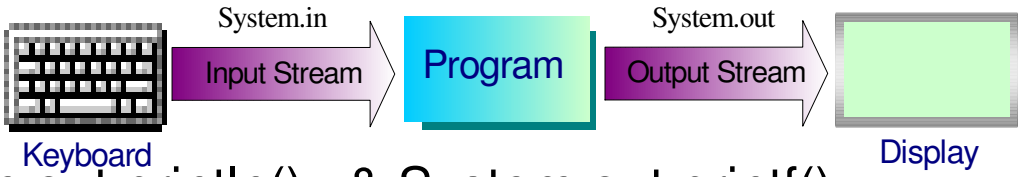
Chapter 6

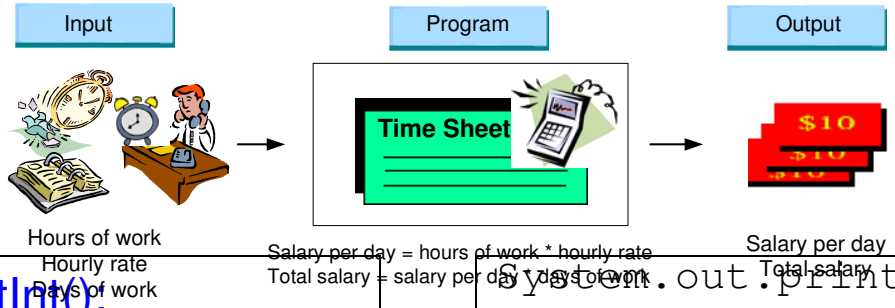
Branching

Review: Console Input/Output

- **Standard Input/Output Streams**
  - System.in, System.out, System.err



- **Console Output**
  - System.out.print(), System.out.println(), & System.out.printf() for formatted output
- **Console Input**
  - Scanner class: next(), nextBoolean(), nextDouble(), etc.




```
empNo = sc.nextInt();
hoursOfWork = sc.nextDouble();
...
```

```
System.out.println("The salary is:");
System.out.println("Employee no: " +
empNo);
```

```
salaryPerDay = hoursOfWork * hourlyRate;
totalSalary = salaryPerDay * daysOfWork;
```

Branching

- **Branching (or Selection)**
- Relational and Logical Operators
- The if Statement
- The if-else Statement
- The if-else if-else Statement
- The Nested-if Statement
- The switch Statement
- The Conditional Operator
- Case Study

Why Branching (or Selection)?

Decision! Decision! Decision!!!!

- Java statements are executed normally in **sequence**.
- Sometimes we need to tell the computer to do something only when some **conditions** are satisfied.
- This will **alter** the normal sequential execution.

Note: During program runtime!!!

Branching Constructs

There are two types of constructs that provide branching support:

Basically two-way

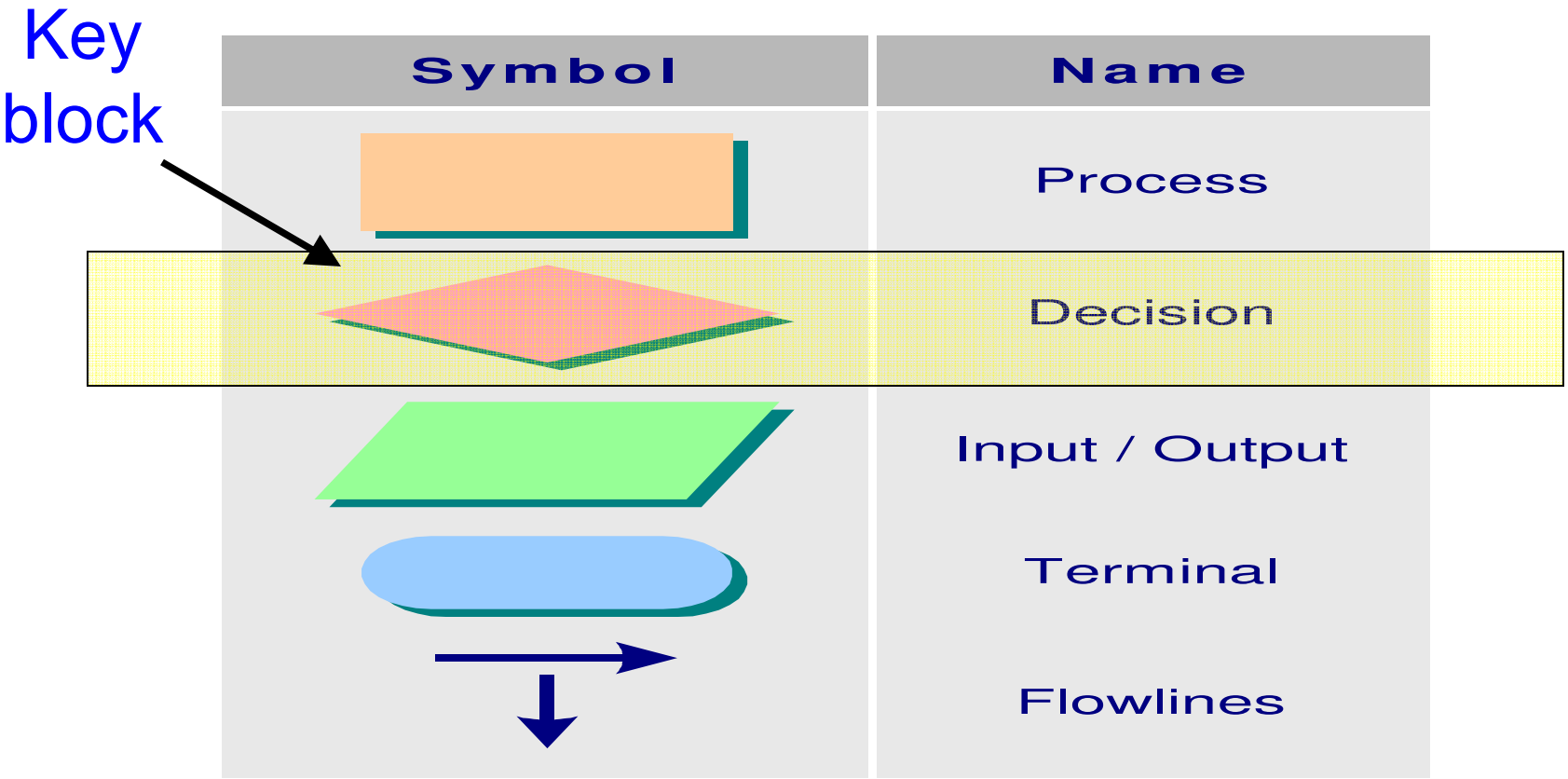
(1) The ***if*** statement - there are three forms:

- if statement
- if ... else statement
- if ... else if ... else statement

(2) The ***switch*** statement - it provides a multi-way decision structure. We can choose one action out of a number of actions depending on some conditions.

Recap: Flowcharts

We will use flowcharts to show branching logic in the examples discussed in this chapter.



Branching

- Branching (or Selection)
- **Relational and Logical Operators**
- The if Statement
- The if-else Statement
- The if-else if-else Statement
- The Nested-if Statement
- The switch Statement
- The Conditional Operator
- Case Study

Relational Operators

Used for **comparison** between **two values**.

Return **boolean** result: true or false.

Relational Operators:

operator	example	meaning
==	ch == 'a'	equal to
!=	f != 0.0	not equal to
<	num < 10	less than
<=	num <=10	less than or equal to
>	f > -5.0	greater than
>=	f >= 0.0	greater than or equal to

Note: checking equality of two floating-point numbers

Logical Operators

- Work on one or more relational (boolean) expressions to yield a logical value: true or false.
- Allows testing results of comparison expressions.

Logical operators:

operator	example	meaning
!	!(num < 0)	not
&&	(num1 > num2) && (num2 >num3)	and
||	(ch == '\t') || (ch == ' ')	or

	A is true	A is false
!A	false	true

A || B	A is true	A is false
B is true	true	true
B is false	true	false

A && B	A is true	A is false
B is true	true	false
B is false	false	false

True table

(1) List of operators of **decreasing precedence**:

!	**not**
*** /**	**multiply and divide**
+ -	**add and subtract**
< <= > >=	**less, less or equal, greater, greater or equal**
== !=	**equal, not equal**
&&	**logical and**
||	**logical or**

- The result of evaluating an expression involving relational and/or logical operators is
  ➔ either true or false.

(2) Evaluating Relational and Logical Expressions

```
public class EvaluateExpressions {
   public static void main(String[] args) {
      boolean result;
      System.out.println("The results of logic
            relations:");
      result = (3 > 7);
      System.out.println("(3 > 7) is " + result);
      result = (7 < 3) && (3 <= 7);
      System.out.println("(7 < 3) && (3 <= 7) is " +
            result);
      result = (7 < 3) && (7 / 0 <= 5);
      System.out.println("(7 < 3) && (7/0 <= 5) is " +
            result);
      result = (32 / 4 > 3 * 4) || (4 == 4);
      System.out.println("(32/4 > 3*4) || (4 == 4) is " +
            result);
      result = (4 == 4) || (32 / 0 == 0);
      System.out.println("(4 == 4) || (32/0 == 0) is " +
            result);
   }
}
```

A bit smart!!! Early termination!!!

Program Sample Output

```
The results of logic relations:
(3 > 7) is false
(7 < 3) && (3 <= 7) is false
(7 < 3) && (7/0 <= 5) is false
(32/4 > 3*4) || (4 == 4) is true
(4 == 4) || (32/0 == 0) is true
```

- Composite Condition:

result = (7 < 3) && (3 <= 7) && (12 < 18);

Can we have early termination here?

Branching

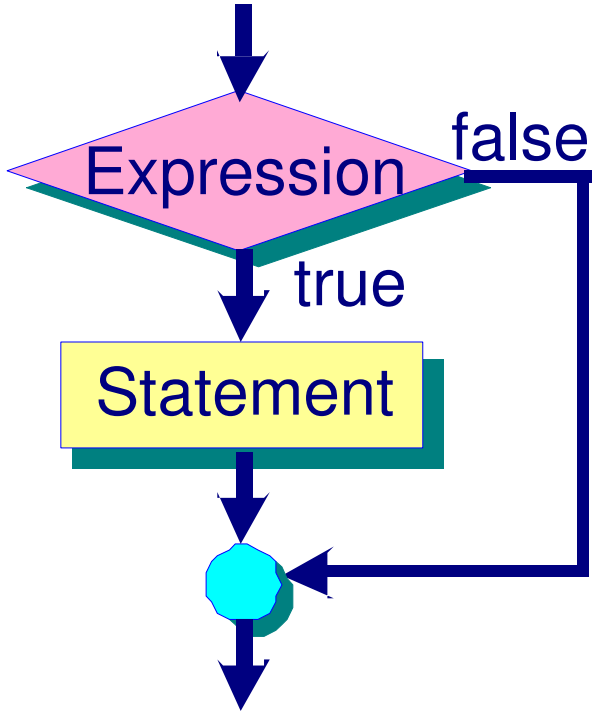
- Branching (or Selection)
- Relational and Logical Operators
- **The if Statement**
- **The if-else Statement**
- **The if-else if-else Statement**
- The Nested-if Statement
- The switch Statement
- The Conditional Operator
- Case Study

The if Statement

- The format of the if statement:

```
if (Expression)
        Statement;
```

Body of the true part of “if”



- **Statement** may be
  (1) a single statement terminated by a semicolon; or
  (2) a compound statement enclosed by { }

Example: Displaying Exam Marks

```
import java.util.Scanner;
public class ExamMark {
   public static void main(String[] args) {
      int mark; Scanner sc= new Scanner(System.in);
      System.out.print("Give your examination mark => ");
      mark = sc.nextInt();
      if (mark >= 80)
         System.out.println("You scored A in
            examination");
      System.out.println("Your mark is " + mark);
   }
}
```

Program Input and Output

```
Give your examination mark => 50
Your mark is 50
```

Program Input and Output

```
Give me your examination mark => 88
You scored A in examination
Your mark is 88
```

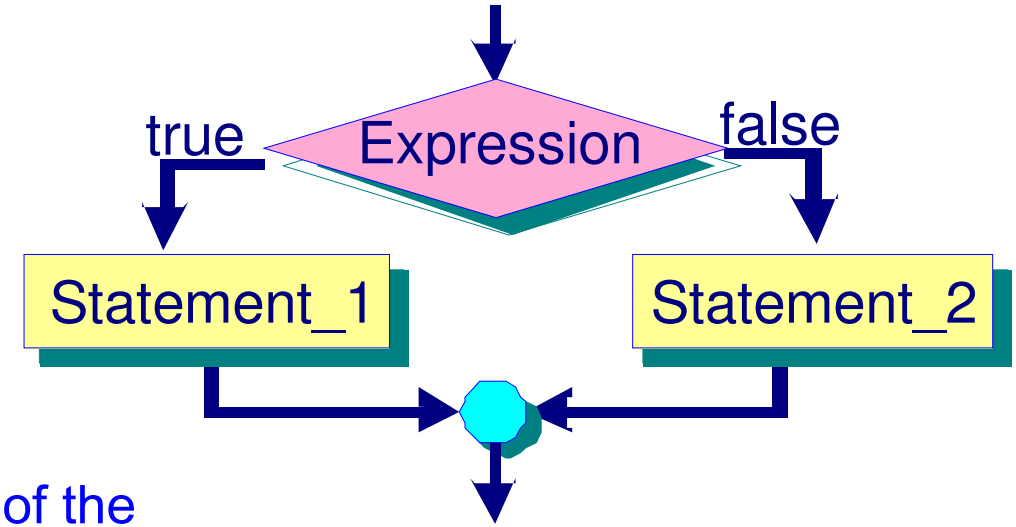
The if-else Statement

The format of the if ... else statement is

```
if (Expression)
    Statement_1;
else
    Statement_2;
```

Body of the false part of "if" OR Body of "else"

Body of the true part of "if"



- Both Statement_1 and Statement_2 may be a single statement terminated by a semicolon or a compound statement enclosed by { }

Example: Having a fever

```
import java.util.Scanner;
public class Fever {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      System.out.print( "Enter your temperature: " );
      temperature = sc.nextDouble();
      if (temperature >= 37.0)
         System.out.println( "You are having a fever" );
      else
         System.out.println( "You have no fever" );
   }
}
```

fever!!

Program Input and Output

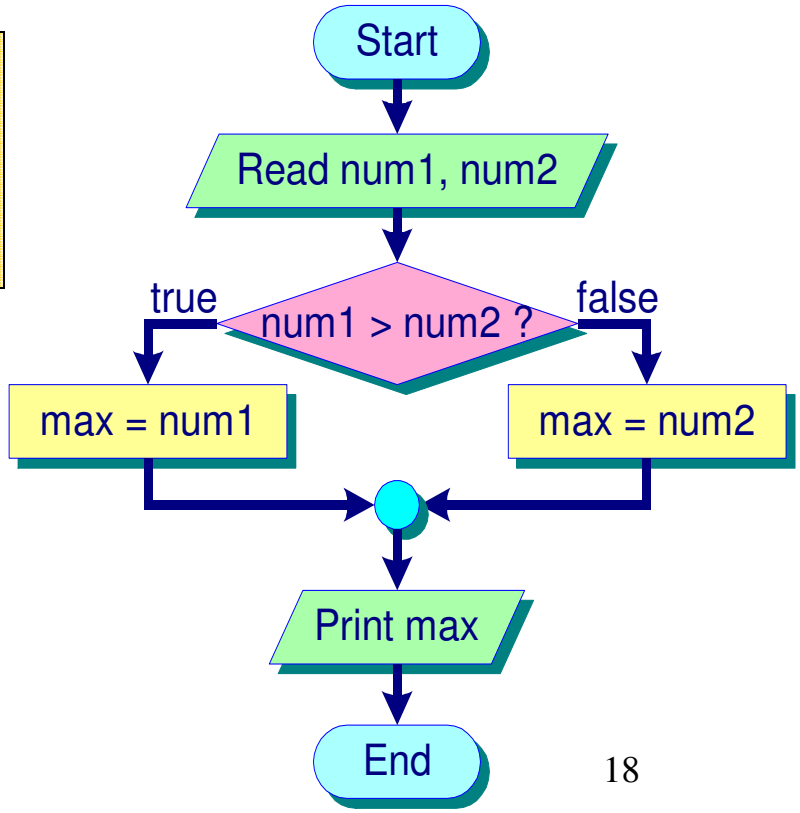
```
Enter your temperature => 37.8
You are having a fever
```

Example: Computing the Maximum Value of Two Integers

```
import java.util.Scanner;
public class ComputeMax {
   public static void main(String[] args) {
      int num1,num2,max;Scanner sc=new Scanner(System.in);
      System.out.print("Enter the 1st number : ");
      num1 = sc.nextInt();
      System.out.print("Enter the 2nd number : ");
      num2 = sc.nextInt();
      if (num1 > num2)
         max = num1;
      else
         max = num2;
      System.out.println(
         "The maximum is " + max);
   }
}
```

Program Input and Output

```
Enter the 1st number : 9
Enter the 2nd number : 4
The maximum is 9
Enter the 1st number : -2
Enter the 2nd number : 0
The maximum is 0
```



Example: Computing the Maximum Value of Two Integers

Can we change from:

```
if (num1 > num2)
    max = num1;
else
    max = num2;
```

to

```
if (num1 > num2)
    max = num1;
if (num1 <= num2)
    max = num2;
```

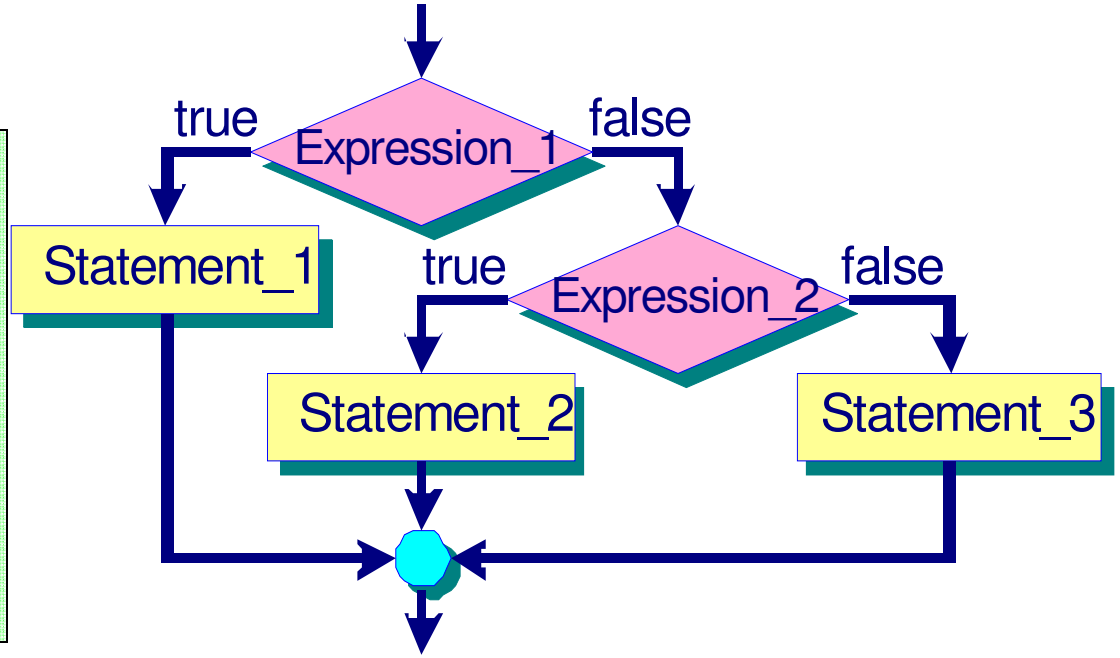
Logically the same, but…
You miss the power of **if-else**

where we make one decision **at one time**, and we can go to two different possible paths

The if-else if-else Statement

The format is

```
if  (Expression_1)
    Statement_1;
else if  (Expression_2)
    Statement_2;
else
    Statement_3;
```



Each of *Statement_1*, *Statement_2* and *Statement_3* may be a single statement terminated by a semicolon or a compound statement enclosed by { }

- We may have as many else if in the if statement as we want (within the compiler limit):

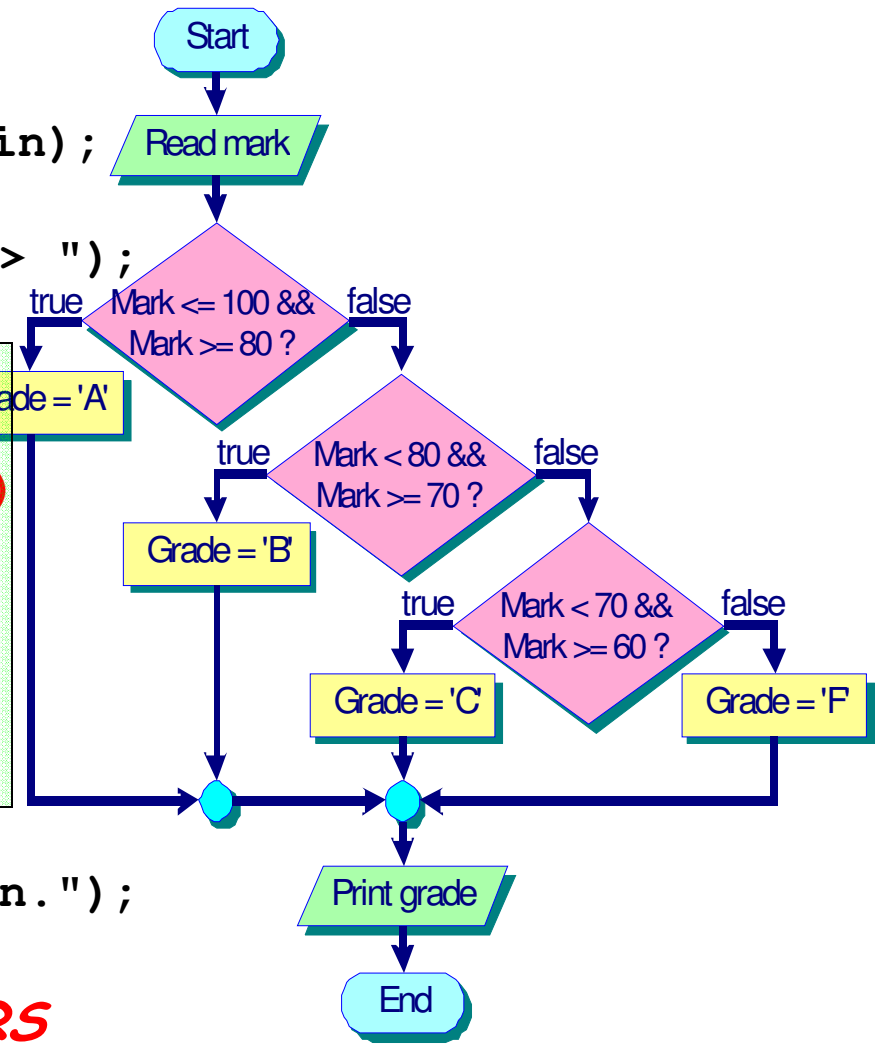
```
if  (Expression_1)
        Statement_1;
else if  (Expression_2)
        Statement_2;
else if (Expression_3)
        Statement_3;
......
else
        Statement_n;
```

- Statement_i is executed when the first i-1 expressions are false and the ith expression is true.

Example: Computing Grade

```
import java.util.Scanner;
public class ExamGrade {
  public static void main(String[] args)
  {
    int mark; char grade;
    Scanner sc = new Scanner(System.in);
    System.out.println(
        "Give your examination mark => ");
   mark = sc.nextInt();
   if (mark <= 100 && mark >= 80)
        grade = 'A';
    else if (mark < 80 && mark >= 70)
        grade = 'B';
   else if (mark < 70 && mark >= 60)
        grade = 'C';
   else
        grade = 'F';
   System.out.println("You scored "
        + grade+" in your examination.");
  }
}
```

NB: USING LOGICAL OPRTATORS



Program input and output

```
Give your examination mark => 85
You scored A in your examination.

Give your examination mark => 65
You scored C in your examination.

Give your examination mark => 35
You scored F in your examination.
```

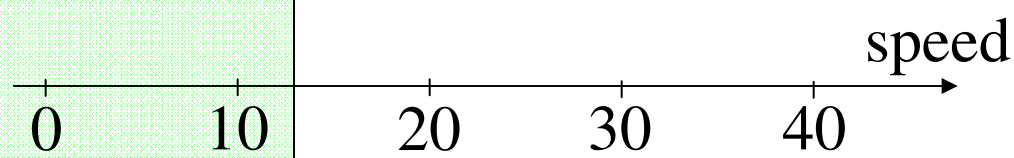
Self study: Any assumption on user input? If user inputs….

Example: Computing Fine

```
import java.util.Scanner;
public class ComputeFine {
  public static void main(String[] args)
  {
    int speed, fine;
    Scanner sc = new Scanner(System.in);
    System.out.println("Give your speed => ");
    speed = sc.nextInt();
    if (speed > 10 && speed <= 20)
        fine = 10;
    else if (speed > 20 && speed <= 30)
        fine = 20;
    else if (speed > 30 && speed <= 40)
        fine = 30;
    else if (speed > 40)
        fine = 40;
    else
        fine = 0;
    System.out.println("You fine is " + fine);
  }
}
```

Over-speed	Fine
>10 km/h	$10
>20 km/h	$20
>30 km/h	$30
>40 km/h	$40

EXERCISE:
DRAW THE FLOWCHART?



Is this program efficient?

Example: Computing Fine (II)

```
...
if (speed > 10 && speed <= 20)
    fine = 10;
else if (speed > 20 && speed <= 30)
    fine = 20;
else if (speed > 30 && speed <= 40)
    fine = 30;
else if (speed > 40)
    fine = 40;
else
    fine = 0;
System.out.println( ... );
```

Can you make this change?

```
...
if (speed > 10 && speed <= 20)
    fine = 10;
if (speed > 20 && speed <= 30)
    fine = 20;
if (speed > 30 && speed <= 40)
    fine = 30;
if (speed > 40)
    fine = 40;
else
    fine = 0;
System.out.println( ... );
```

Logically wrong

Example: Computing Fine (III)

```
...
if (speed > 10 && speed <= 20)
    fine = 10;
else if (speed > 20 && speed <= 30)
    fine = 20;
else if (speed > 30 && speed <= 40)
    fine = 30;
else if (speed > 40)
    fine = 40;
else
    fine = 0;
System.out.println( ... );
```

How about this change?

```
...
fine = 0;
if (speed > 10 && speed <= 20)
    fine = 10;
if (speed > 20 && speed <= 30)
    fine = 20;
if (speed > 30 && speed <= 40)
    fine = 30;
if (speed > 40)
    fine = 40;
System.out.println( ... );
```

Logically correct but miss the power of if-else if-else

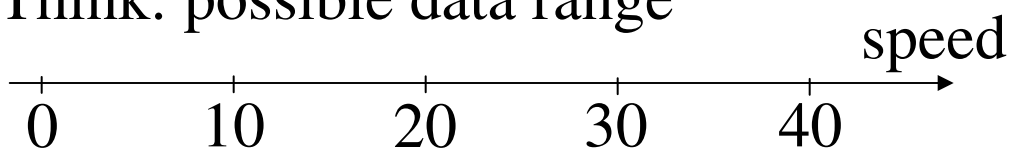
Example: Computing Fine (IV)

Over-speed	Fine
>10 km/h	$10
>20 km/h	$20
>30 km/h	$30
>40 km/h	$40

```
import java.util.Scanner;
public class ComputeFine {
  public static void main(String[] args)
  {
    int speed, fine;
    Scanner sc = new Scanner(System.in);
    System.out.println("Give your speed => ");
    speed = sc.nextInt();
    if (speed > 40)
        fine = 40;
    else if (speed > 30)
        fine = 30;
    else if (speed > 20)
        fine = 20;
    else if (speed > 10)
        fine = 10;
    else
        fine = 0;
    System.out.println("You fine is " + fine);
  }
}
```

EXERCISE:
DRAW THE FLOWCHART?

Think: possible data range



=> More logical and efficiency

Branching

- Branching (or Selection)
- Relational and Logical Operators
- The if Statement
- The if-else Statement
- The if-else if-else Statement
- **The Nested-if Statement**
- The switch Statement
- The Conditional Operator
- Case Study

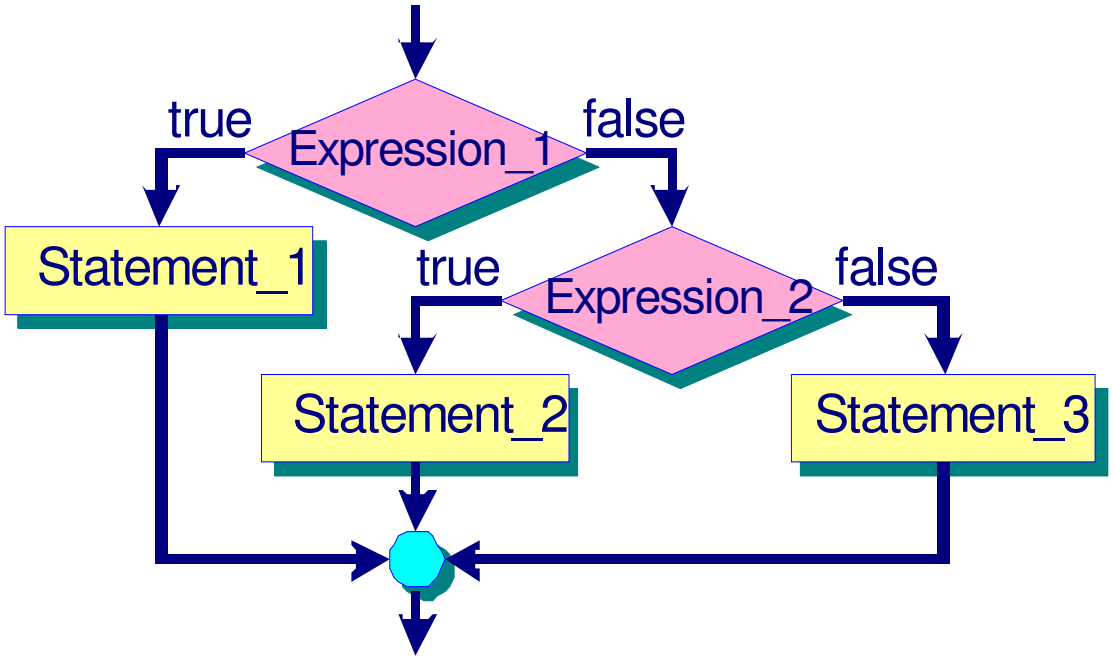
Nested-if

Why nested-if?

- Both the if branch and the else branch may contain if statement(s). The level of nested if statements can be as many as we want (up to the compiler limit).

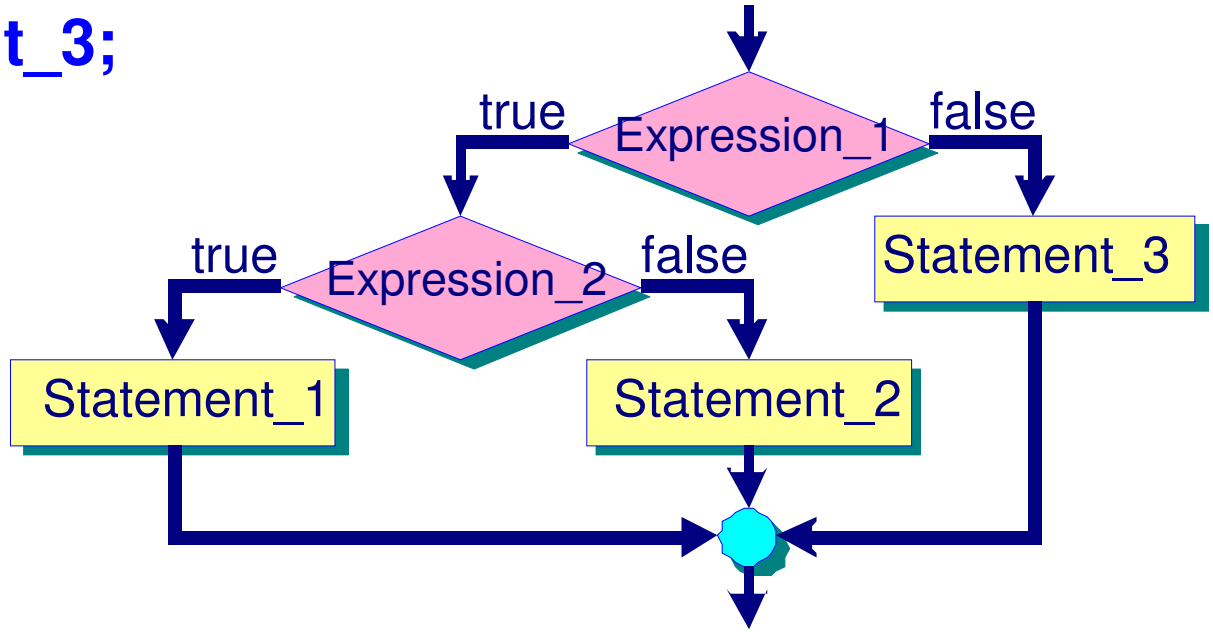
E.g.

```
if (Expression_1)
    Statement_1;
else
    if (Expression_2)
        Statement_2;
    else
        Statement_3;
```



```
if (Expression_1)
    if (Expression_2)
        Statement_1;
    else
        Statement_2;
else
    Statement_3;
```

Nested



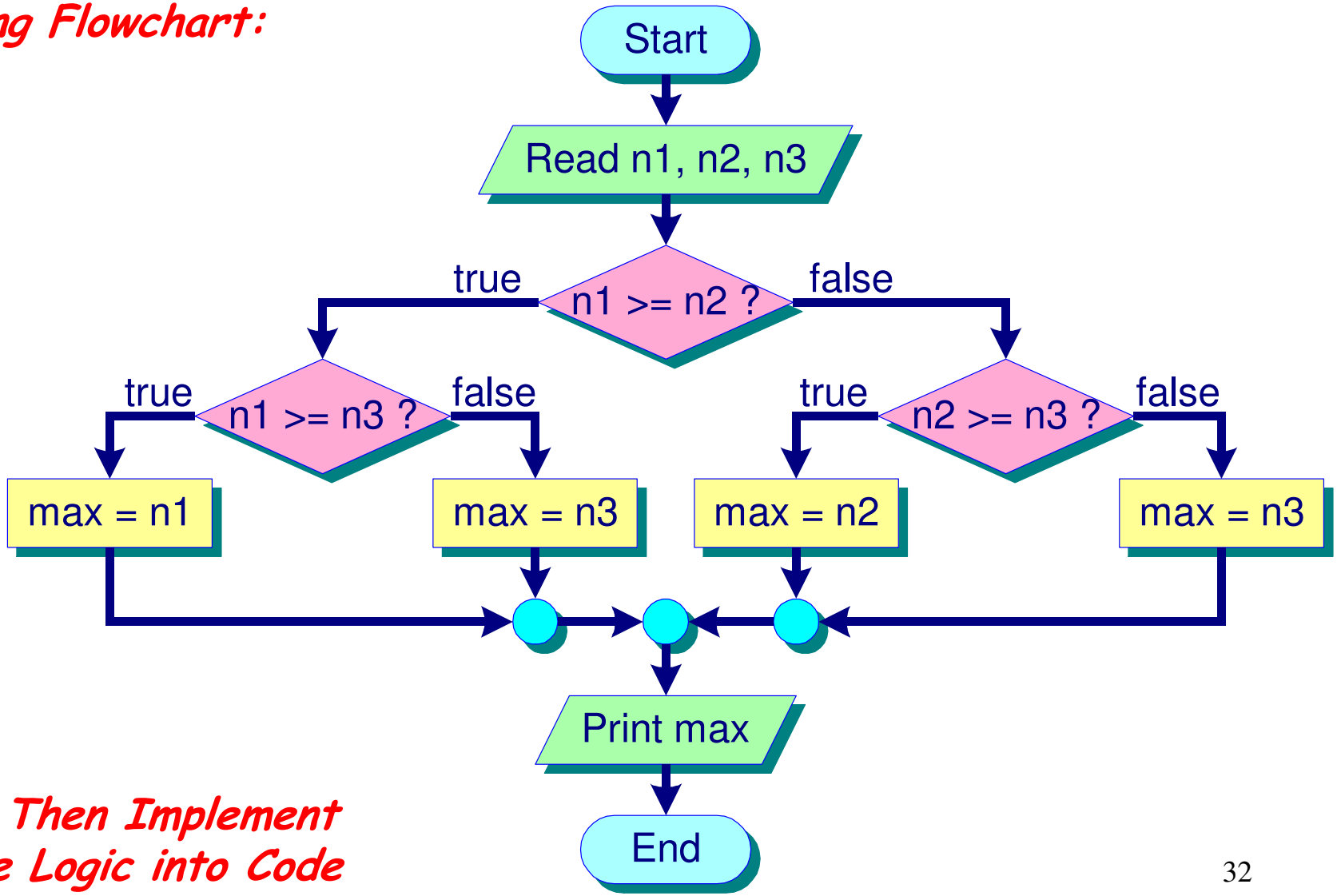
- Rule -> associates an *else* part with the nearest unresolved *if*.

```
if (Expression_1)
    if (Expression_2)
        Statement_1;
else
        Statement_2;
```

- *Statement_1* - is executed when *Expression_1* and *Expression_2* are true.
- *Statement_2* - is executed when *Expression_1* is true and *Expression_2* is false.
- When *Expression_1* is false - neither statements are executed.

Example: Finding the Maximum Value of 3 numbers

1. Design the Logic using Flowchart:



2. Then Implement the Logic into Code

Example: Finding Maximum Value Using Nested-if Statement

```
import java.util.Scanner;
public class MaxThreeInteger {
    public static void main(String[] args) {
    int n1, n2, n3, max;
    Scanner sc = new Scanner(System.in);
    System.out.println("Enter 1st integer: ");
    n1 = sc.nextInt();
    System.out.println("Enter 2nd integer: ");
    n2 = sc.nextInt();
    System.out.println("Enter 3rd integer: ");
    n3 = sc.nextInt();
    if (n1 >= n2) {
        if (n1 >= n3)
          max = n1;
        else
          max = n3;
    }
    else if (n2 >= n3)
        max = n2;
    else
        max = n3;
    System.out.println("The max is "+max);
  } }
```

Program Output

```
Enter 1st integer:1
Enter 2nd integer:2
Enter 3rd integer:3
The maximum is 3
```

How about 4 numbers? 5 numbers?

Example: Member charges

```
import java.util.Scanner;
public class MemberCharges {
    public static void main(String[] args) {
    ...
    Scanner sc = new Scanner(System.in);
    System.out.println("Enter ...: ");
    member = sc.nextBoolean();
    ...
    if (member)
    {
        if (age > 12)
          charge = 5.6;
        else
          charge = 3.3;
     }
     else {
        if (age > 12)
           charge = 8.7;
         else
           charge = 4.5;
     }
}
```

There are two conditions:

1. *Membership?*
2. *Age?*

in order to determine membership charges.

EXERCISE:
DRAW THE FLOWCHART?

Use parenthesis -> Always safe!!!

Review Questions

What's wrong with the following code fragment?

```
(A) 1 : if (length = MAX_LENGTH)
    2 :   System.out.println("The length is maximal!!");

(B) 1 : if (total == MAX)
    2 :   if (total < sum)
    3 :     System.out.println("total == Max and < sum");
    4 : else
    5 :   System.out.println("total is != MAX");
```

Note: Indentation is for improving program readability only!!!
No effect on JAVA Compiler but effective for Human!!!

Review Questions

What output is produced?

```
(A) 1 : int num1=50, max=10;
    2 : if (num1 >= max*2)
    3 :   System.out.println("apple");
    4 :   System.out.println("orange");
    5 : System.out.println("grape");
```

```
(B) 1 : if (x > 2) {
    2 :   if y > 2) {
    3 :       int z = x + y;
    4 :       System.out.println("z = " + z);
    5 :   }
    6 : }
    7 : else
    8 :   System.out.println("x = " + x);
```

Note: Do indentation meaningfully!!!

Suppose x = 2, y = 3. What is the output?

Suppose x = 3, y = 2. What is the output?

Suppose x = 3, y = 4. What is the output?

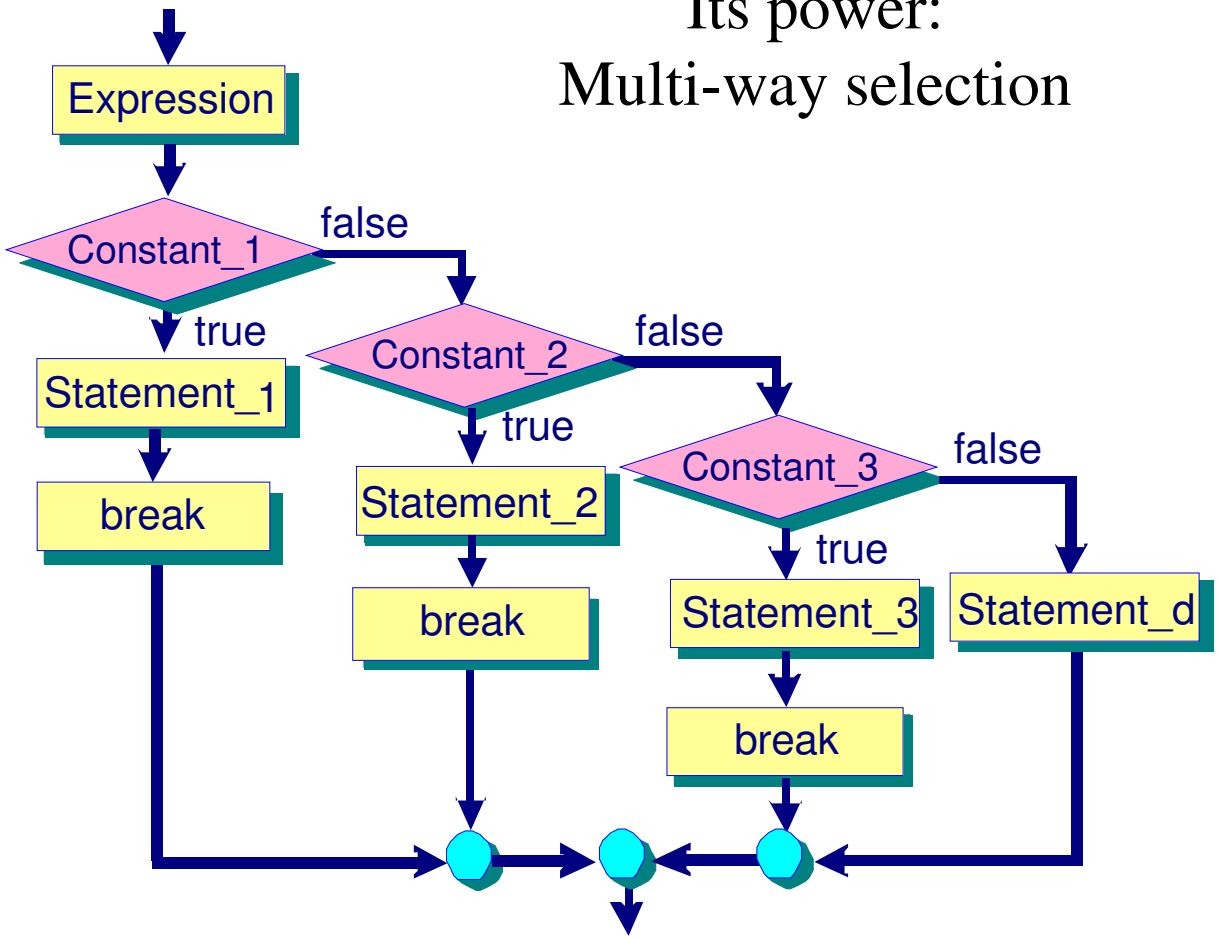
Branching

- Branching (or Selection)
- Relational and Logical Operators
- The if Statement
- The if-else Statement
- The if-else if-else Statement
- The Nested-if Statement
- **The switch Statement**
- The Conditional Operator
- Case Study

The switch Statement

The syntax of a **switch** statement is:

```
switch (Expression) {
  case  Constant_1:
        Statement_1;
        break;
  case  Constant_2:
        Statement_2;
        break;
  case  Constant_3:
        Statement_3;
        break;
  default :
        Statement_d;
}
```



Its power:
Multi-way selection

- *switch*, *case, break* and *default* are reserved words.
- The result of *Expression* in ( ) must be **integral type.**
- *Constant_1, Constant_2*, ... are called labels. Each must be an integer constant, a character constant or an integer constant expression, e.g. 3, 'A', 4+'b', 5+7, ... etc.
- Each of the labels *Constant_1, Constant_2*, ... must deliver **unique integer value.** Duplicates are not allowed.
- We may also have **multiple labels** for a statement, for example, to allow both the lower and upper case selection.
- default branch is optional

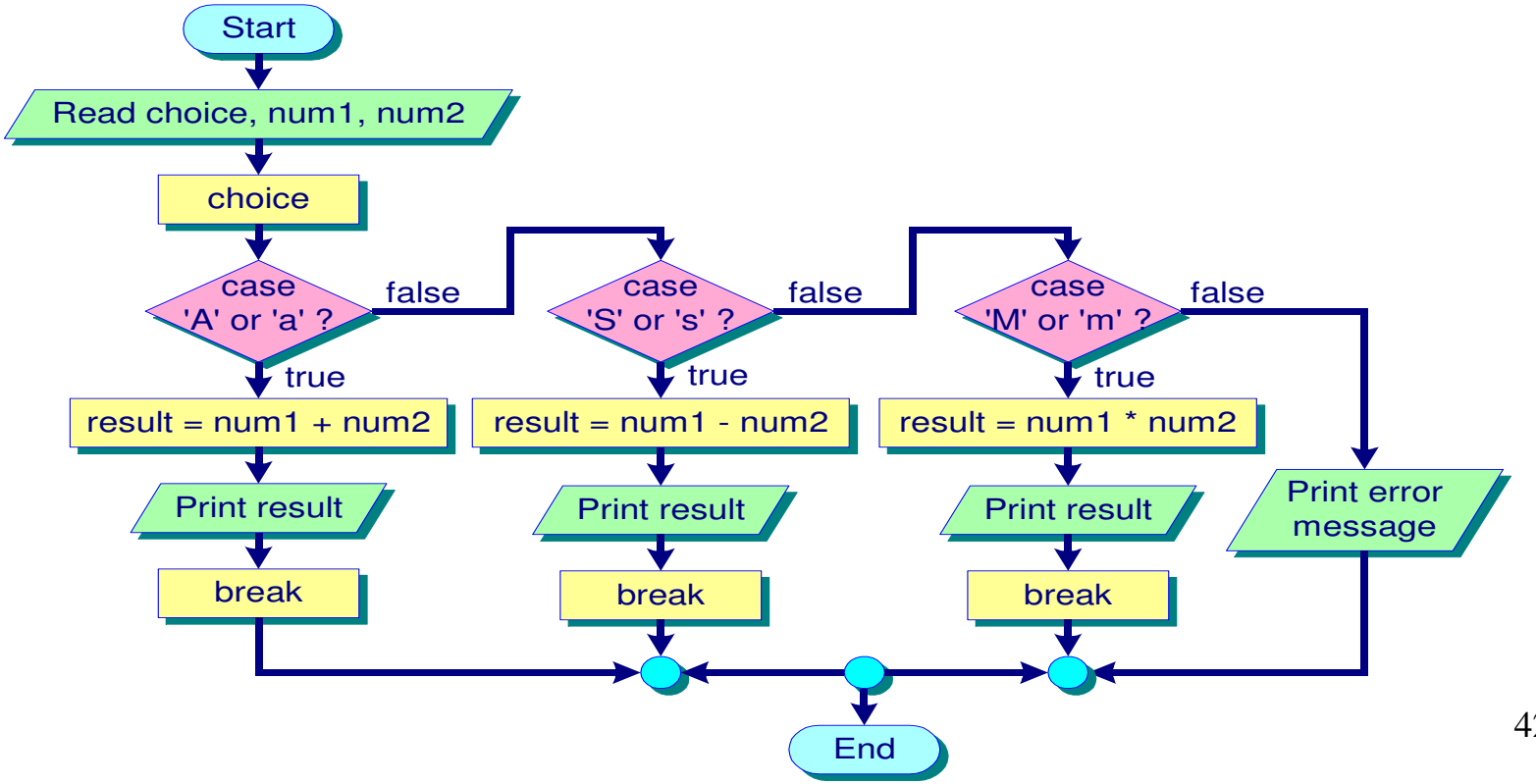
Example: Using the switch Statement

```
import java.util.Scanner;
public class UsingSwitchApp {
   public static void main(String[] args) {
      char choice;
      int num1, num2, result;
      Scanner sc = new Scanner(System.in);
      /* get input */
      System.out.println("Select an operation:");
      System.out.println("A) Addition");
      System.out.println("S) Subtraction");
      System.out.println("M) Multiplication");
      System.out.println("Your choice (A, S or M) => ");
      // String choiceStr = sc.next();
      // choice = choiceStr.charAt(0);          (detail: Chp 11)
      choice = sc.next().charAt(0); //read in a char
      System.out.println("Enter the first integer: ");
      num1 = sc.nextInt();
      System.out.println("Enter the second integer: ");
      num2 = sc.nextInt();
```

```
    switch (choice) {
      case 'a':                    // supporting multiple labels
      case 'A':
        result = num1 + num2;
        System.out.println(num1 + " + " + num2 + " = " +
              result);
        break;
      case 's':
      case 'S':
        result = num1 - num2;
        System.out.println(num1 + " - " + num2 + " = " +
             result);
        break;
      case 'm':
      case 'M':
        result = num1 * num2;
        System.out.println(num1 + " * " + num2 + " = " +
              result);
        break;
      default:
        System.out.println("Not a proper choice!");
    }
  }
}
```

Program Input and Output

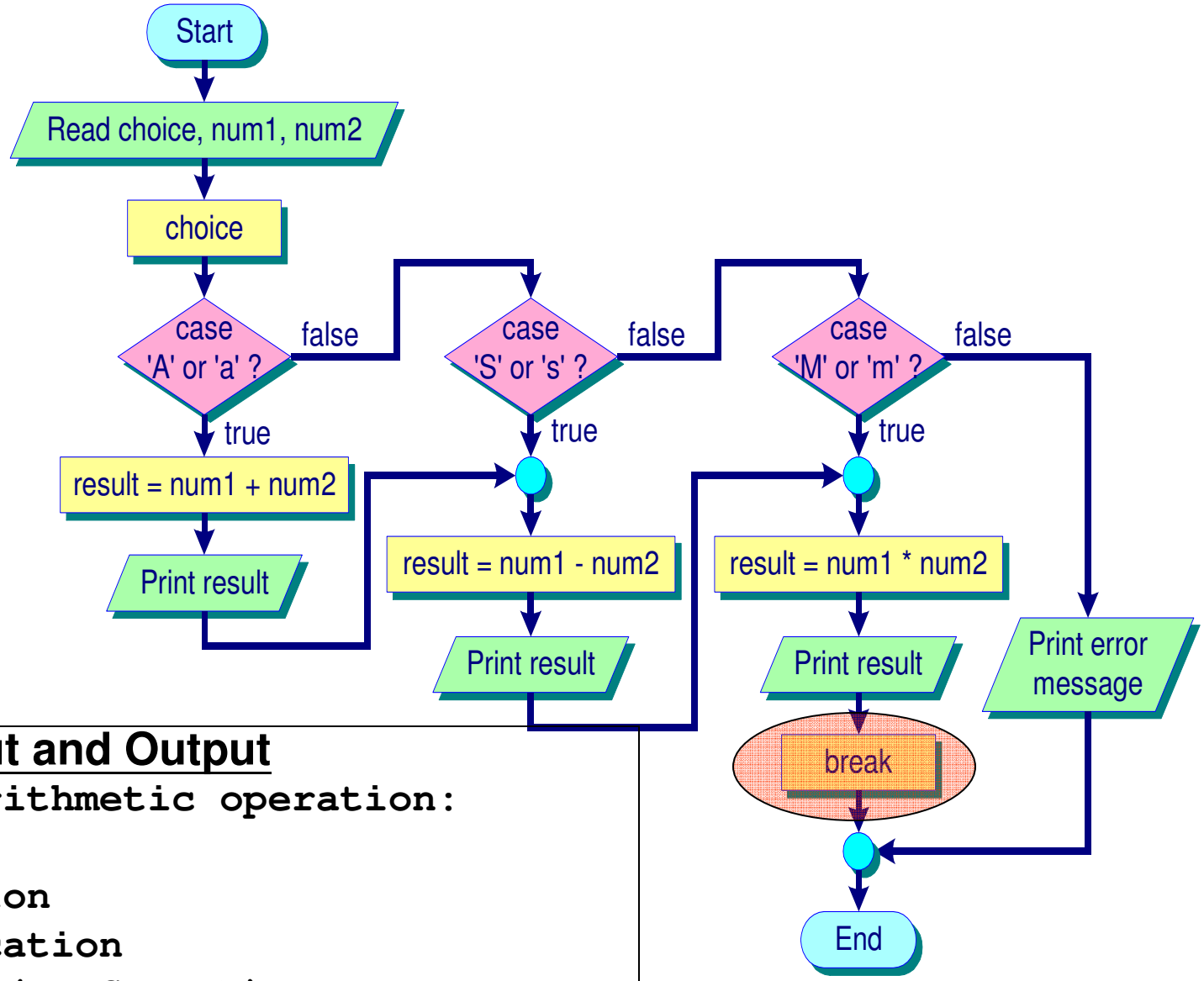
```
Select an operation:
A) Addition
S) Subtraction
M) Multiplication
Your choice (A, S or M) => S
Enter the first integer: 9
Enter the second integer: 5
9 - 5 = 4
```



If we **DO NOT** use break after some statements in the switch statement, execution will continue with the statements for the subsequent labels until a break statement or the end of switch statement. This is called **fall through** situation.

```
switch (choice) {
  case 'a':
  case 'A':
    result = num1 + num2;
    System.out.println(num1 + " + " + num2 + " = " + result);
  case 's':
  case 'S':
    result = num1 - num2;
    System.out.println(num1 + " - " + num2 + " = " + result);
  case 'm':
  case 'M':
    result = num1 * num2;
    System.out.println(num1 + " * " + num2 + " = " +
            result);
    break;
  default:
    System.out.println("Not a proper choice!");
 }
```

No break



Program Input and Output

```
Select an arithmetic operation:
A) Addition
S) Subtraction
M) Multiplication
Your choice (A, S or M) => A
Enter the first integer: 5
Enter the second integer: 3
```

-- WHAT ARE THE OUTPUTS??

Review Questions

What output is produced?

```
int key = 1;
switch ( key ){
    case 1 :  System.out.println("apples");
              break;
    case 2 :  System.out.println("oranges");
              break;
    case 3 :  System.out.println("grapes");
    case 4 :  System.out.println("peaches");
              break;
    default : System.out.println("bananas");
}
```

Suppose key = 3. What is the output?

Suppose key = 5. What is the output?

Branching

- Branching (or Selection)
- Relational and Logical Operators
- The if Statement
- The if-else Statement
- The if-else if-else Statement
- The Nested-if Statement
- The switch Statement
- **The Conditional Operator**
- Case Study

The Conditional Operator

- The conditional operator is used in the following way:

Expression_1 ? Expression_2 : Expression_3

The value of this expression depends on whether *expression_1* (boolean expression) is true or false.

If Expression_1 is true

=> value of the expression is that of *Expression_2*

Else

=> value of the expression is that of *Expression_3*

For example:

```
max = (x > y) ? x : y;
```

<==>

```
if (x > y)
        max = x;
else
        max = y;
```

Example: Using Conditional Operator

```
import java.util.Scanner;
public class ConditionOperator {
   public static void main(String[] args) {
      double ledVoltage, resistorVoltage;
      double sourceVoltage, circuitCurrent;
      double resistorValue;
      Scanner sc = new Scanner(System.in);
      System.out.print("Enter source voltage(volts)=> ");
      sourceVoltage = sc.nextDouble();
      System.out.print("Enter resistor value(ohms) => ");
      resistorValue = sc.nextDouble();

      // min(sourceVoltage,4.5)
      ledVoltage = (sourceVoltage<4.5)?sourceVoltage:4.5;

      resistorVoltage = sourceVoltage   - ledVoltage;
      circuitCurrent  = resistorVoltage / resistorValue;

      System.out.print("Total circuit current = " +
            circuitCurrent + " amperes");
   }
}
```

Program Input and Output

```
Enter source voltage(volts) => 1
Enter resistor value (ohms) => 20
Total circuit current = 0.0 amperes

Enter source voltage(volts) => 5
Enter resistor value (ohms) => 40
Total circuit current = 0.0125 amperes
```

Branching

- Branching (or Selection)
- Relational and Logical Operators
- The if Statement
- The if-else Statement
- The if-else if-else Statement
- The Nested-if Statement
- The switch Statement
- The Conditional Operator
- **Case Study**

Case Study: Computing Weighted Average Score & Grade

Problem Specification

Write a program to compute the **weighted average score** for each student taking the programming course. There are three components in the calculation. They are the **laboratory assignment**, **the mid-semester test** and **the examination**. The weighting factor for each component is given as follows:

- Laboratory assignment = 30%
- Test = 20%
- Examination = 50%

The program will

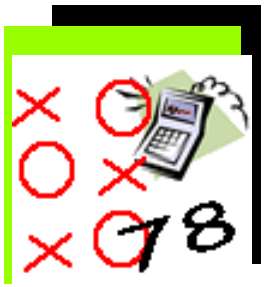
- read in the score of each component for a student,
- perform computation and
- display the corresponding weighted average score and grade of the student.

Problem Analysis

Input

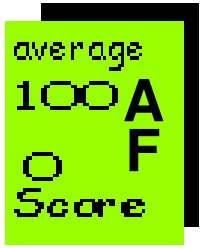
Student name
Lab score
Test score
Exam score

Program



Weighted average score = laboratory assignment score*weight for laboratory assignment + test score*weight for mid-semester test + examination score*weight for examination

Output



Average score
Grade

Problem Analysis

***Required inputs*:**

- the student name
- the laboratory assignment score
- the test score
- the examination score

***Required output*:**

- the printout of the weighted average score and grade for a student

***Program Constants*:**

- weight for laboratory assignment = 0.3
- weight for mid-semester test = 0.2
- weight for examination = 0.5

***Formulas*:**

- weighted average score = lab score * weight for lab + test score * weight for test + exam score * weight for exam
- grades are calculated according to the weighted average score as shown below:

Weighted Average Score	Grade
80 <= weighted average score	A
70 <= weighted average score < 80	B
60 <= weighted average score < 70	C
50 <= weighted average score < 60	D
40 <= weighted average score < 50	E
weighted average score < 40	F

Program Design

Initial Algorithm

1. Read name of a student.
2. Read laboratory assignment score, test score and examination score.
3. Compute the weighted average score.
4. Determine the grade based on the weighted average score.
5. Print the grade and weighted average score of the student.

Program Design

Algorithm in Pseudocode

```
main:
    READ studentName, labScore, testScore, examScore
    COMPUTE averageScore = labScore*labWeight +
        examScore*examWeight + testScore*testWeight
    IF 80 <= averageScore
        SET grade TO 'A'
    ELSE IF 70 <= averageScore AND averageScore < 80
        SET grade TO 'B'
    ELSE IF 60 <= averageScore AND averageScore < 70
        SET grade TO 'C'
    ELSE IF 50 <= averageScore AND averageScore < 60
        SET grade TO 'D'
    ELSE IF 40 <= averageScore AND averageScore < 50
        SET grade TO 'E'
    ELSE
        SET grade TO 'F'
    ENDIF
    PRINT studentName, averageScore, grade
```

Program Design

Program Dry-run

Inputs:

studentName = Alex Soh, labScore = 80, testScore = 80, examScore = 75.6

averageScore = 80*0.3 + 80*0.2 + 75.6*0.5 = 77.8
(70 <= averageScore) and (averageScore < 80) is true
grade = B

Outputs:

Student name is Alex Soh
The weighted average score is 77.8
The grade is B

Implementation

```
import java.util.Scanner;
public class ComputeScore {
   static final double LAB_WEIGHT = 0.3;
   static final double TEST_WEIGHT = 0.2;
   static final double EXAM_WEIGHT = 0.5;
   public static void main(String[] args) {
      String studentName;
      double averageScore,labScore,examScore,testScore;
      char grade = ' ';
      Scanner sc = new Scanner(System.in);
      // READ INPUT
      System.out.println("Enter student name: ");
      studentName = sc.nextLine();
      System.out.println("Enter lab assign score: ");
      labScore = sc.nextDouble();
      System.out.println("Enter test score: ");
      testScore = sc.nextDouble();
      System.out.println("Enter exam score: ");
      examScore = sc.nextDouble();
```

```
    averageScore = labScore * LAB_WEIGHT + examScore *
        EXAM_WEIGHT + testScore * TEST_WEIGHT;
```

```
    switch ((int)averageScore/ 10) {
        case 10: case 9: case 8:
                grade = 'A'; break;
        case 7:
                grade = 'B'; break;
        case 6:
                grade = 'C'; break;
        case 5:
                grade = 'D'; break;
        case 4:
                grade = 'E'; break;
        default:
                grade = 'F';
    }
```

```
    System.out.println("Student name is " + studentName);
    System.out.println("Weighted aveerage score is "
        + averageScore);
    System.out.println("The grade is " + grade);
  }
}
```

Testing

Program input and output

Enter student name:
Alex Soh
Enter lab assignment score:
80
Enter test score:
80
Enter exam score:
75.6
Student name is Alex Soh
The weighted average score is 77.8
The grade is B

Key Terms

- relational operator
- logical operator
- selection statement
- switch statement
- fall-through behavior
- break statement
- conditional operator

Further Reading

- Read Chapter 6 on “Branching” of the textbook
- Read other case studies from Section 6.8