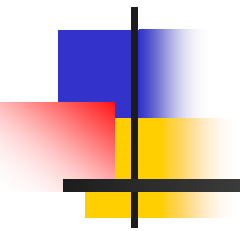
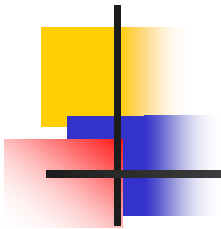
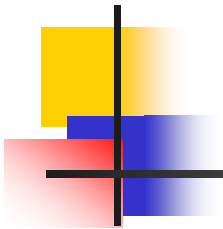


Selection Statements



Outline

- Block statement
- Branching Statements
  - Simple if statement
  - if-else statement
  - if-else-elseif statement
  - switch statement
- Nested if statements

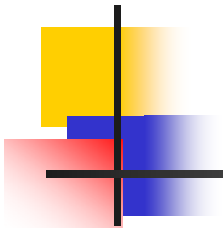


- Block Statement

- A block statement consists of one or more Java statements enclosed in braces.
- Example of a block statement:

```
{
        statement 1;
        statement 2;
           …
        statement n;
}
```

- Blocks can be nested.
- A block statement can be used anywhere that a single statement can be used.



- Branching Statement

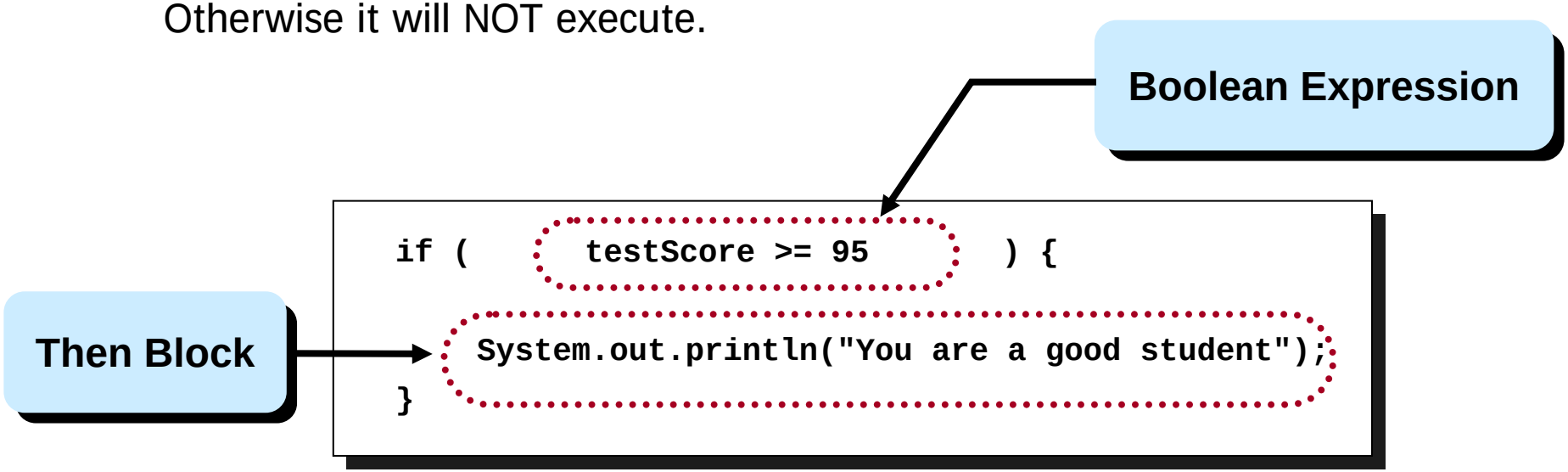
- A branching statement consists of one or more block statements
- The execution of a block statement in a branching statement is controlled by a boolean expression which we call a condition.
- There are manly the following four types of branching statement.
  - Simple if
  - If-else
  - If-elseif-else
  - switch

-- Simple if Statement ...

- Simple if statement has the following structure:

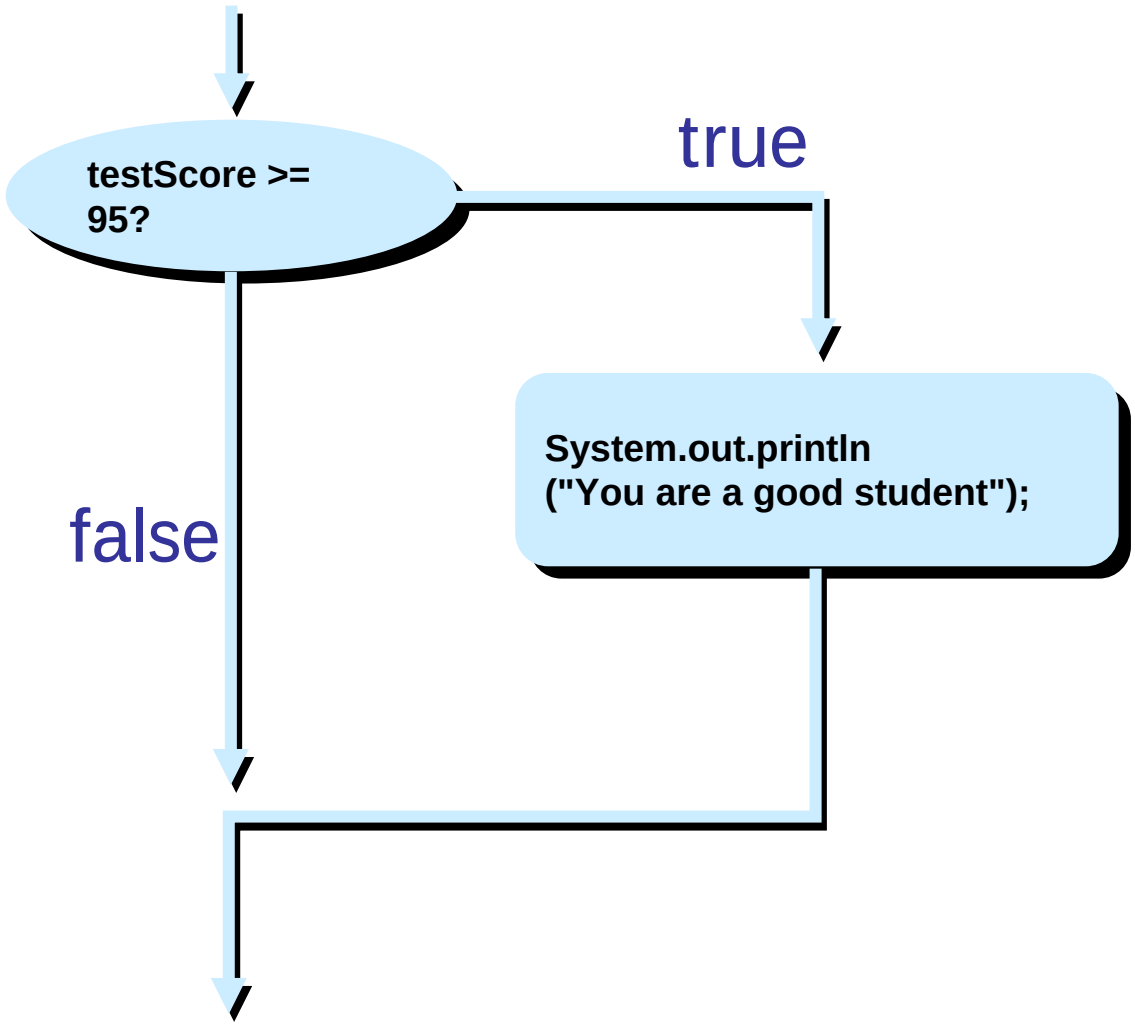
```
if ( <boolean_expression> ) {
   <then_block>
}
```

- The **boolean_expression** must be enclosed in parentheses
- If the **boolean_expression** is **true**, then the **then_block** is executed. Otherwise it will NOT execute.



-- Simple if Statement ...

Control Flow of if:



... -- Simple if Statement

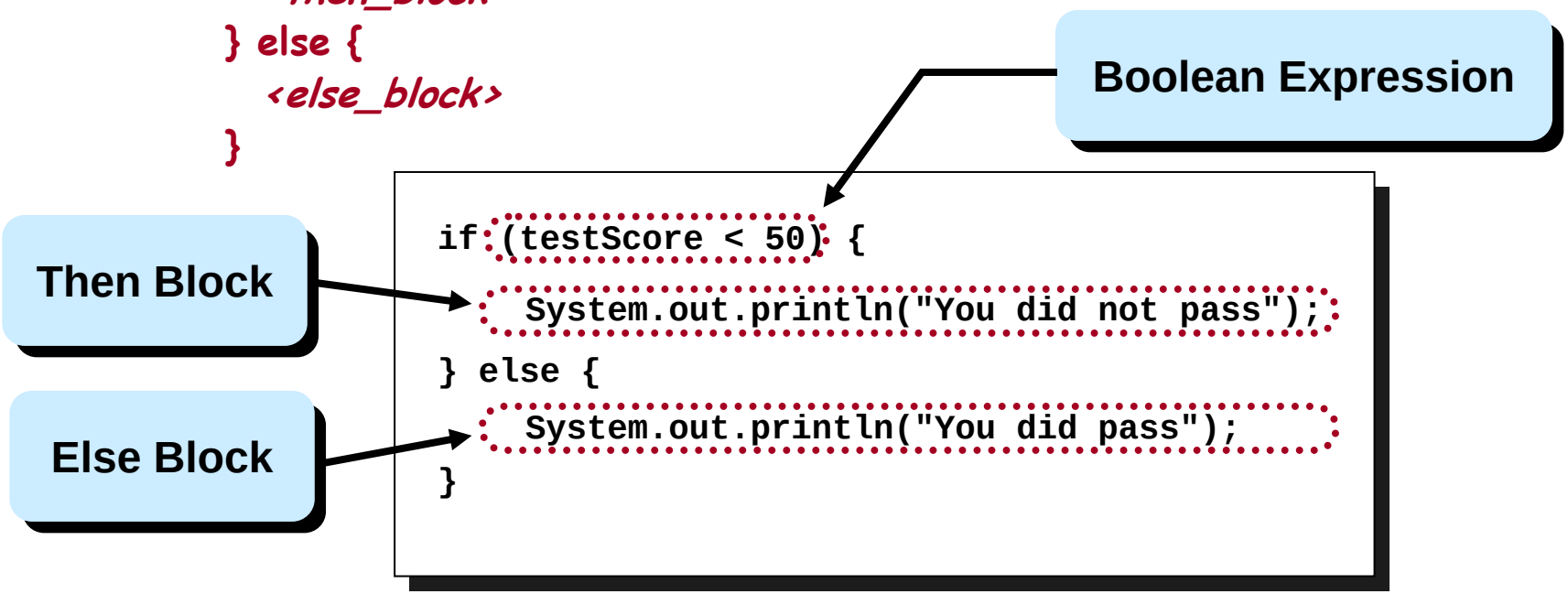
- Example: Design and write a Java program prints the absolute value of a number.

```
import java.util.Scanner;
class absolute {
   public static void main(String [] args) {
      Scanner kb = new Scanner(System.in);
      System.out.print("Enter a number: ");
      double x = kb.nextDouble();
      double y = x;
      if( y < 0)
      {
           y = -y;
      }
      System.out.print("The absolute value of " + x + " is " + y);
   }
}
```

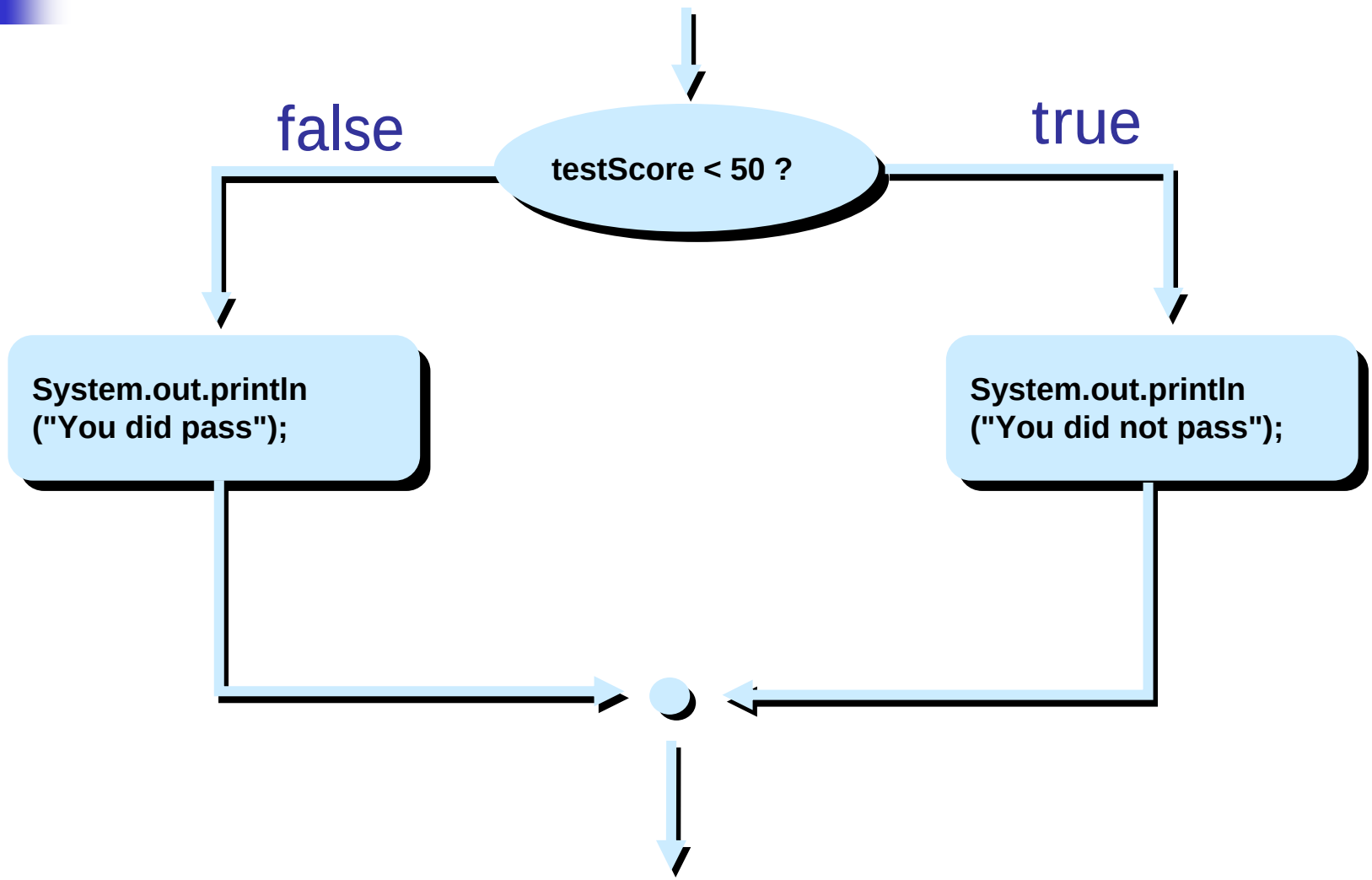
-- if-else Statement ...

- An **if-else** statement chooses between two alternative statements based on the value of a Boolean expression
- If the **boolean_expression** is **true**, then the **then_block** is executed, otherwise the **else_block** is executed.

```
if (<boolean_expression>) {
   <then_block>
} else {
   <else_block>
}
```



-- if-else Statement ...



Compound Statements

- You have to use braces if the <then> or <else> block has multiple statements.
- if only one statement is there braces are optional but it is advisable to always use them to enhance readability

```
if (testScore < 70)
{
    System.out.println("You did not pass");
    System.out.println("Try harder next time");
}
else
{
     System.out.println("You did pass");
     System.out.println("Keep up the good work");
}
```

Then Block

Else Block

... -- if-else Statement

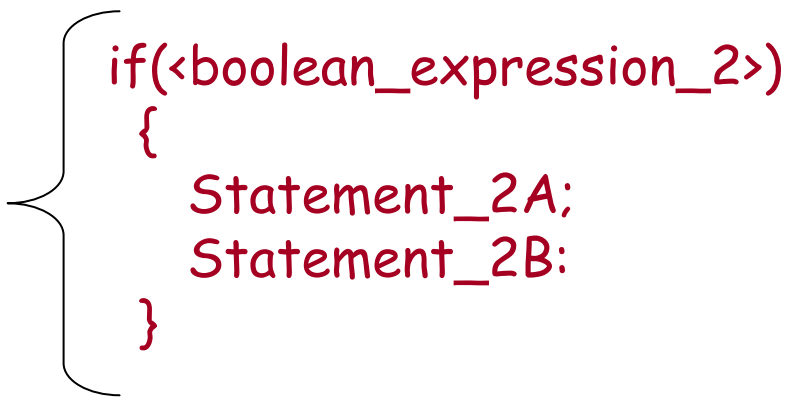
- Design and write a Java program which prints the difference of two numbers.

```
Scanner kb = new Scanner(System.in);
System.out.println("Enter the value of the first number: ");
double first = kb.nextDouble();
System.out.println("Enter the value of the second number: ");
double second = kb.nextDouble();
if ( first > second)
{
        double diff = first – second;
        System.out.println(diff);
}
else
{
        double diff = second – first;
        System.out.println(diff);
}
```

- Nested if Statements ...

- One of the block statements of a branching statement can be another if statement.
- The inner if statement is executed when the enclosing block statement is executed
- If statements can be nested to many levels

```
If(<boolean_expresion_1>)
{
    Statement_1;
    <block_statement_2>;
    Statement_3;
}
```



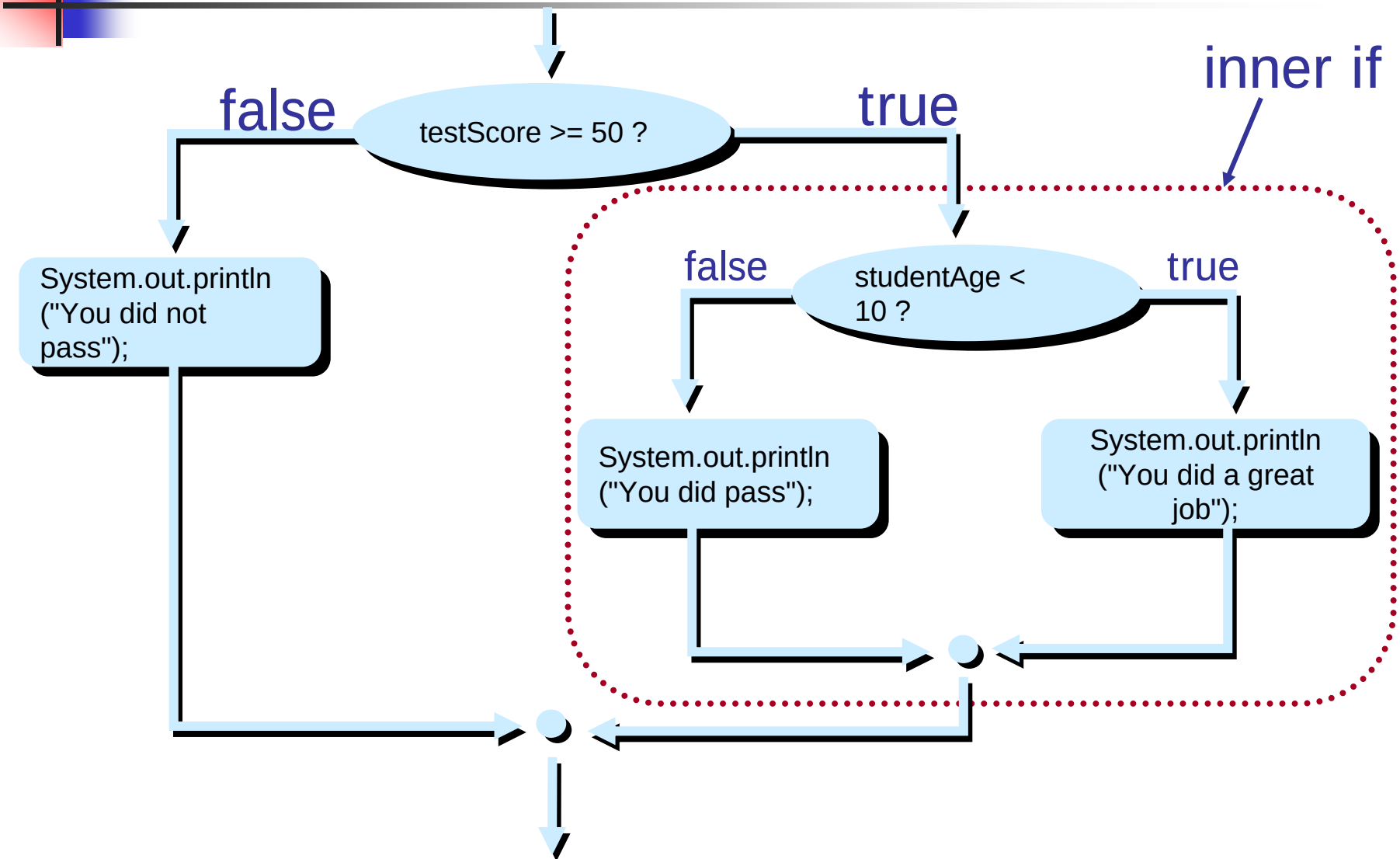
➔ *Statement_2A* and *2B* are only executed if *boolean_expresions 1* and *2* are true.

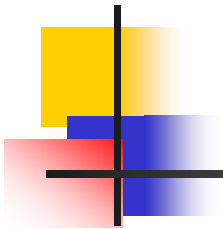
- Nested if Statements ...

```
if (testScore >= 50) {
    if (studentAge < 10) {
        System.out.println("You did a great job");
    } else {
        System.out.println("You did pass");
    }
} else { //test score < 50
    System.out.println("You did not pass");
}
```

Nested if

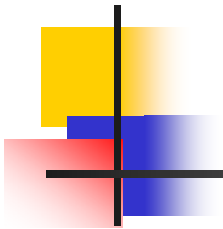
- Nested if Statements ...





-- if-elseif-else Statement ...

- The multiway **`if-else`** statement is simply a normal **`if-else`** statement that nests another **`if-else`** statement at every **`else`** branch
  - It is indented differently from other nested statements
  - All of the **`Boolean_Expressions`** are aligned with one another, and their corresponding actions are also aligned with one another
  - The **`Boolean_Expressions`** are evaluated in order until one that evaluates to **`true`** is found
  - The final **`else`** is optional



... -- if-elseif-else Statement ...

```
if (Boolean_Expression_1)
   Block_Statement_1
else if (Boolean_Expression_2)
   Block_statement_2
                .
                .
                .

else if (Boolean_Expression_n)
   Block_statement_n
else
  Block_statement_For_All_Other_Possibilities
```

if - else- if

```
if (score >= 85) {
   System.out.println(”Grade is A");
} else {
   if (score >= 75) {
      System.out.println(”Grade is B");
   } else {
      if (score >= 65) {
         System.out.println(”Grade is C");
      } else {
         if (score >= 50) {
            System.out.println(”Grade is D");
         } else {
            System.out.println(”Grade is N");
         }
      }
   }
}
```

Test Score	Grade
$85 \leq$ score	A
$75 \leq$ score < 85	B
$65 \leq$ score < 75	C
$50 \leq$ score < 65	D
score < 50	N

if - else- if

```
if (score >= 85) {
   System.out.println(”Grade is A");
} else if (score >= 75) {
   System.out.println(”Grade is B");
} else if (score >= 65) {
   System.out.println(”Grade is C");
} else if (score >= 50) {
   System.out.println(”Grade is D");
} else {
   System.out.println(”Grade is N");
}
```

Test Score	Grade
$85 \leq$ score	A
$75 \leq$ score < 85	B
$65 \leq$ score < 75	C
$50 \leq$ score < 65	D
score < 50	N

Matching else

```
if (x < y)
   if (x < z)
      System.out.println("Hello");
else
   System.out.println("Good bye");
```

really means

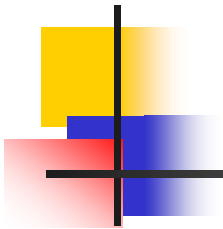
```
if (x < y) {
   if (x < z) {
      System.out.println("Hello");
   } else {
      System.out.println("Good bye");
   }
}
```

Matching else

```
if (x < y) {
   if (x < z)
      System.out.println("Hello");
   } else {
       System.out.println("Good bye");
   }
```

means

```
if (x < y) {
   if (x < z) {
      System.out.println("Hello");
   }
} else {
   System.out.println("Good bye");
}
```

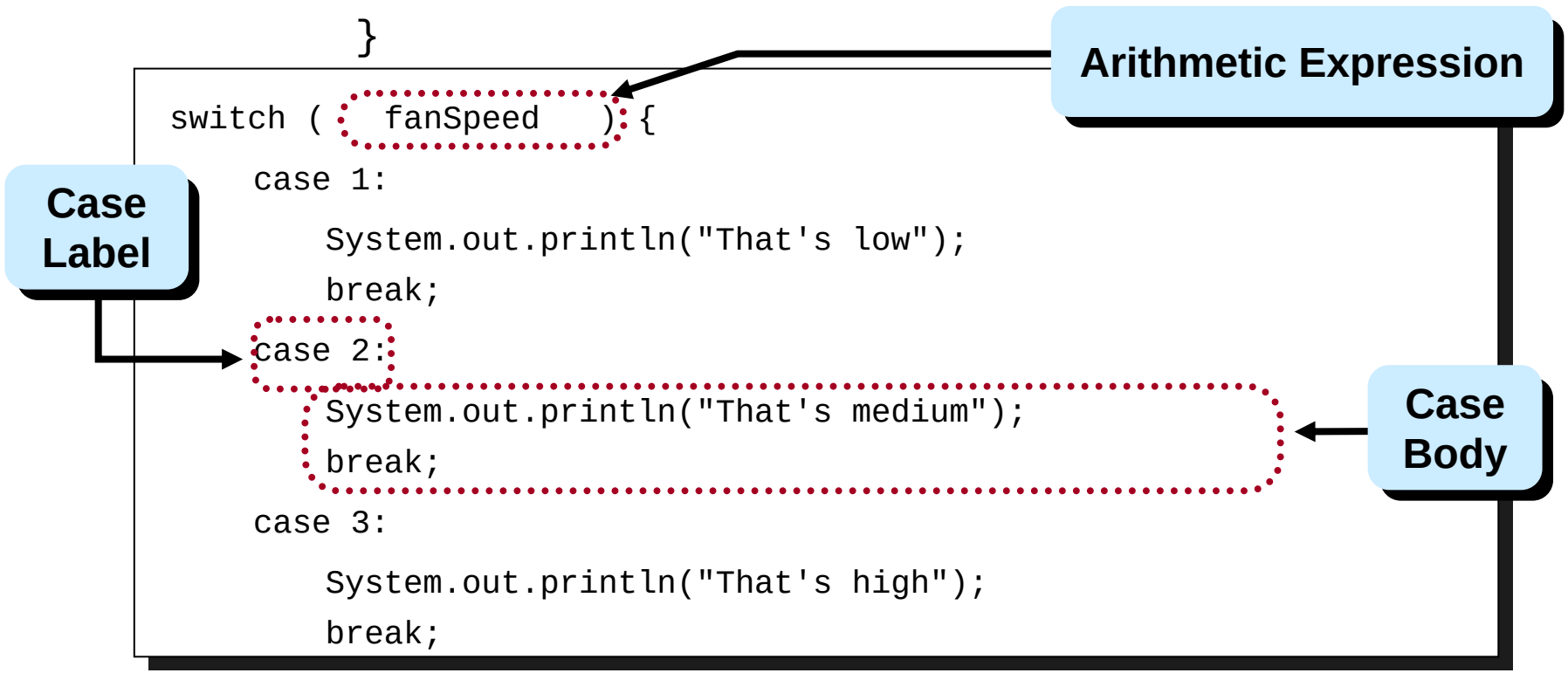


- The **switch** Statement ...

- The **switch** statement is the only other kind of Java statement that implements *multiway* branching
  - When a **switch** statement is evaluated, one of a number of different branches is executed
  - The choice of which branch to execute is determined by a *controlling expression* enclosed in parentheses after the keyword **switch**
    - The controlling expression must evaluate to a **char**, **int**, **short**, or **byte**

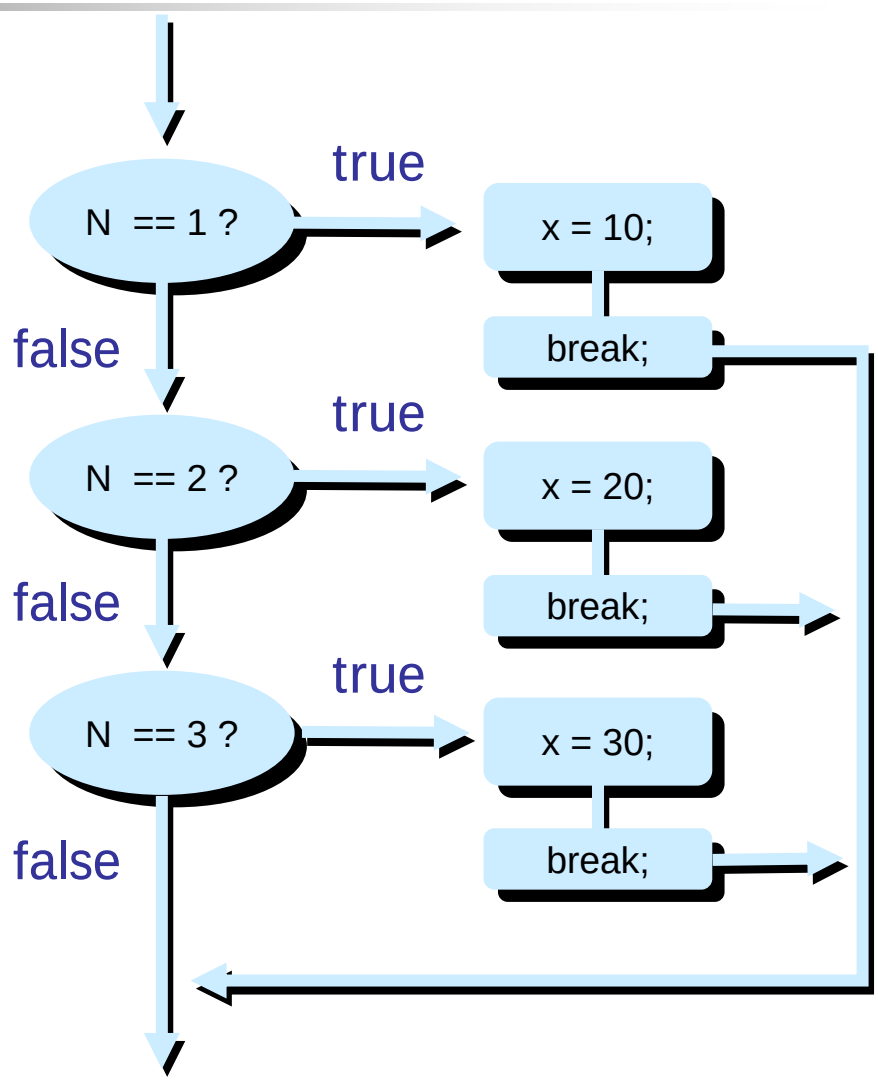
Syntax for the switch Statement

```
switch ( <arithmetic expression> ) {
        <case label 1> : <case body 1>
        …
        <case label n> : <case body n>
}
```



switch With break Statements

```
switch ( N ) {
   case 1: x = 10;
           break;
   case 2: x = 20;
           break;
   case 3: x = 30;
           break;
}
```



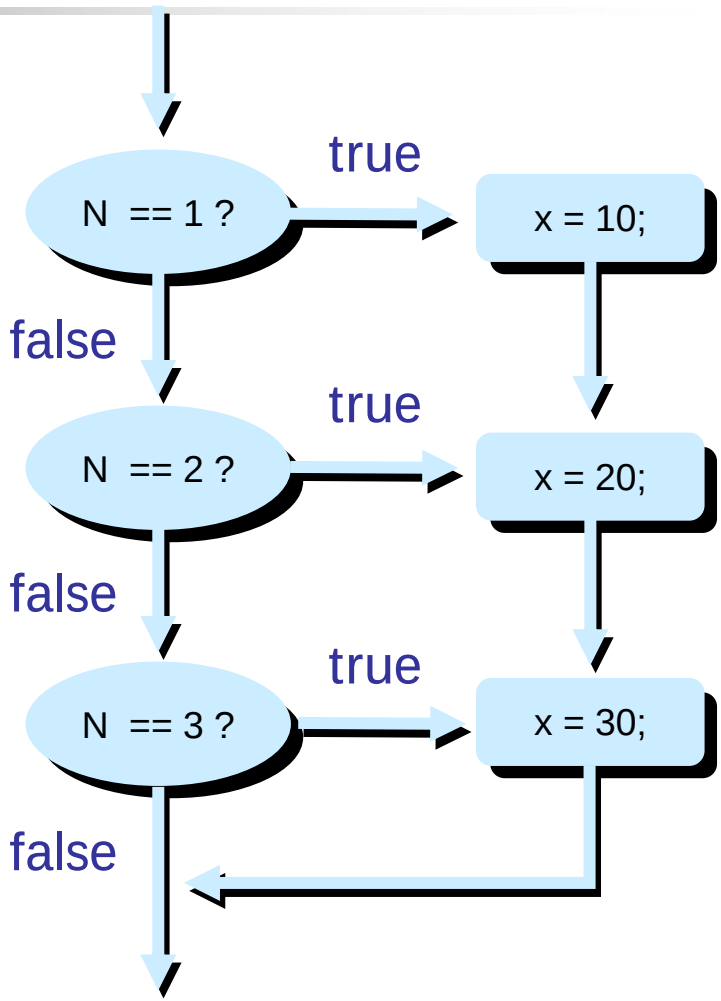
The switch Statement with default

```
switch ( <arithmetic expression> ) {
        <case label 1> : <case body 1>
        …
        <case label n> : <case body n>
        default: <default body>
}
```

```
switch (   binaryDigit   ) {
    case 0:
        System.out.println("zero");
        break;
    case 1:
        System.out.println("one");
        break;
    default:
        System.out.println("That's not a binary digit");
        break;
}
```

switch With No break Statements

```
switch ( N ) {
    case 1: x = 10;
    case 2: x = 20;
    case 3: x = 30;
}
```



... -- The **switch** Statement

```
double y = 30;
double z = 20;
Scanner kb = new Scanner(System.in);
System.out.println("1. add ");
System.out.println("2. Subtract ");
System.out.println("3. Multiply ");
System.out.println("Enter a value: between 1 and 3 ");
int x = kb.nextInt();
switch (x)  {
          case 1:   System.out.println(z + y);
                    break;
          case 2:   System.out.println(z - y);
                    break;
          case 3:   System.out.println(z * y);
                    break;
          default:  System.out.println("Wrong Choice.");
                    break;
}
```

THE END