

Pemrograman Dasar

Control Flow Statements:
Decision Making or Selection

PTIIK - UB



Statements

- * **Control flow statements** regulate the order in which statements get executed.
- * Kinds of control flow statements:
 - Decision making:
 - * if-then statement
 - * if-then-else statement,
 - * switch-case statement
 - Repetition/looping:
 - * while statement
 - * for statement
 - * do-while statement
 - Branching statement:
 - * break statement
 - * continue statement
 - * return statement

if-then statements

- * The most basic of all the control flow statements.
- * Tells your program to execute a certain section of code only if a particular test evaluates to **true**.

```
if (isMoving) {           // the "if" clause
    currentSpeed--;        // the "then" clause
}
```

// or

```
if (isMoving)           // the "if" clause
    currentSpeed--;       // the "then" clause
```

if-then statements

- * Syntax :

```
if (boolean_expression) statement;
```

or

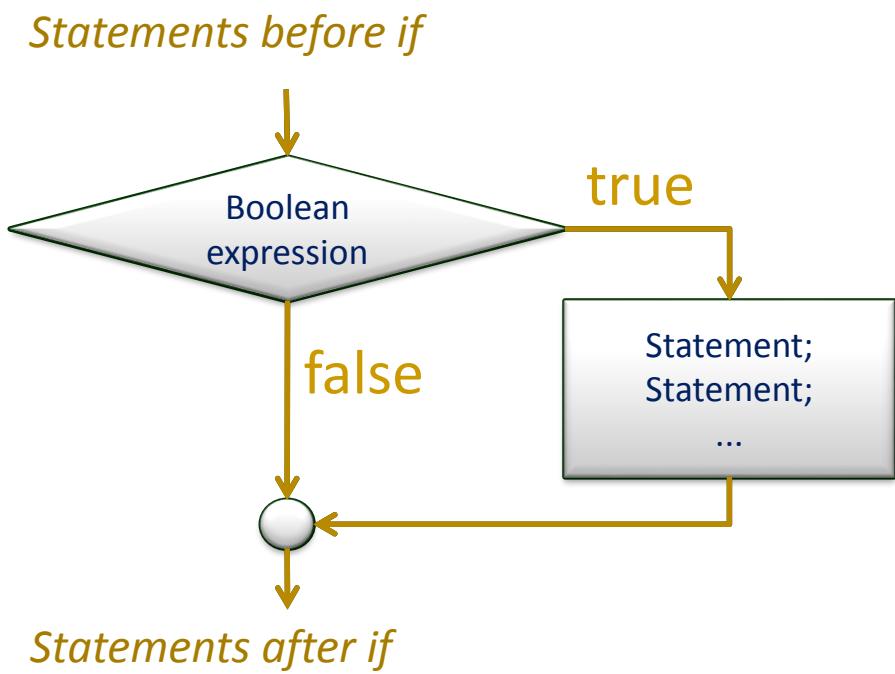
```
if (boolean_expression)
{
    statement1;
    statement2;
    .....
}
```



Block statement

If the *boolean_expression* is true, then the *statement* or block statements are executed.

if-then statements



if-then-else statements



Syntax :

```
if (boolean_expression) statement_01;  
else statement_02;
```

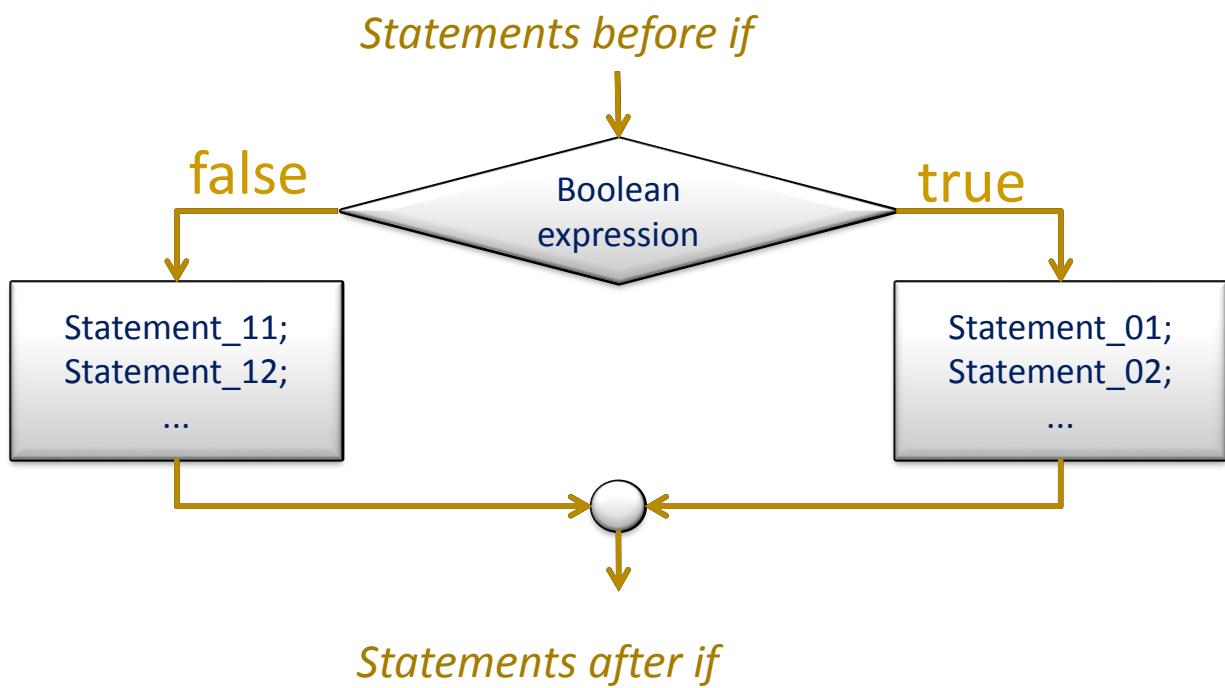
atau

```
if (boolean_expression)  
{  
    statement_01;  
    .....  
}  
else  
{  
    statement_02;  
    .....  
}
```

A diagram illustrating the scope of code blocks. Two curly braces are shown: one spanning from the opening brace of the if-block to the closing brace, labeled "Block statement_01"; and another spanning from the opening brace of the else-block to its closing brace, labeled "Block statement_02".

If *boolean_expression* evaluates to **true**, then *statement_01* or block *statement_01* are executed. But if *boolean_expression* evaluates to **false**, then *statement_02* or block *statement_02* are executed.

if-then-else statements



if-then-else statements

- * if-then-else constructs may have **more than one conditions to evaluate.**
- * Example:

statement_01, statement_02, and statement_03 are of the same level

```
if (boolean_expression) {  
  
    statement_01;  
  
}  
  
else if (boolean_expression) {  
  
    statement_02;  
  
}  
  
else {  
  
    statement_03;  
  
}
```

if-then-else statements

- * if-then-else can be nested
- * Example:

statement_01 and **statement_03** are of the same level, but they are of the different level from **statement_02a** and **statement_02b**

```
if (boolean_expression) {  
  
    statement_01;  
  
}  
  
else if (boolean_expression) {  
  
    if (boolean_expression) statement_02a;  
  
    else statement_02b;  
  
}  
  
else {  
  
    statement_03;  
  
}
```

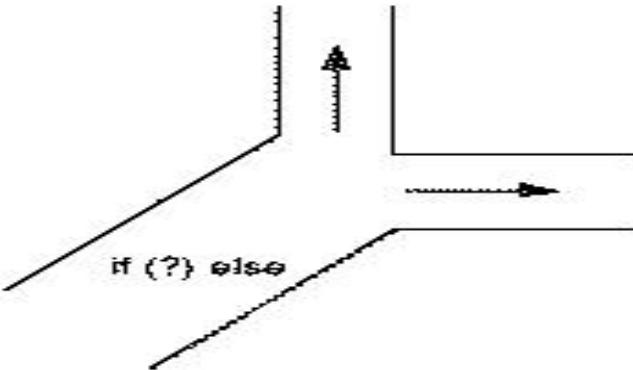
Decision Making or Selection

- * Example:

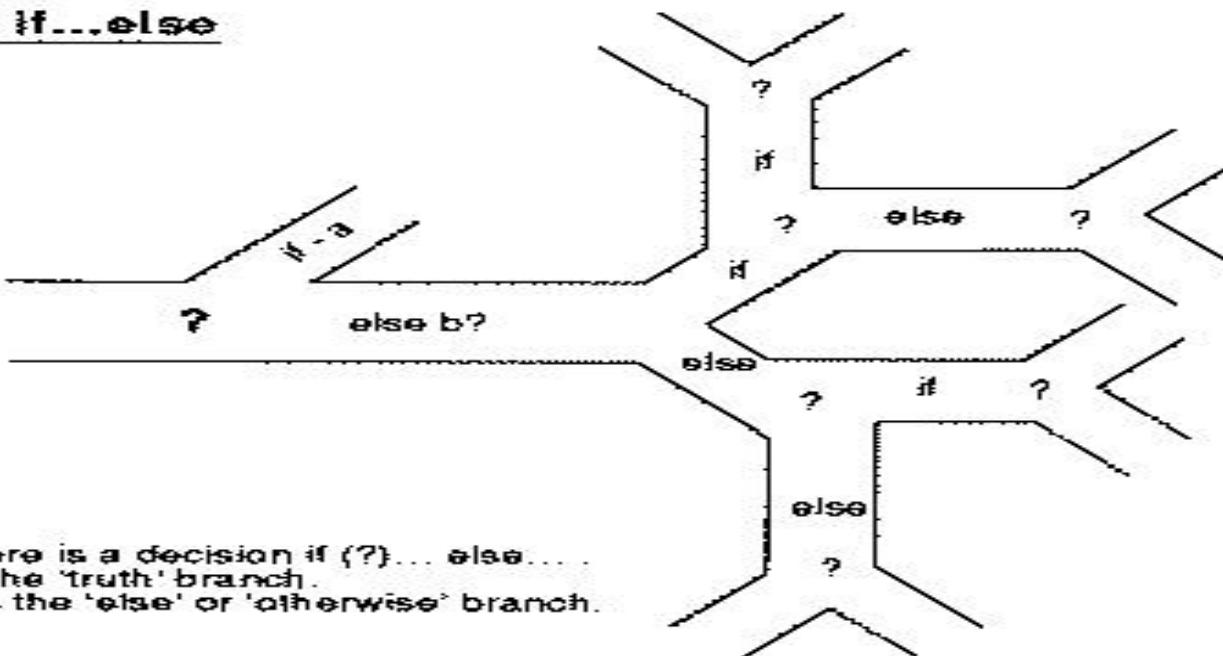
```
if(n > 0)
if(a > b)
z = a;
else
z = b;
```

- *Keyword else berpasangan dengan if yang mana ?*
- *Perbaiki penulisan potongan program di atas agar terbaca jelas algoritmanya!*

if...else



Nested if...else



At each fork there is a decision if (?)... else....
The left fork is the 'truth' branch.
The right fork is the 'else' or 'otherwise' branch.

Figure 17.2. Which route – if...else selects.

switch statements

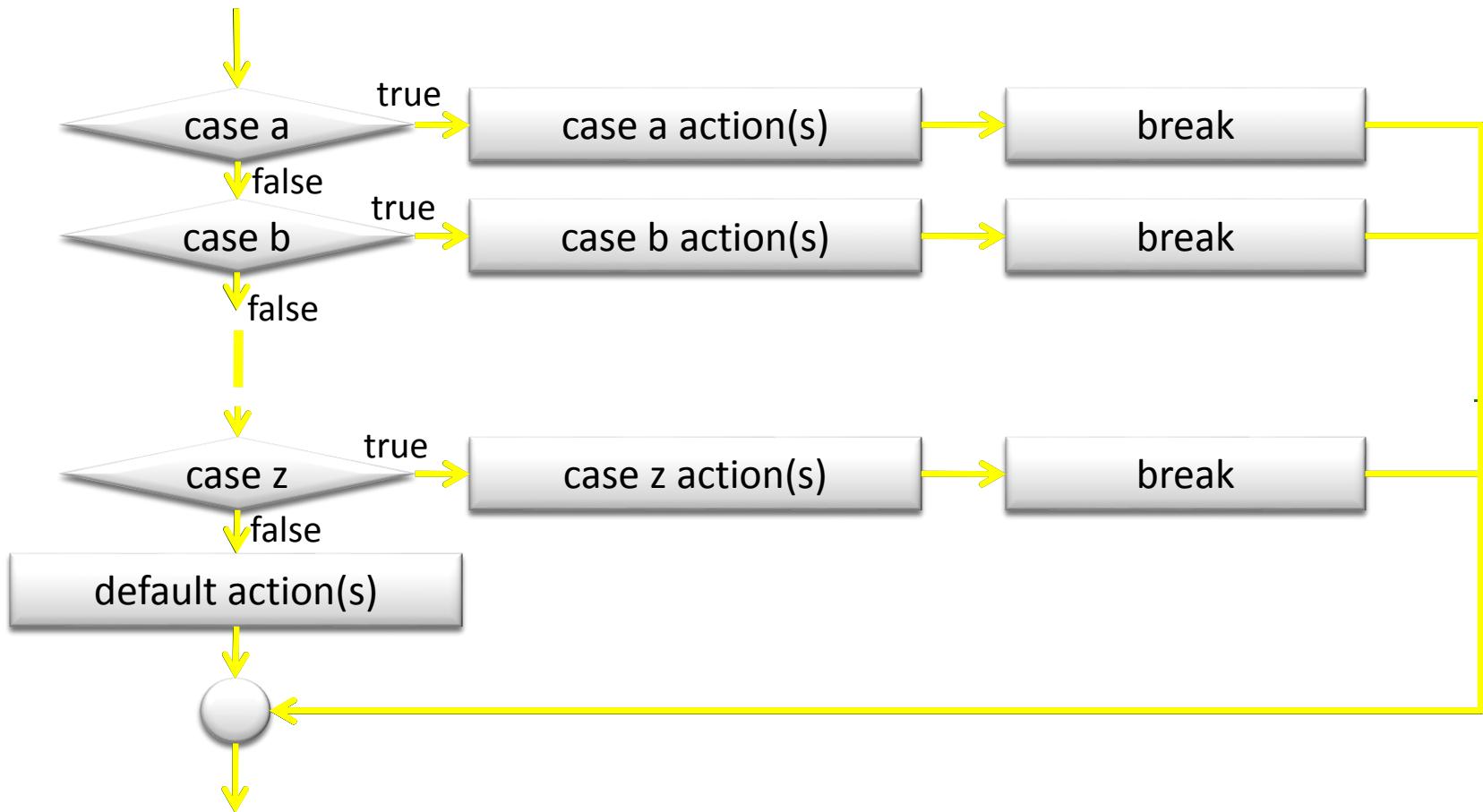
- * **switch** statements can be used as an alternative to **if-then-else** statements.
- * Syntax:

```
switch (expression)
{
    case constant1: statements1;
                        [break;]
    case constant2: statements2;
                        [break;]

    ...
    [default : statements;]
}
```

- * **expression** must be of type char, byte, short, or int, or a corresponding wrapper class, an enum type, or String.

switch statements



switch statements

```
int month = 8;  
switch (month) {  
    case 1: System.out.println("January"); break;  
    case 2: System.out.println("February"); break;  
    case 3: System.out.println("March"); break;  
    case 4: System.out.println("April"); break;  
    case 5: System.out.println("May"); break;  
    case 6: System.out.println("June"); break;  
    case 7: System.out.println("July"); break;  
    case 8: System.out.println("August"); break;  
    case 9: System.out.println("September"); break;  
    case 10: System.out.println("October"); break;  
    case 11: System.out.println("November"); break;  
    case 12: System.out.println("December"); break;  
    default: System.out.println("Invalid month."); break;  
}
```

switch statements

```
float bill, bil2; String op;  
Scanner input = new Scanner( System.in );  
System.out.print("Enter a number: ");  
bill = input.nextFloat();  
System.out.print("Enter an operator: ");  
op = input.next();  
System.out.print("Enter a number: ");  
bil2 = input.nextFloat();  
switch(op){  
    case "+": System.out.println("Result: "+ (bill + bil2)); break;  
    case "-": System.out.println("Result: "+ (bill - bil2)); break;  
    case "*": System.out.println("Result: "+ (bill * bil2)); break;  
    case "/": System.out.println("Result: "+ (bill / bil2)); break;  
    default: System.out.println("Unknown operator");  
}
```

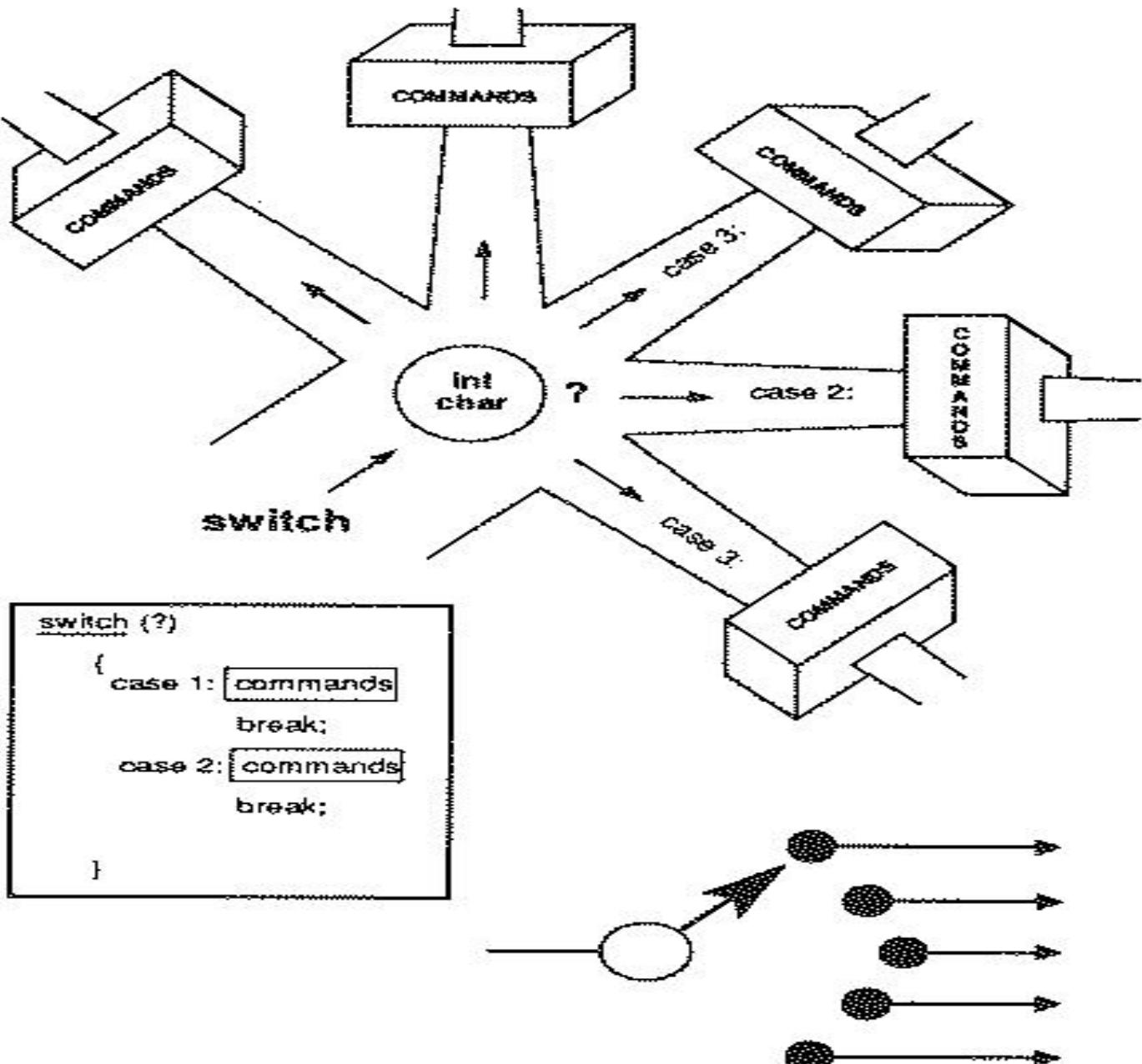


Figure I7.3. `switch`.