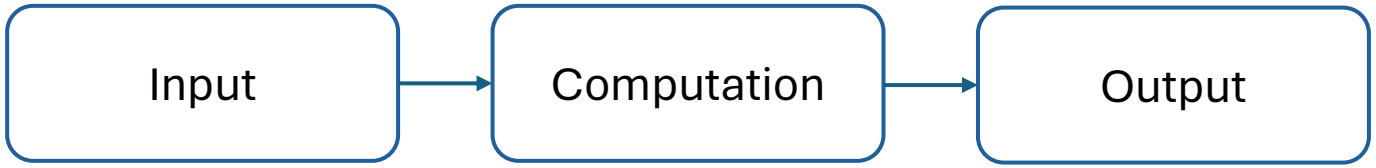
Method abstraction is achieved by separating the use of a method from its implementation.

- The client can use a method without knowing how it is implemented.
- If you decide to change the implementation, the client program will not be affected.



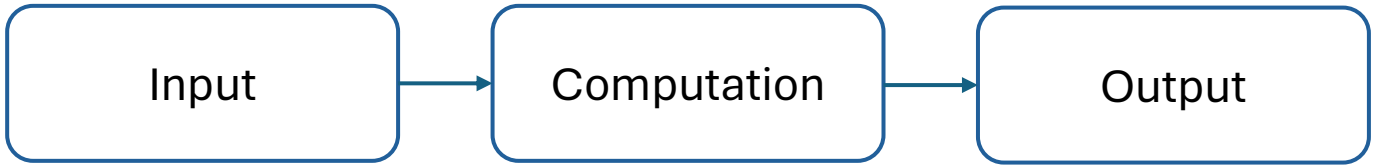
Method Abstraction

1. Write a method once and reuse it anywhere.
2. Information hiding. Hide the implementation from the user.
3. Reduce complexity.



Method Abstraction

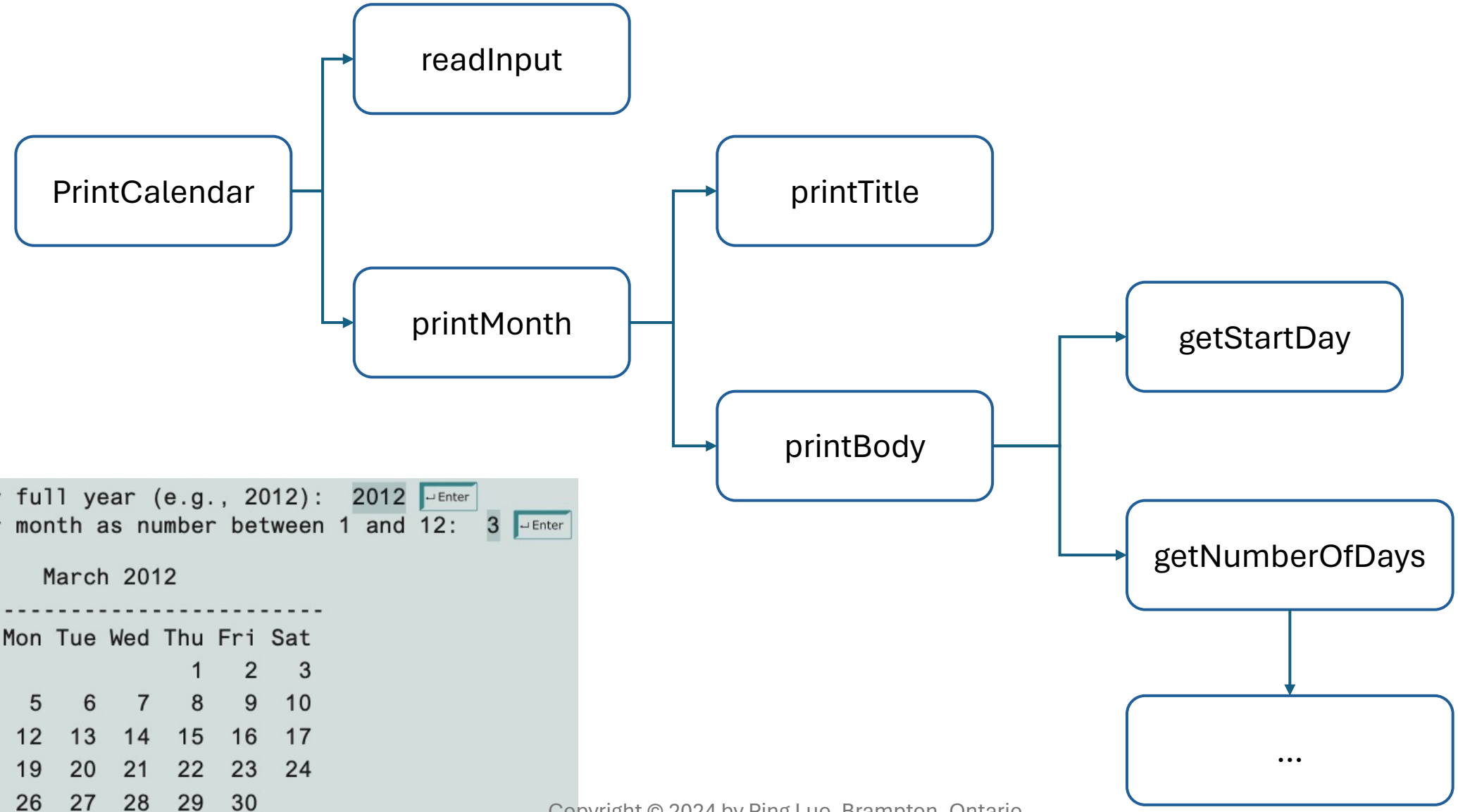
- The concept of method abstraction can be applied to the process of developing programs.
  - When writing a large program, you can use the divide-and-conquer strategy, also known as stepwise refinement, to **decompose it into subproblems**.
  - The subproblems can be further decomposed into smaller, more manageable problems.



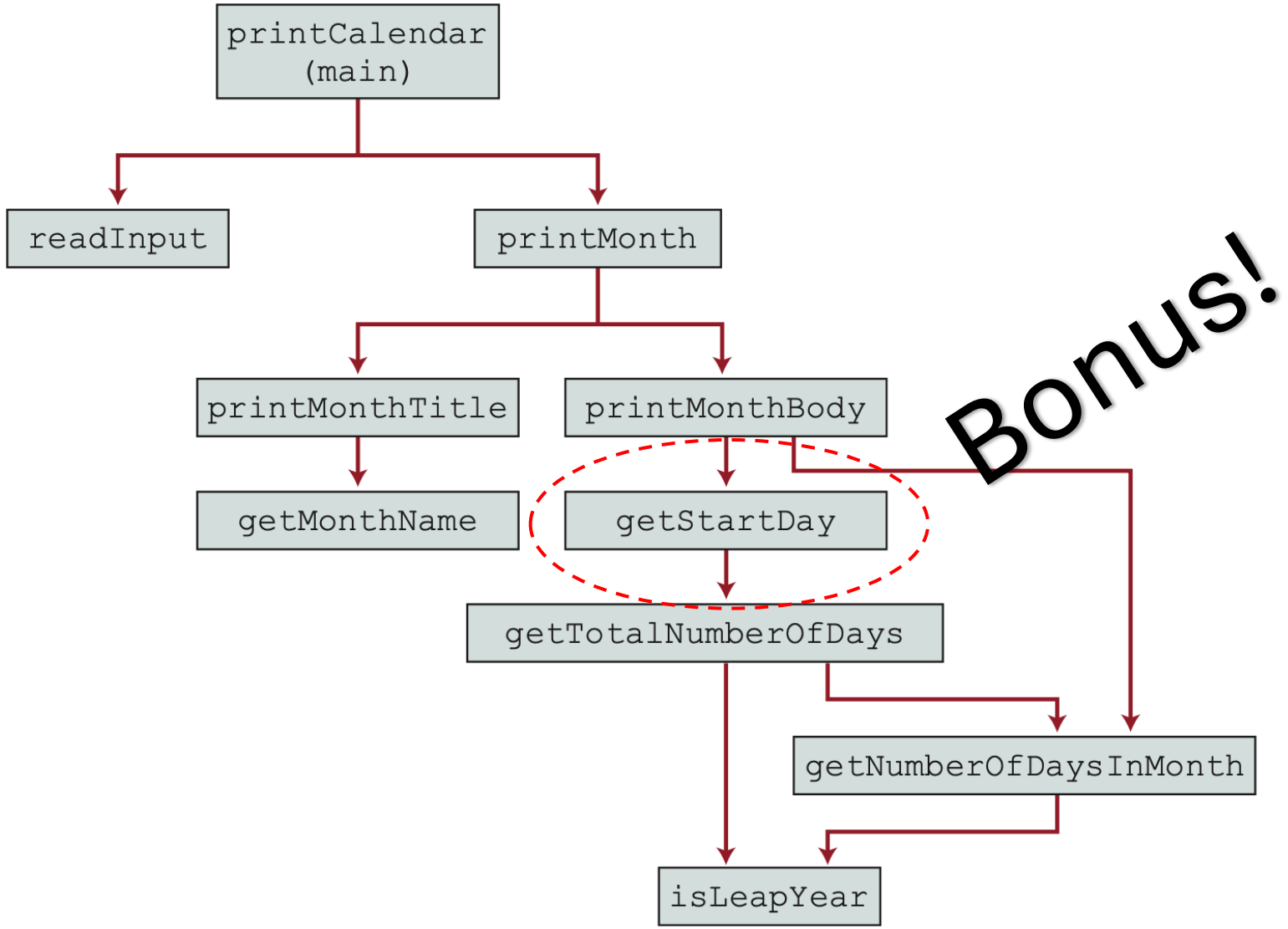
Print Calendar

- Write a program that displays the calendar for a given month of the year.
- The program prompts the user to enter the year and the month, and then displays the entire calendar for the month.

```
Enter full year (e.g., 2012):  2012 Enter
Enter month as number between 1 and 12:  3 Enter

        March 2012
-----------------------------
 Sun Mon Tue Wed Thu Fri Sat
                   1   2   3
   4   5   6   7   8   9  10
  11  12  13  14  15  16  17
  18  19  20  21  22  23  24
  25  26  27  28  29  30
```



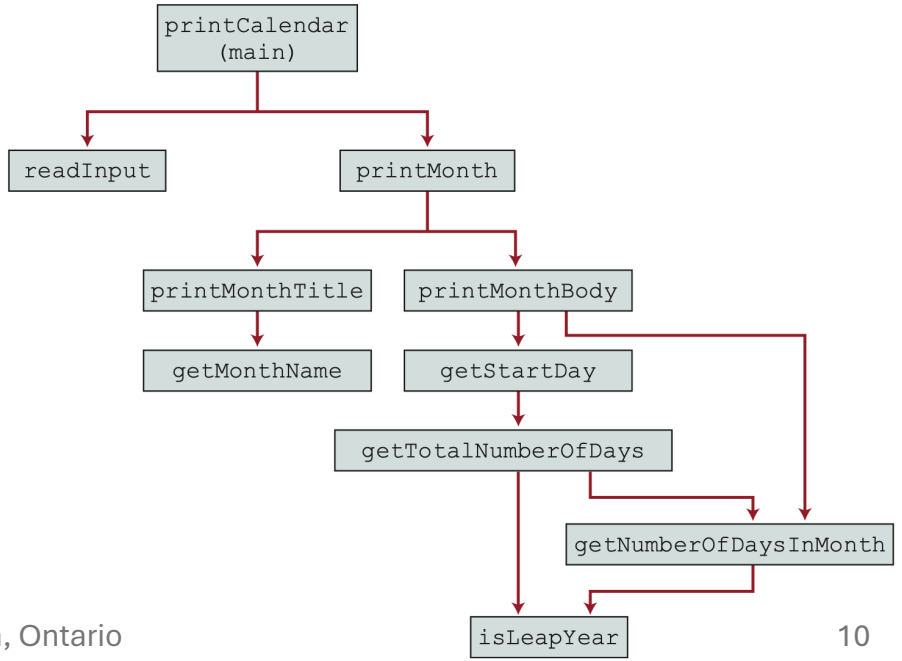

```
Enter full year (e.g., 2012):  2012 Enter
Enter month as number between 1 and 12:  3 Enter

        March 2012
-----------------------------
 Sun Mon Tue Wed Thu Fri Sat
                   1   2   3
   4   5   6   7   8   9  10
  11  12  13  14  15  16  17
  18  19  20  21  22  23  24
  25  26  27  28  29  30
```



```java
public static int getStartDay(int year, int month) {
    final int START_DAY_FOR_JAN_1_1800 = 3;
    int totalNumberOfDays = getTotalNumberOfDays(year, month);
    return (totalNumberOfDays + START_DAY_FOR_JAN_1_1800) % 7;
}
```

Benefits of Stepwise Refinement

- Simpler Program
- Reusing Methods
- Easier Developing, Debugging, and Testing
- Better Facilitating Teamwork



Review

- Chapter 1: Introduction to Computers, Programs and Java
- Chapter 2: Elementary Programming
- Chapter 3: Selections
- Chapter 4: Mathematical Functions, Characters and String
- Chapter 5: Loops
- Chapter 6: Methods

Chapter 1

- A program written in a high-level language is called a **source program** or **source code**. Because a computer cannot understand a source program, a source program must be translated into machine code for execution. The translation can be done using another programming tool called an **interpreter** or a **compiler**.
- Every statement in Java ends with a semicolon (;).

Chapter 2

- int i;
- double b;
- char a;
- final **datatype** CONSTANTNAME = VALUE;
  - final double PI = 3.1415926;
- nextDouble(), nextLine();
- 1.23456e-2 is equivalent to 0.0123456;
- int i = (int) 3.9

Example 1

- Which of the following statements is correct?

a) Every line in a program must end with a semicolon.
b) Every statement in a program must end with a semicolon.
c) Every comment line must end with a semicolon.
d) Every method must end with a semicolon.
e) Every class must end with a semicolon.

Example 2

- Which of the following is a constant, according to Java naming conventions?

a) MAX_VALUE
b) Test
c) read
d) ReadInt
e) COUNT

Chapter 3 - Conditional Statement

TABLE 3.1 Relational Operators

Java Operator	*Mathematics Symbol*	*Name*	*Example (radius is 5)*	*Result*
`<`	<	Less than	`radius < 0`	`false`
`<=`	≤	Less than or equal to	`radius <= 0`	`false`
`>`	>	Greater than	`radius > 0`	`true`
`>=`	≥	Greater than or equal to	`radius >= 0`	`true`
`==`	=	Equal to	`radius == 0`	`false`
`!=`	≠	Not equal to	`radius != 0`	`true`

Chapter 3 - Conditional Statement

```
if (boolean-expression) {
  statement(s);
}
```

```
if (boolean-expression) {
  statement(s)-for-the-true-case;
}
else {
  statement(s)-for-the-false-case;
}
```

```
switch (status) {
  case 0:  compute tax for single filers;
           break;
  case 1:  compute tax for married jointly or qualifying widow(er);
           break;
  case 2:  compute tax for married filing separately;
           break;
  case 3:  compute tax for head of household;
           break;
  default: System.out.println("Error: invalid status");
           System.exit(1);
}
```

Example 3

- What is y after the following switch statement is executed?

```
int x = 3;
int y = 4;
switch (x + 3) {
    case 6:  y = 0;
    case 7:  y = 1;
    default: y += 1;
}
```

a) 1

b) 2

c) 3

d) 4

e) 0

Chapter 4 - Mathematical Functions, Characters, and Strings

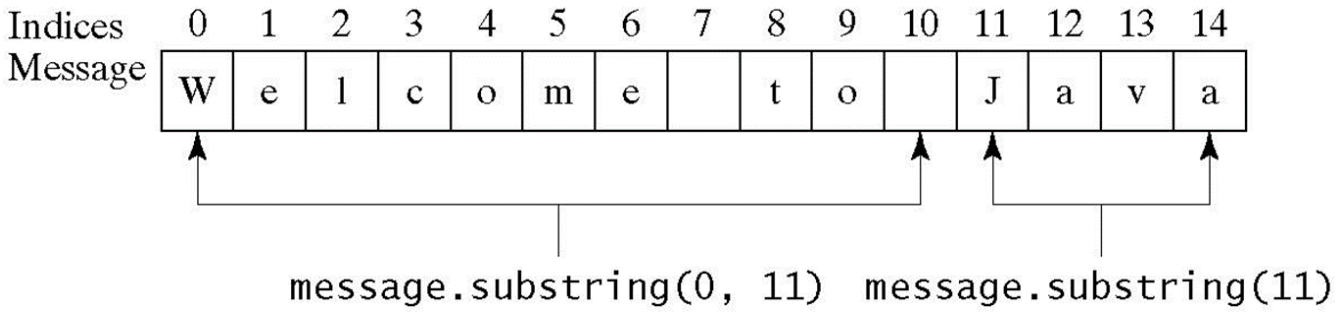
- Math.random() generates a random double value in the range [0.0, 1.0);

```
jshell> Math.rint(3.5)
$6 ==> 4.0

jshell> Math.round(3.5)
$7 ==> 4
```

Characters	*Code Value in Decimal*	*Unicode Value*
'0' to '9'	48 to 57	\u0030 to \u0039
'A' to 'Z'	65 to 90	\u0041 to \u005A
'a' to 'z'	97 to 122	\u0061 to \u007A

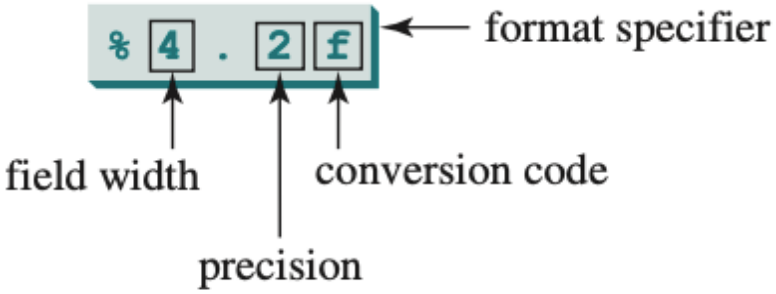
Chapter 4 - Methods for String Objects

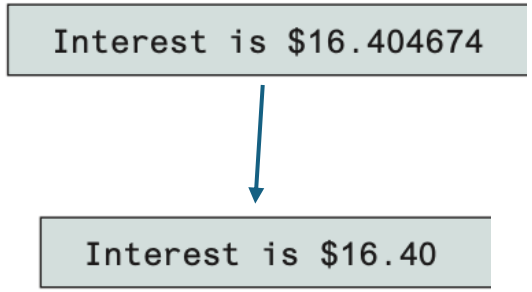
Method	Description
`length()`	Returns the number of characters in this string.
`charAt(index)`	Returns the character at the specified index from this string.
`concat(s1)`	Returns a new string that concatenates this string with string s1.
`toUpperCase()`	Returns a new string with all letters in uppercase.
`toLowerCase()`	Returns a new string with all letters in lowercase.
`trim()`	Returns a new string with whitespace characters trimmed on both sides.



Chapter 4 - Format Output

- System.out.printf(format, **items**);
- System.out.printf("Interest is $%4.2f", **interest**);






```
jshell> System.out.printf("%4.2f", 12345.5678)
12345.57$1 ==> java.io.PrintStream@66a29884
```

Example 4

- Which of the following is correct to obtain a random integer between 5 and 10?

a) 5 + Math.random() * 6

b) 5 + (int)(Math.random() * 6)

c) 5 + Math.random() * 5

d) 5 + (int)(Math.random() * 5)

Example 5

- Suppose s1 and s2 are two strings. What is the result of the following code?

- s1.equals(s2) == s2.equals(s1)

a) true
b) false

Chapter 5 - Loops

```
while (loop-continuation-condition) {
  // Loop body
  Statement(s);
}

do {
  // Loop body;
  Statement(s);
} while (loop-continuation-condition);

for (initial-action; loop-continuation-condition;
     action-after-each-iteration) {
  // Loop body;
  Statement(s);
}
```

```
public class TestBreak {
  public static void main(String[] args) {
    int sum = 0;
    int number = 0;

    while (number < 20) {
      number++;
      sum += number;
      if (sum >= 100)
        break;
    }

    System.out.println("The number is " + number);
    System.out.println("The sum is " + sum);
  }
}
```

Example 6

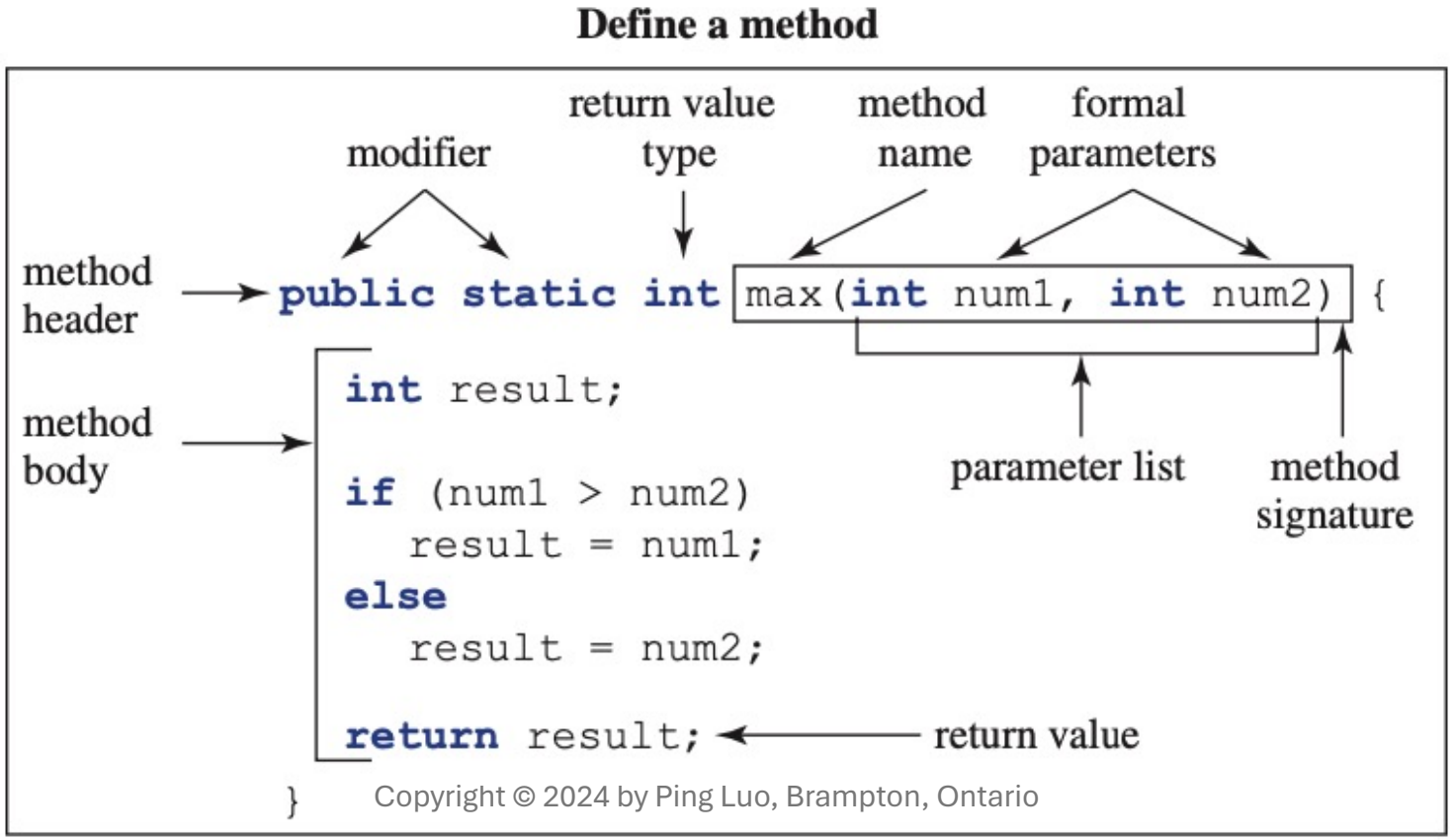
- How many times will the following code print "Welcome to Java"?

```
int count = 0;
do {
    System.out.println("Welcome to Java");
    count++;
} while (count < 10);
```

a) 8
b) 9
c) 10
d) 11
e) 0

Chapter 6 - Method

```
modifier returnValueType methodName(list of parameters) {
  // Method body;
}
```



Example 7

- Analyze the following code:

```
class Test {
  public static void main(String[] args) {
    System.out.println(xmethod(5));
  }
  public static int xmethod(int n, long t) {
    System.out.println("int");
      return n;
  }
  public static long xmethod(long n) {
      System.out.println("long");
      return n;
  }
}
```

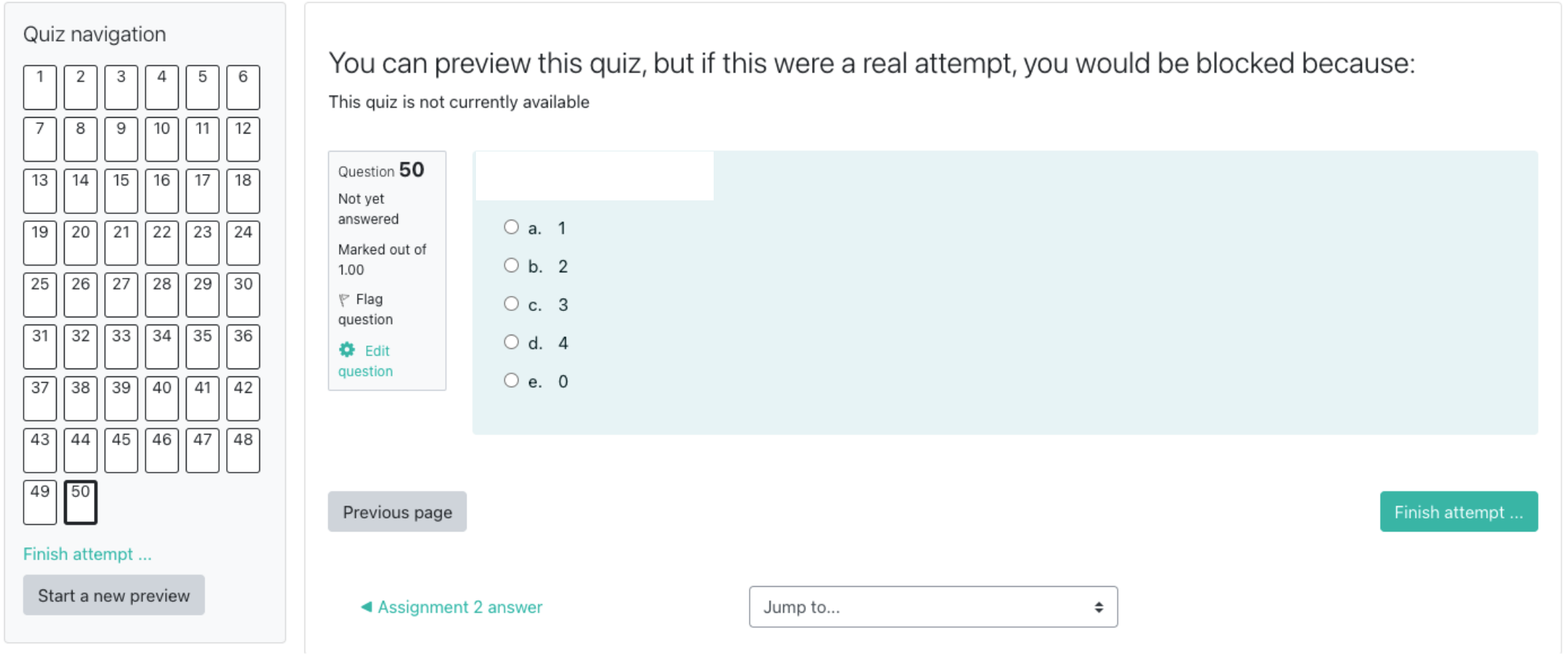
a) The program displays int followed by 5.

b) The program displays long followed by 5.

c) The program runs fine but displays things other than 5.

d) The program does not compile because the compiler cannot distinguish which xmethod to invoke.

Midterm Exam



Q&A