Advanced Programming

Java Basic 2

ThS. Trần Thị Thanh Nga
Khoa CNTT, Trường ĐH Nông Lâm TPHCM
Email: ngattt@hcmuaf.edu.vn

Selections

- When you compute area, if you enter a negative value:
  - the program prints an invalid result.
  - you don't want the program to compute the area.
- How can you deal with this situation?

```
if (radius < 0)
        System.out.println("Incorrect input");
else {
        area = radius * radius * 3.14159;
        System.out.println("Area is " + area);
}
```

boolean Data Type

- How do you compare two values, such as whether a radius is greater than 0, equal to 0, or less than 0?

Comparison Operators

Operator	*Name*	*Example*	*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`

Problem: A Simple Math Learning Tool

- Develop a program to let a first-grader practice addition.
- The program randomly generates two single-digit integers, **number1** and **number2**, and displays to the student a question such as “What is 7 + 9?”.
- After the student types the answer, the program displays a message to indicate whether it is *true* or *false*.

Solution

- Bước1: Tạo ra 2 số ngẫu nhiên n1, n2
- Bước 2: Đưa ra thông báo: n1 + n2 =?
- Bước 3: Người dùng nhập vào kết quả, ví dụ result
- Bước 4: So sánh kết quả của n1+n2 với result
- Bước 5: Hiển thị thông báo: đúng hoặc sai

Problem: A Simple Math Learning Tool

```
public class AdditionQuiz {
   public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        int number1 = (int) (System.currentTimeMillis() * 7 % 10);
        int number2 = (int) (System.currentTimeMillis() % 10);
        System.out.print("What is " + number1 + " + " + number2 +
                                                  "? ");
        int answer = input.nextInt();
        System.out.println(number1 + " + " + number2 + " = " +
                answer + " is “ + (number1 + number2 == answer));
   }
}
```

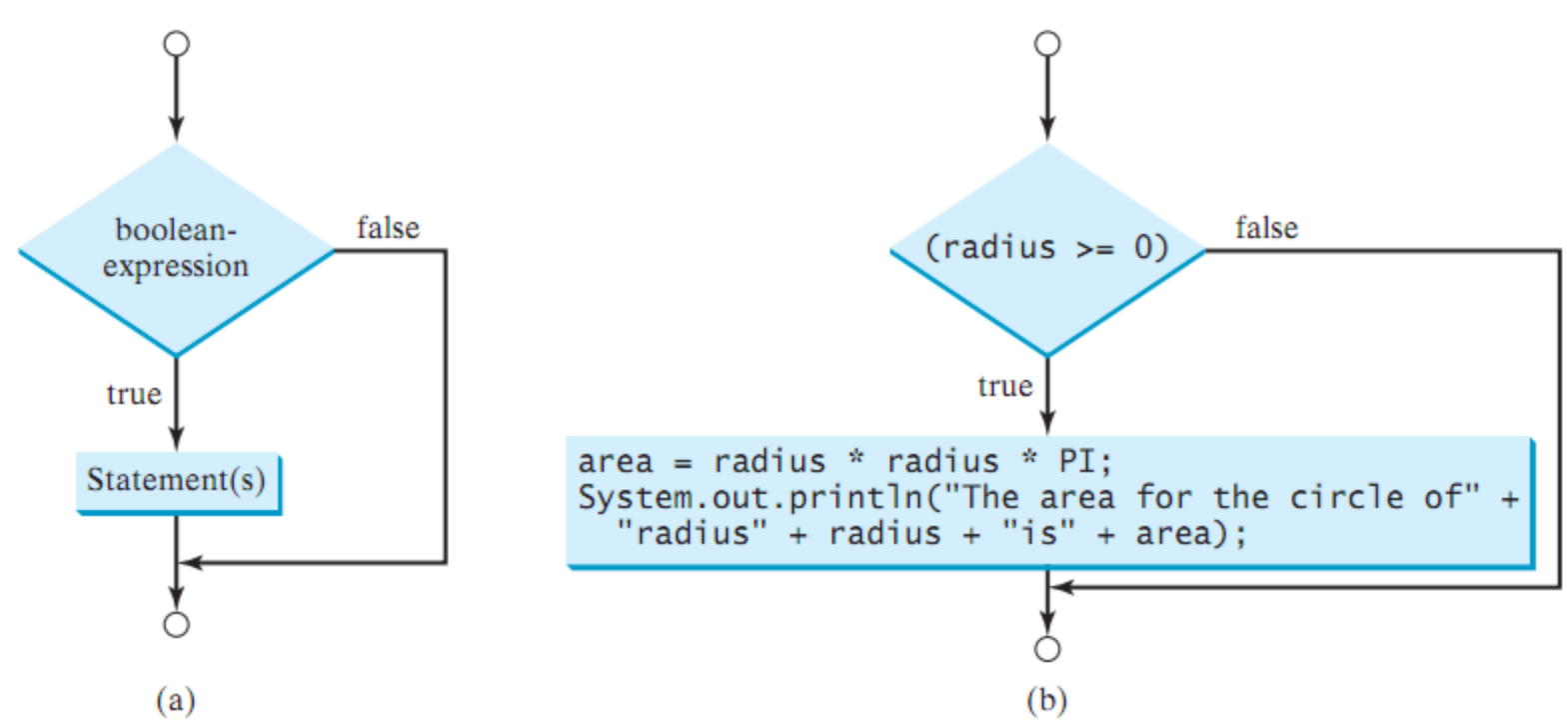
if Statements

- one-way **if** statements,
- two-way **if** statements,
- nested **if** statements,
- **switch** statements,
- and conditional expressions.

One-Way if Statements

- A *one-way* **if** statement executes an action if and only if the condition is **true**.

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

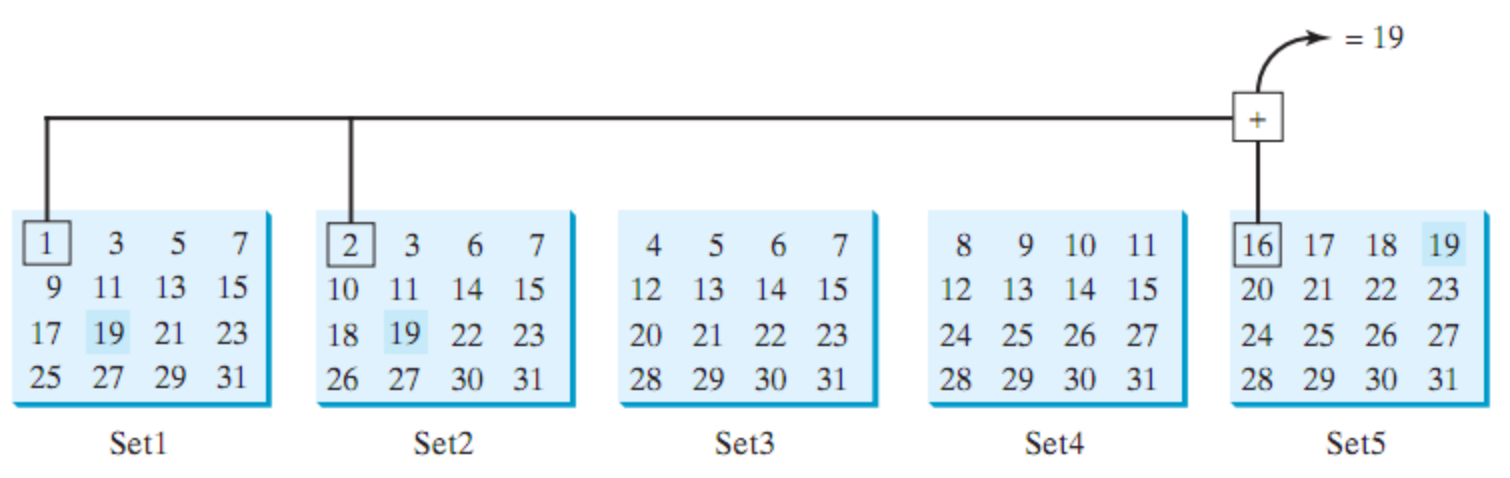


One-Way if Statements

```
public class SimpleIfDemo {
   public static void main(String[] args) {
       Scanner input = new Scanner(System.in);
       System.out.print("Enter an integer: ");
       int number = input.nextInt();
       if (number % 5 == 0)
             System.out.println("HiFive");

       if (number % 2 == 0)
             System.out.println("HiEven");
   }
}
```

Problem: Guessing Birthdays

- You can find out the date of the month when your friend was born by asking five questions.
- Each question asks whether the day is in one of the five sets of numbers.



Problem: Guessing Birthdays

```
public class GuessBirthday {
   public static void main(String[] args) {
       String set1 = " 1  3  5  7\n" + " 9 11 13 15\n"
                     + "17 19 21 23\n"+ "25 27 29 31";
       String set2 = " 2  3  6  7\n" + "10 11 14 15\n"
                     + "18 19 22 23\n"+ "26 27 30 31";
       String set3 = " 4  5  6  7\n" + "12 13 14 15\n"
                     + "20 21 22 23\n" + "28 29 30 31";
       String set4 = " 8  9 10 11\n" + "12 13 14 15\n"
                     + "24 25 26 27\n" + "28 29 30 31";
       String set5 = "16 17 18 19\n" + "20 21 22 23\n"
                     + "24 25 26 27\n"+ "28 29 30 31";
       int day = 0;
       // Create a Scanner
       Scanner input = new Scanner(System.in);
```

Problem: Guessing Birthdays

```
// Prompt the user to answer questions
System.out.print("Is your birthday in Set1?\n");
System.out.print(set1);
System.out.print("\nEnter 0 for No and 1 for Yes: ");
int answer = input.nextInt();

if (answer == 1)
        day += 1;
// Prompt the user to answer questions
System.out.print("\nIs your birthday in Set2?\n");
System.out.print(set2);
System.out.print("\nEnter 0 for No and 1 for Yes: ");
answer = input.nextInt();
if (answer == 1)
        day += 2;
```

Problem: Guessing Birthdays

```
// Prompt the user to answer questions
System.out.print("Is your birthday in Set3?\n");
System.out.print(set3);
System.out.print("\nEnter 0 for No and 1 for Yes: ");
answer = input.nextInt();
if (answer == 1)
        day += 4;

// Prompt the user to answer questions
System.out.print("\nIs your birthday in Set4?\n");
System.out.print(set4);
System.out.print("\nEnter 0 for No and 1 for Yes: ");
answer = input.nextInt();
if (answer == 1)
        day += 8;
```

Problem: Guessing Birthdays

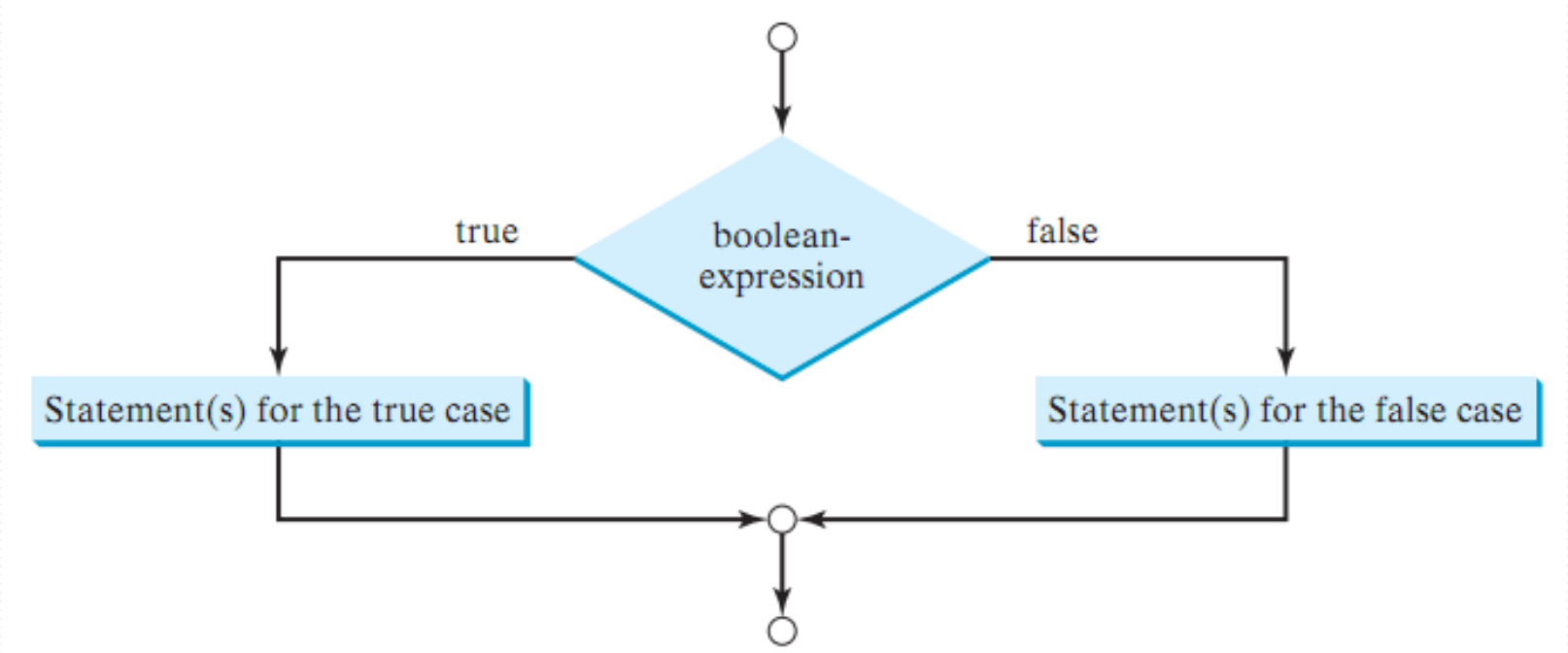
```
        // Prompt the user to answer questions
        System.out.print("\nIs your birthday in Set5?\n");
        System.out.print(set5);
        System.out.print("\nEnter 0 for No and 1 for Yes: ");
        answer = input.nextInt();

        if (answer == 1)
                day += 16;
        System.out.println("\nYour birthday is " + day + "!");
    }
}
```

Two-Way if Statements

- **Syntax**:

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

Two-Way if Statements



Two-Way if Statements

```
if (radius >= 0) {
        area = radius * radius * PI;
        System.out.println("The area for the   circle of
                radius " + radius + " is " + area);
} else {
        System.out.println("Negative input");
}
```

Nested if Statements

- The inner **if** statement is said to be nested inside the outer **if** statement.
- The inner **if** statement can contain another **if** statement;
- There is no limit to the depth of the nesting.

Nested if Statements

```
if (i > k) {
      if (j > k)
            System.out.println("i and j are
                        greater than k");
} else
      System.out.println("i is less than or
                        equal to k");
```

Nested if Statements

```
if (score >= 90.0)
  grade = 'A';
else
  if (score >= 80.0)
    grade = 'B';
  else
    if (score >= 70.0)
      grade = 'C';
    else
      if (score >= 60.0)
        grade = 'D';
      else
        grade = 'F';
```

(a)

Equivalent

This is better

```
if (score >= 90.0)
  grade = 'A';
else if (score >= 80.0)
  grade = 'B';
else if (score >= 70.0)
  grade = 'C';
else if (score >= 60.0)
  grade = 'D';
else
  grade = 'F';
```

(b)

Logical Operators

- Whether a statement is executed is determined by a combination of several conditions ➔ use *logical operators* to combine them.
- *Logical operators*, also known as *Boolean operators*, operate on Boolean values to create a new Boolean value.

Logical Operators

TABLE 3.3 Boolean Operators

Operator	*Name*	*Description*
!	not	logical negation
&&	and	logical conjunction
||	or	logical disjunction
^	exclusive or	logical exclusion

Logical Operators

TABLE 3.4 Truth Table for Operator `!`

p	*!p*	*Example (assume age = 24, gender = 'F')*
`true`	`false`	`!(age > 18)` is `false`, because `(age > 18)` is `true`.
`false`	`true`	`!(gender == 'M')` is `true`, because `(gender == 'M')` is `false`.

TABLE 3.5 Truth Table for Operator `&&`

p1	*p2*	*p1 && p2*	*Example (assume* age = 24, gender = 'F'*)*
`false`	`false`	`false`	`(age > 18) && (gender == 'F')` is `true`, because `(age > 18)` and `(gender == 'F')` are both `true`.
`false`	`true`	`false`	
`true`	`false`	`false`	`(age > 18) && (gender != 'F')` is `false`, because `(gender != 'F')` is `false`.
`true`	`true`	`true`	

Logical Operators

TABLE 3.6 Truth Table for Operator `||`

p1	*p2*	*p1* || *p2*	*Example (assume* age = 24, gender = 'F'*)*
`false`	`false`	`false`	`(age > 34) || (gender == 'F')` is `true`, because `(gender == 'F')` is `true`.
`false`	`true`	`true`	
`true`	`false`	`true`	`(age > 34) || (gender == 'M')` is `false`, because `(age > 34)` and `(gender == 'M')` are both `false`.
`true`	`true`	`true`	

TABLE 3.7 Truth Table for Operator `^`

p1	*p2*	*p1* ^ *p2*	*Example (assume* age = 24, gender = 'F'*)*
`false`	`false`	`false`	`(age > 34) ^ (gender == 'F')` is `true`, because `(age > 34)` is `false` but `(gender == 'F')` is `true`.
`false`	`true`	`true`	
`true`	`false`	`true`	`(age > 34) || (gender == 'M')` is `false`, because `(age > 34)` and `(gender == 'M')` are both `false`.
`true`	`true`	`false`	

Logical Operators

```
public class TestBooleanOperators {
   public static void main(String[] args) {
      Scanner input = new Scanner(System.in);
      // Receive an input
      System.out.print("Enter an integer: ");
      int number = input.nextInt();
      System.out.println("Is " + number + "\n\t divisible
      by 2 and 3? "
            + (number % 2 == 0 && number% 3 == 0)
            + "\n\t divisible by 2 or 3? "
            + (number % 2 == 0 || number % 3 == 0)
            + "\n\t divisible by 2 or 3, but not both? "
            + (number % 2 == 0 ^ number % 3 == 0));
   }
}
```

Logical Operators

```
Enter an integer: 18 ↵Enter
Is 18
  divisible by 2 and 3? true
  divisible by 2 or 3? true
  divisible by 2 or 3, but not both? false
```

Determining Leap Year

- **Finger exercise**: A year is a *leap year* if it is divisible by 4 but not by 100 or if it is divisible by 400. So you can use the following Boolean expressions to check whether a year is a leap year:

```
// A leap year is divisible by 4
boolean isLeapYear = (year % 4 == 0);

// A leap year is divisible by 4 but not by 100
isLeapYear = isLeapYear && (year % 100 != 0);

// A leap year is divisible by 4 but not by 100 or divisible by 400
isLeapYear = isLeapYear || (year % 400 == 0);
```

Determining Leap Year

```
public class LeapYear {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.print("Enter a year: ");
        int year = input.nextInt();
        // Check if the year is a leap year
        boolean isLeapYear = (year % 4 == 0 && year % 100
                != 0)|| (year % 400 == 0);
        // Display the result
        System.out.println(year + " is a leap
        year? " + isLeapYear);
    }
}
```

Problem: Lottery

```
public class Lottery {
   public static void main(String[] args) {
      // Generate a lottery
      int lottery = (int) (Math.random() * 100);
      // Prompt the user to enter a guess
      Scanner input = new Scanner(System.in);
      System.out.print("Enter your lottery pick (two
                        digits): ");
      int guess = input.nextInt();
      // Get digits from lottery
      int lotteryDigit1 = lottery / 10;
      int lotteryDigit2 = lottery % 10;
      // Get digits from guess
      int guessDigit1 = guess / 10;
      int guessDigit2 = guess % 10;
```

Problem: Lottery

```
        System.out.println("The lottery number is " + lottery);
        // Check the guess
        if (guess == lottery)
                System.out.println("Exact match: you win $10,000");
        else if (guessDigit2 == lotteryDigit1
                                && guessDigit1 == lotteryDigit2)
                System.out.println("Match all digits: you win $3,000");
        else if (guessDigit1 == lotteryDigit1
                                || guessDigit1 == lotteryDigit2
                                || guessDigit2 == lotteryDigit1
                                || guessDigit2 == lotteryDigit2)
                System.out.println("Match one digit: you win $1,000");
        else
                System.out.println("Sorry, no match");
    }
}
```

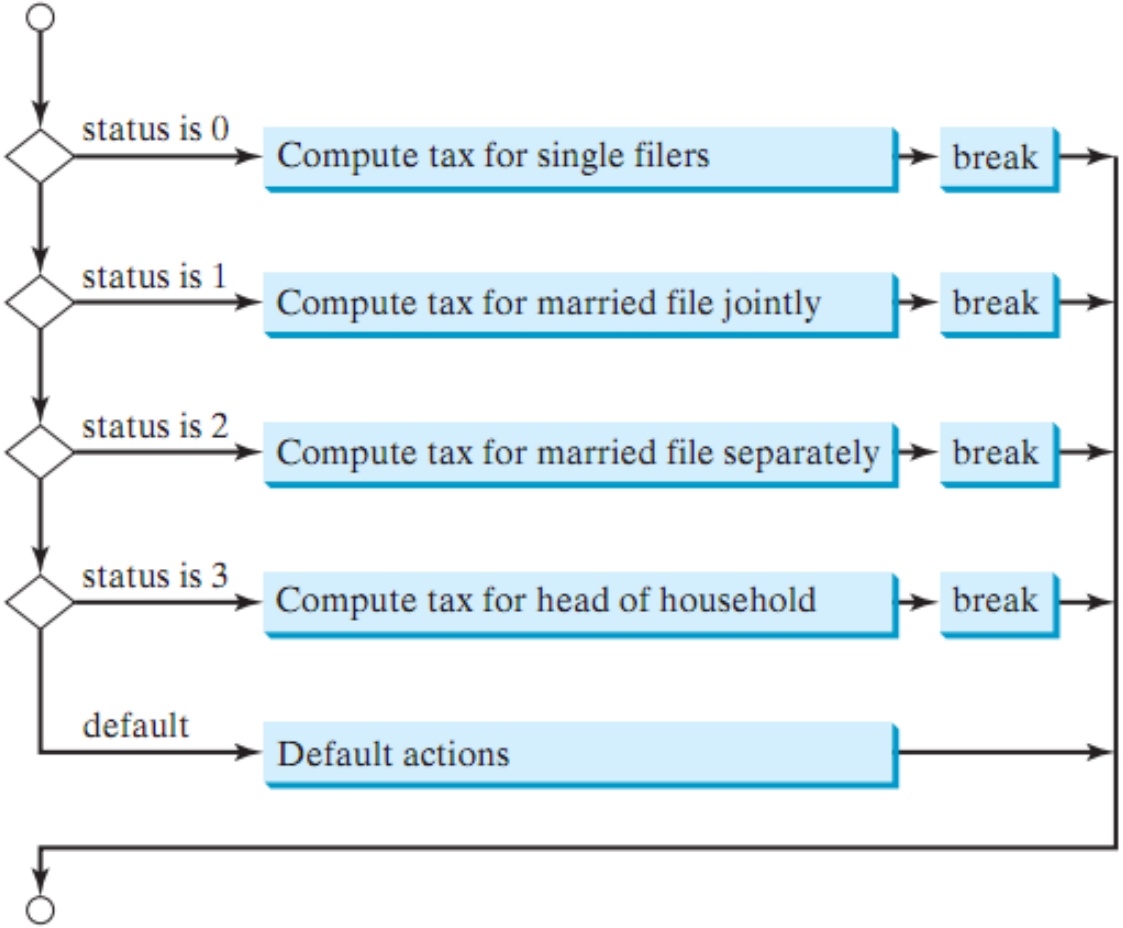
switch Statements

- Overuse of nested if statements makes a program difficult to read.
- Java provides a **switch** statement to handle multiple conditions efficiently.

switch Statements

```
switch (status) {
        case 0: // compute taxes for single filers;
                break;
        case 1: // compute taxes for married filing jointly;
                break;
        case 2: // compute taxes for married filing separately;
                break;
        case 3: // compute taxes for head of household;
                break;
        default:
                System.out.println("Errors: invalid status");
        System.exit(0);
}
```

switch Statements



switch syntax

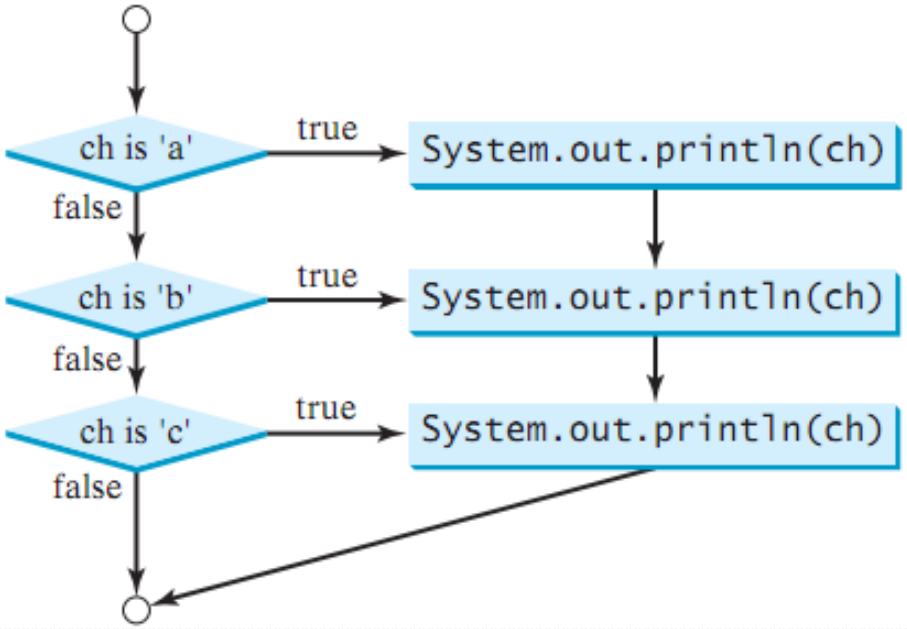
```
switch (switch_expression) {
      case value1: statement(s)1;
            break;
      case value2: statement(s)2;
            break;
       ...
      case valueN: statement(s)N;
            break;
      default:
            statement(s)_for_default;
}
```

Rules

- The **switch_expression** must yield a value of **char**, **byte**, **short**, or **int** type and must always be enclosed in parentheses.
- The **value1**,…, and **valueN** must have the same data type as the value of the **switch_expression**.
  - Note : **value1**, and **valueN** are constant expressions.
- When the value in a **case** statement matches the value of the **switch-expression**, the statements *starting from this case* are executed until either a **break** statement or the end of the switch statement is reached.
- The **break** statement immediately ends the **switch** statement.
- The **default** case can be used to perform actions when none of the specified cases matches the **switch-expression**.
- The **case** statements are checked in sequential order, but the order of the cases (including the default case) does not matter.

without **break**

```
switch (ch) {
  case 'a': System.out.println(ch);
  case 'b': System.out.println(ch);
  case 'c': System.out.println(ch);
}
```



Conditional Expressions

- Use a conditional expression:

```
y = (x > 0)? 1:-1;
d1 = (a1>a2)?a1:a2
```

- Equivalent:

```
if (x > 0)
    y = 1;
else
    y = -1;
```

Conditional Expressions

- Conditional expressions are in a completely different style, with no explicit **if** in the statement.
- **Syntax** :

  boolean-expression ? expression1 : expression2;

- The result of this conditional expression is **expression1** if **boolean-expression** is **true**; otherwise the result is **expression2**.

Formatting Console Output

- To format the output using the **printf** method.

  System.out.**printf**(format, item1, item2, ..., itemk)

  - where **format** is a string that may consist of substrings and format *specifiers*.

```
int count = 5;
double amount = 45.56;
System.out.printf("count is %d and amount is %f", count, amount);
```

items

```
display            count is 5 and amount is 45.560000
```

Formatting Console Output

Frequently Used Specifiers

Specifier	*Output*	*Example*
%b	a Boolean value	true or false
%c	a character	'a'
%d	a decimal integer	200
%f	a floating-point number	45.460000
%e	a number in standard scientific notation	4.556000e+01
%s	a string	"Java is cool"

Introduction

- You need to print a string (e.g., "**Welcome to Java!**") a hundred times.

100 times

```
System.out.println("Welcome to Java!");
System.out.println("Welcome to Java!");
...
System.out.println("Welcome to Java!");
```

Introduction

```
int count = 0;
while (count < 100) {
      System.out.println("Welcome to Java!");
      count++;
}
```

The **while** Loop

- **Syntax:**

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

The **while** Loop

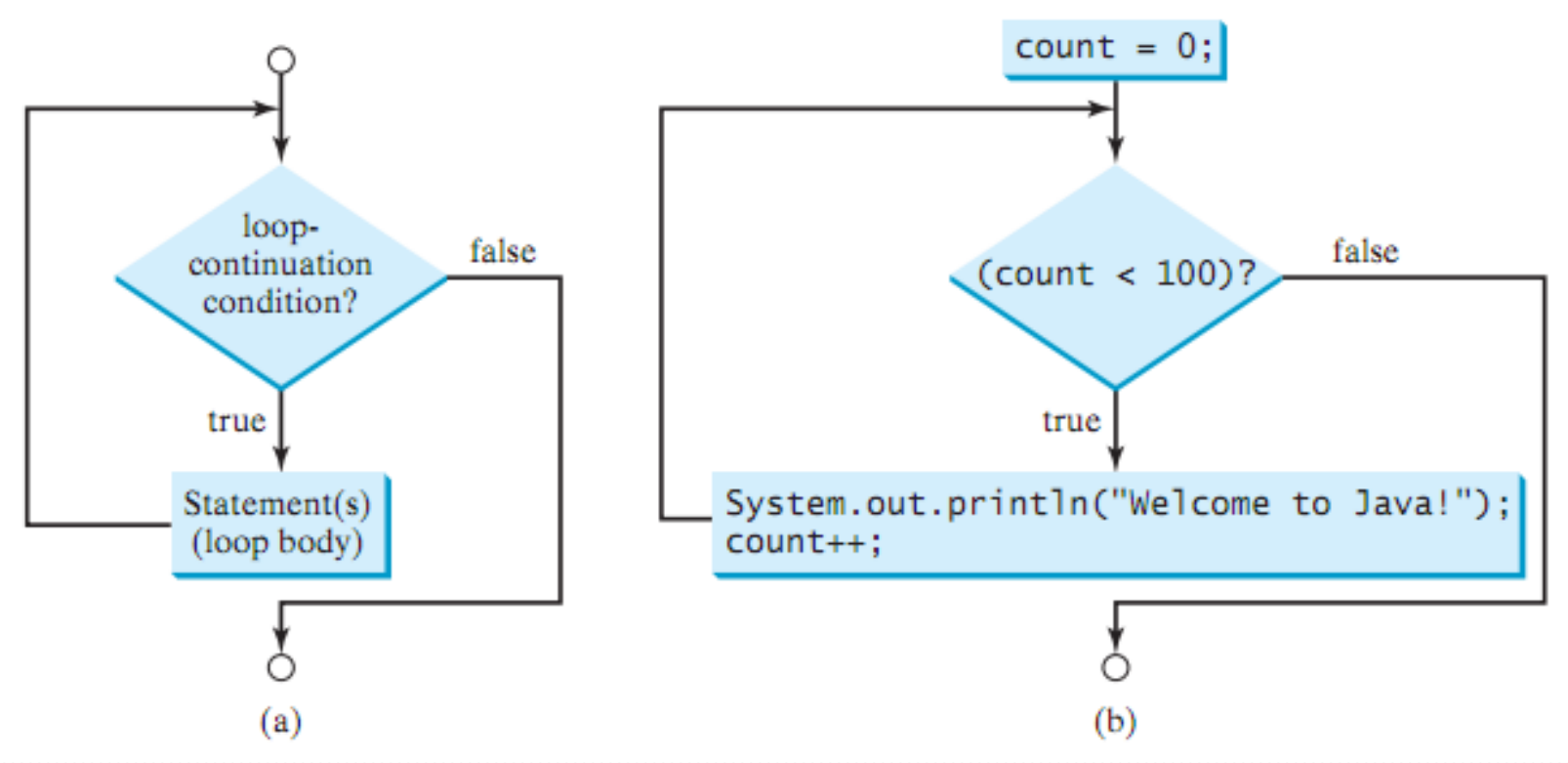
- The part of the loop that contains the statements to be repeated is called the *loop body*.
- A one-time execution of a loop body is referred to as an *iteration of the loop*.
- Each loop contains a *loop-continuation-condition*, a Boolean expression that controls the execution of the body.

loop-continuation-condition

```
int count = 0;
while (count < 100) {
  System.out.println("Welcome to Java!");
  count++;
}
```

loop body

The **while** Loop



The **while** Loop

```
int sum = 0, i = 1;
while (i < 10) {
      sum = sum + i;//1
      i++;
}
System.out.println("sum is " + sum);// sum is 45
```

The **while** Loop

- What happens if the loop is mistakenly written as follows:

```
int sum = 0, i = 1;
while (i < 10) {
    sum = sum + i;
}
```

Problem: Guessing Numbers

- Randomly generates an integer between 0 and 100.
- The program prompts the user to enter a number continuously until the number matches the randomly generated number.
- For each user input, the program tells the user whether the input is *too low* or *too high*, so the user can make the next guess intelligently.

Problem: Guessing Numbers

```
Guess a magic number between 0 and 100

Enter your guess: 50 ↵Enter
Your guess is too high

Enter your guess: 25 ↵Enter
Your guess is too high

Enter your guess: 12 ↵Enter
Your guess is too high

Enter your guess: 6 ↵Enter
Your guess is too low

Enter your guess: 9 ↵Enter
Yes, the number is 9
```

Problem: Guessing Numbers

```
public class GuessNumberOneTime {
   public static void main(String[] args) {
      // Generate a random number to be guessed
      int number = (int)(Math.random() * 101);
      Scanner input = new Scanner(System.in);
      System.out.println("Guess a magic number between 0 and 100");
      // Prompt the user to guess the number
      System.out.print("\nEnter your guess: ");
      int guess = input.nextInt();
      if (guess == number)
            System.out.println("Yes, the number is " + number);
      else if (guess > number)
            System.out.println("Your guess is too high");
      else
            System.out.println("Your guess is too low");
      }
   }
```

Problem: Guessing Numbers

```
public class GuessNumber {
   public static void main(String[] args) {
         int number = (int) (Math.random() * 101);
         Scanner input = new Scanner(System.in);
         System.out.println("Guess a magic number between 0 and 100");
         int guess = -1;
         while (guess != number) {
                  System.out.print("\nEnter your guess: ");
                  guess = input.nextInt();
                  if (guess == number)
                           System.out.println("Yes, the number is " + number);
                  else if (guess > number)
                           System.out.println("Your guess is too high");
                  else
                           System.out.println("Your guess is too low");
                  // End of loop
         }
   }
}
```

Loop Design Strategies

- **Step 1**: Identify the statements that need to be repeated.
- **Step 2**: Wrap these statements in a loop like this:

```
while (true) {
        Statements;
}
```

- **Step 3**: Code the *loop-continuation-condition* and add appropriate statements for controlling the loop.

```
while (loop-continuation-condition) {
        Statements;
        Additional statements for controlling the loop;
}
```

Problem: An Advanced Math Learning Tool

- How do you write the code to generate five questions?
  - **First** identify the statements that need to be repeated. These are the statements for obtaining two random numbers, prompting the user with a subtraction question, and grading the question.
  - **Second**, wrap the statements in a loop.
  - **Third**, add a *loop control variable* and the *loop-continuation-condition* to execute the loop five times.

Problem: An Advanced Math Learning Tool

```
public class SubtractionQuizLoop {
   public static void main(String[] args) {
        final int NUMBER_OF_QUESTIONS = 5; // Number of questions
        int correctCount = 0; // Count the number of correct answers
        int count = 0; // Count the number of questions
        long startTime = System.currentTimeMillis();
        String output = ""; // output string is initially empty
        Scanner input = new Scanner(System.in);
        while (count < NUMBER_OF_QUESTIONS) {
                // 1. Generate two random single-digit integers
                int number1 = (int) (Math.random() * 10);
                int number2 = (int) (Math.random() * 10);
                // 2. If number1 < number2, swap number1 with number2
                if (number1 < number2) {
                        int temp = number1;
                        number1 = number2;
                        number2 = temp;
                }
```

```java
            // 3. Prompt the student to answer "What is number1-number2?"
            System.out.print("What is " + number1 + " - " + number2 + "? ");
            int answer = input.nextInt();
            // 4. Grade the answer and display the result
            if (number1 - number2 == answer) {
            System.out.println("You are correct!");
            correctCount++;
            } else
                 System.out.println("Your answer is wrong.\n" + number1 + " -
                         "+ number2 + " should be " + (number1 - number2));
            // Increase the count
            count++;
            output += "\n" + number1 + "-" + number2 + "=" + answer +
                 ((number1 - number2 == answer) ? " correct" : " wrong");
            long endTime = System.currentTimeMillis();
            long testTime = endTime - startTime;
            System.out.println("Correct count is " + correctCount
             + "\nTest time is " + testTime / 1000 + " seconds\n"+ output);
        } //end while
    }
}
```

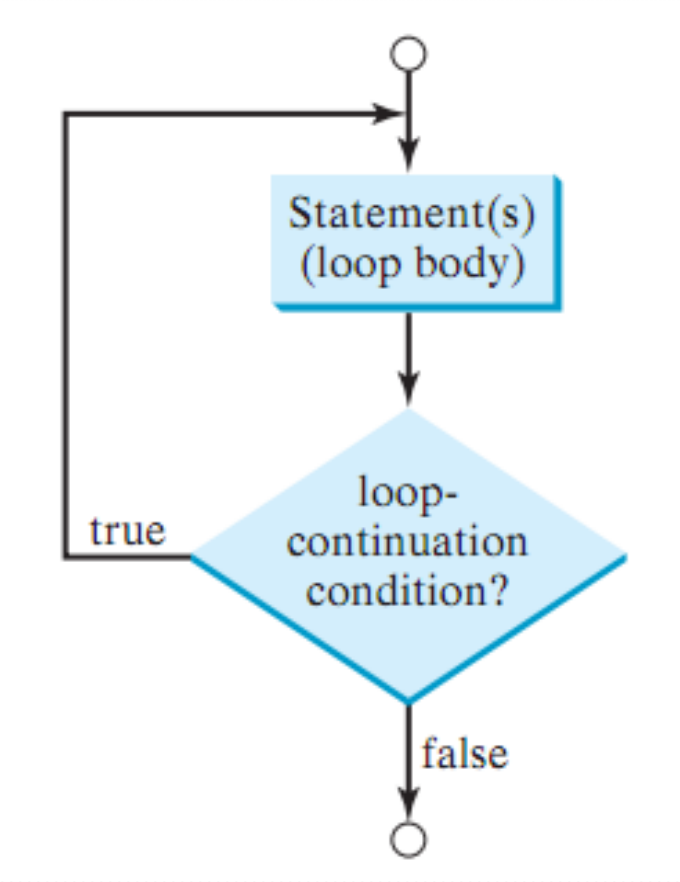
The do-while Loop

- **Syntax**:

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

- The **do-while** loop executes the loop body first, then checks the *loop-continuation-condition* to determine whether to *continue* or *terminate* the loop.

The **do-while** Loop



```java
public class TestDoWhile {
   public static void main(String[] args) {
      int data, sum = 0;
      // Create a Scanner
      Scanner sc = new Scanner(System.in);
      // Keep reading data until the input is 0
      do {
            // Read the next data
            System.out.print("Enter an int value (the
                  program exits if the input is 0): ");
            data = sc.nextInt();
            sum += data;
      } while (data != 0);
      System.out.println("The sum is " + sum);
   }
}
```

The for Loop

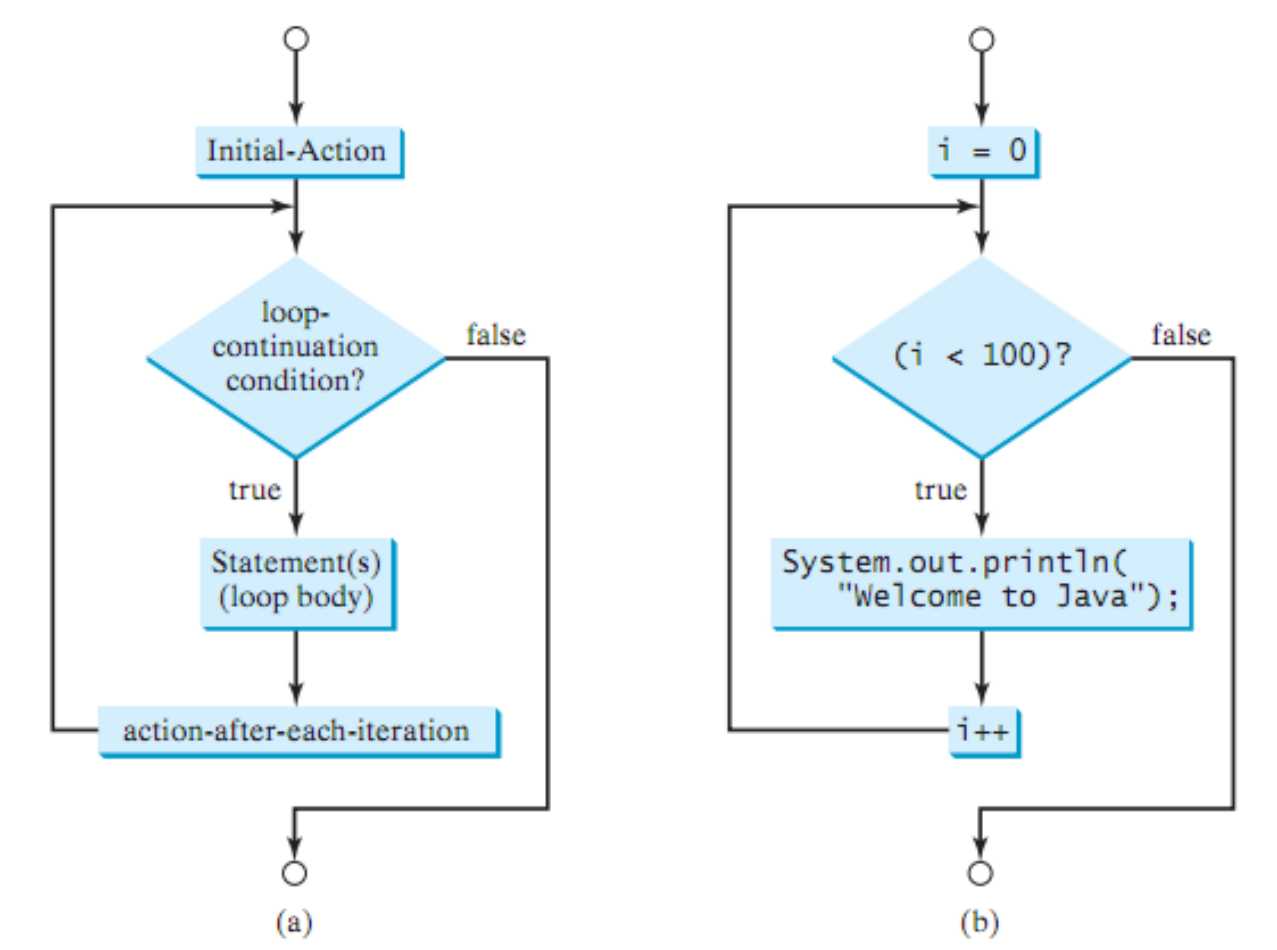
- Syntax:

```
for (i = initialValue; i < endValue; i++) {
    // Loop body
    Statement(s);
}
```

- Example:

```
for (int i = 0; i < 100; i++) {
    System.out.println("Welcome to Java!");
}
```

The for Loop


Initial-Action
loop-continuation condition?
false
true
Statement(s) (loop body)
action-after-each-iteration
(a)
i = 0
(i < 100)?
false
true
System.out.println("Welcome to Java");
i++
(b)

The for Loop

```
for ( ; ; ) {
  // Do something
}
```

(a)

Equivalent

```
for ( ; true; ) {
  // Do something
}
```

(b)

Equivalent

This is better

```
while (true) {
  // Do something
}
```

(c)

The for Loop

Error

```
for (int i = 0; i < 10; i++);
{
  System.out.println("i is " + i);
}
```

(a)

Empty Body

```
for (int i = 0; i < 10; i++)  { };
{
  System.out.println("i is " + i);
}
```

(b)

Error

```
int i = 0;
while (i < 10);
{
  System.out.println("i is " + i);
  i++;
}
```

(c)

Empty Body

```
int i = 0;
while (i < 10) { };
{
  System.out.println("i is " + i);
  i++;
}
```

(d)

Nested Loops

- Nested loops consist of an **outer loop** and one or **more inner loops**.
- Each time the outer loop is repeated, the inner loops are reentered, and started anew.

```java
public class MultiplicationTable {
    public static void main(String[] args) {
        System.out.println(" Multiplication Table"); // heading
        // Display the number title
        System.out.print("    ");
        for (int j = 1; j <= 9; j++)
            System.out.print("   " + j);
        System.out.println("\n———————————————————————");
        // Print table body
        for (int i = 1; i <= 9; i++) {
            System.out.print(i + " | ");
            for (int j = 1; j <= 9; j++) {
                // Display the product and align properly
                System.out.printf("%4d", i * j);
            }
            System.out.println();
        }
    }
}
```

Problem: Finding the Greatest Common Divisor

```
int gcd = 1; // Initial gcd is 1
int k = 2; // Possible gcd
while (k <= n1 && k <= n2) {
    if (n1 % k == 0 && n2 % k == 0)
            gcd = k; // Update gcd
            k++; // Next possible gcd
}
```

Keywords **break** and **continue**

- You have used the keyword **break** in a **switch** statement.
- You can also use **break** in a loop to immediately terminate the loop.

Example of **break**

```
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);
    }
}
```

Keywords **break** and **continue**

- Use the **continue** keyword in a loop
  - it ends the *current* iteration.
  - program control goes to the end of the loop body.
- **Continue** breaks out of an iteration while the **break** keyword breaks out of a loop.

Example of **break**

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

       while (number < 20) {
              number++;
              if (number == 10 || number == 11)
                     continue;
              sum += number;
       }
       System.out.println("The sum is " + sum);
   }
}
```

Exercises

Ex1: Cho 2 số thực a và b. Tìm số lớn nhất giữa 2 số đó.

Ex2: Viết chương trình:

1. Cho vào 1 năm dương lịch. Xét năm đó có phải là năm nhuận không.
2. Cho vào tháng và năm. Tính số ngày trong tháng.

Ex3: Cho các hệ số a và b của phương trình $ax + b = 0$. Tìm nghiệm của phương trình.

Ex4: Cho các hệ số a, b và c của phương trình $ax^2 + bx + c = 0$. Tìm nghiệm của phương trình.

Ex5: Cho a1, b1, a2, b2 là các điểm đầu mút của 2 đoạn [a1, b1] và [a2, b2] trên trục số. Tìm độ dài phần giao và phần hợp của 2 đoạn.

Exercises (2)

<u>Ex6</u>: Cho 3 số a, b, c. Xét 3 số đó có là 3 cạnh của tam giác. Nếu đúng, thì tính chu vi, diện tích, và số đo độ của các góc của tam giác.

Hướng dẫn:

- *Để 3 số là các cạnh của một tam giác thì tổng 2 số bất kỳ phải lớn hơn số còn lại.*
- *Diện tích tam giác là S =sqrt(p(p-a)(p-b)(p-c)) với p là nửa chu vi.*
- *Tính số đo góc A: cosA = $(b^2+c^2-a^2)/(2bc)$, tgA = sqrt($(1/\cos^2 A) - 1$), A = arctg(tgA).*

Exercises (3)

Ex7: Cho 3 số a, b, c. Xét 3 số đó có là 3 cạnh của tam giác. Nếu đúng, thì kiểm tra tam giác đó là tam giác gì? (đều, cân, vuông, vuông cân, thường).

Ex8: Cho năm dương lịch n. Xác định năm âm lịch tương ứng. Ví dụ: 1998 là năm Mậu Dần.

Ex9: Cho số tự nhiên n < 1000. Tính ra cách viết số đó bằng chữ. Ví dụ: 125 đọc là Một trăm hai mươi lăm.

Ex10: Cho 3 số nguyên d, m, y. Xét xem ngày được tạo bởi 3 số đó theo dạng d/m/y có hợp lệ không? Nếu hợp lệ, thì in ra ngày hôm sau của ngày đó. Ví dụ: Ngày 29/2/1996 hợp lệ và ngày hôm sau là 1/3/1996.

Exercises: Loop

- **Bài 1:** Nhập từ bàn phím vào các số nguyên và dừng lại khi nhập giá trị 0. Tính tổng, trung bình cộng và tìm giá trị lớn nhất của các số nguyên vừa nhập.
- **Bài 2**: Cho số tự nhiên n.
  - Đếm số chữ số của số nguyên đó.
  - Tìm số đảo ngược của số n.
- **Bài 3**: Năm nay cha 35 tuổi, con 4 tuổi. Tính xem sau bao nhiêu năm nữa tuổi cha gấp đôi tuổi con.
- **Bài 4**: Cho 2 số tự nhiên a và b. Tìm ước số chung lớn nhất và bội số chung nhỏ nhất của 2 số đó.

Exercises: Loop (2)

- **<u>Bài 5:</u>** Cho số tự nhiên n.
  - Tìm ước lẻ lớn nhất của n.
  - Kiểm tra xem số đó có là số nguyên tố không.
  - Phân tích số n ra các thừa số nguyên tố.
  - Tìm và in ra tất cả các số nguyên tố nhỏ hơn n.
- **<u>Bài 6</u>**: In ra màn hình các giá trị sin, cos, tang, cotang của các góc 0^0, 5^0, 10^0, …, 90^0.
- **<u>Bài 7</u>**: In ra màn hình bảng cửu chương (8 bảng từ 2 đến 9)..

Exercises: Loop (3)

- **<u>Bài 8:</u>** Cho số tự nhiên n.

  a. Tính tổng S = 1 + 2 + … + n.

  b. Tính giai thừa n! = 1 x 2 x … x n..

- **<u>Bài 9</u>**: Cho số tiền gởi ngân hàng P, lãi suất tiền gởi từng tháng r, số tháng gởi n. Tính và xuất số tiền sẽ được rút ra F sau n tháng theo công thức $F = P(1 + r)^n$.

Reference

- **Introduction to Java Programming** 8th , Y. Daniel Liang.