



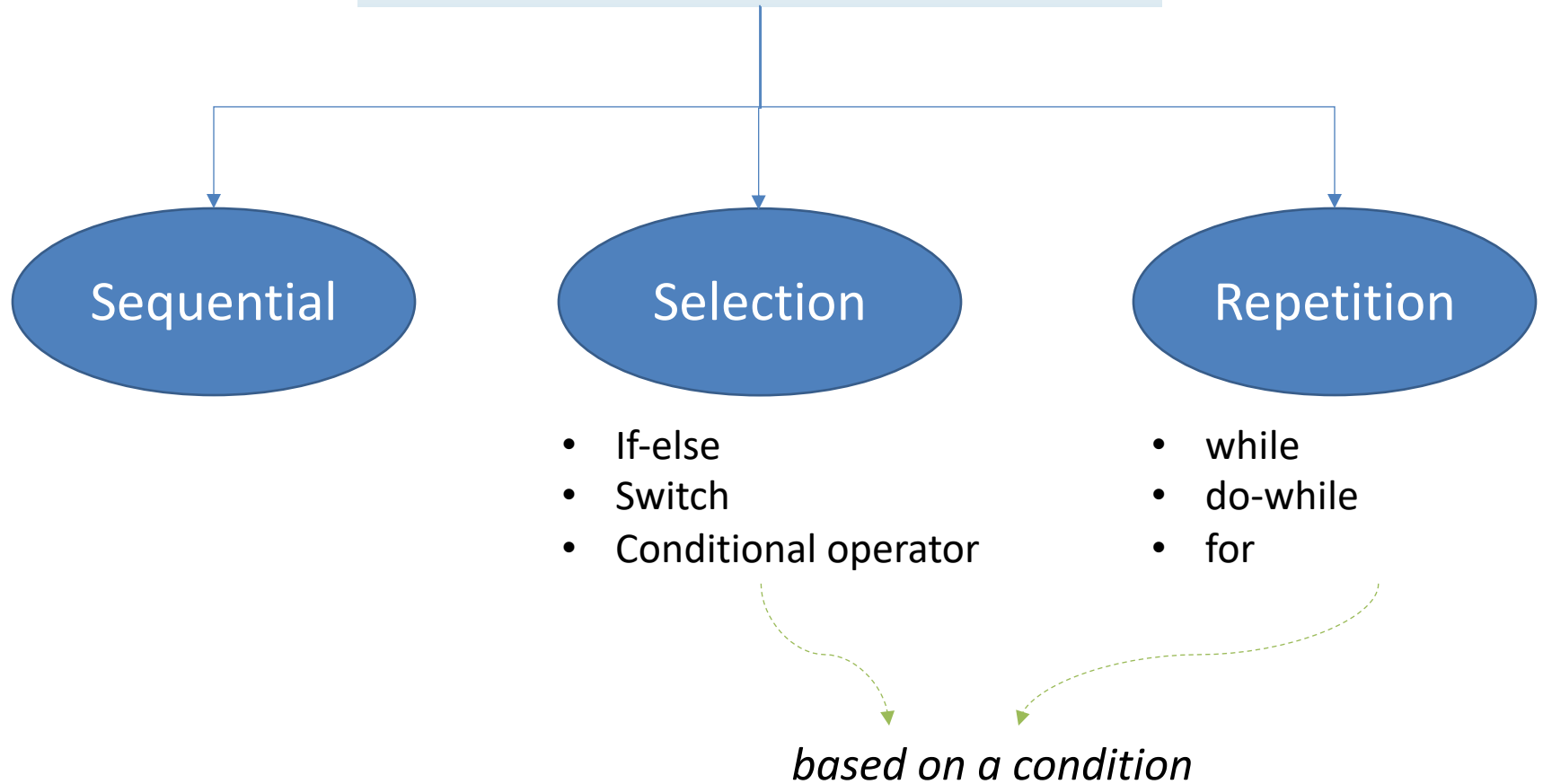
COSC 111

Computer Programming I

Conditionals

Dr. Firas Moosvi

Control Structures



Selection

When you use 'selection', the program can decide which statements to execute based on a condition.

Example:

```
double radius = input.readDouble();

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

Selection statements use **conditions that are Boolean expressions** (evaluate to either true or false)

Boolean Expressions

Relational Operator	Mathematics Symbol	Name	Example (radius is 5)	Result
<	<	less than	<code>radius < 0</code>	<code>false</code>
<=	≤	less than or equal to	<code>radius <= 0</code>	<code>false</code>
>	>	greater than	<code>radius > 0</code>	<code>true</code>
>=	≥	greater than or equal to	<code>radius >= 0</code>	<code>true</code>
==	=	equal to	<code>radius == 0</code>	<code>false</code>
!=	≠	not equal to	<code>radius != 0</code>	<code>true</code>

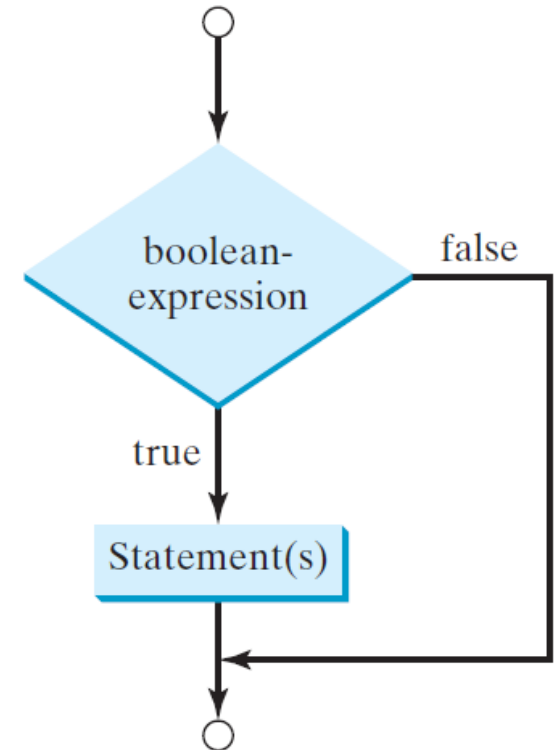
Example:

```
boolean b = (1 < 2);
```

```
System.out.println(b); // displays true
```

One-way if Statements

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

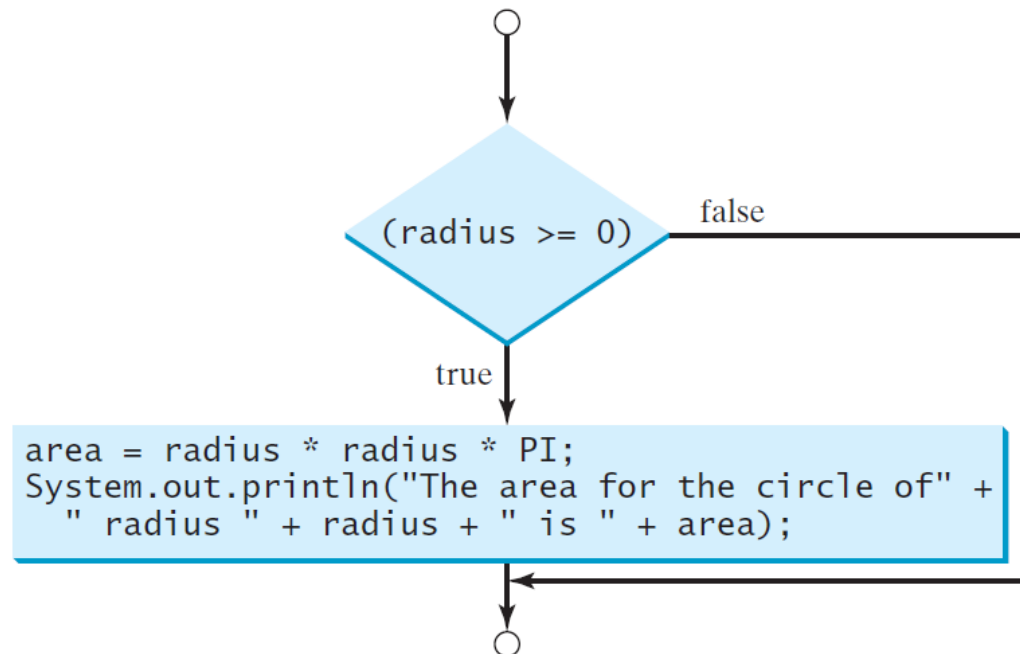


Note: The block braces { } can be omitted if they enclose a single statement.

One-way if Statements

Example:

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



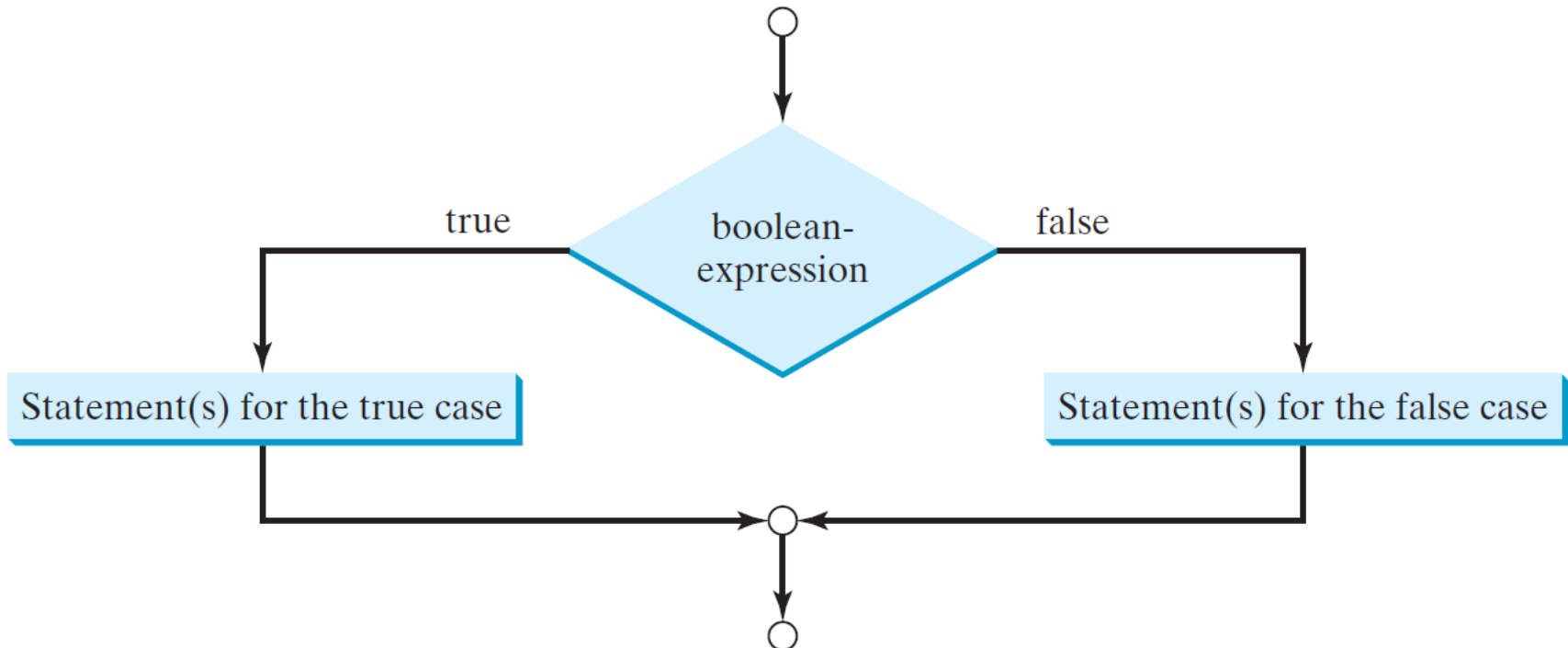
Lecture Activity 4 Task 1

Write a program that prompts the user to enter an integer.

- If the number is a multiple of 5, print HiFive.
- If the number is divisible by 2, print HiEven.
- That is, you could have one of the following three outputs:
 - HiFive
 - HiEven
 - HiFive HiEven

Two-way if Statement

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



Two-way if Statement: Example

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

Lecture Activity 4 Task 2

- If the integer `n` is greater than 0, assign “positive” to a `String` called `type`. If it's anything else, print out a message saying the number is invalid.
- If the integer `n` is larger than or equal to 50, assign "passed" to a `String` called `status` and then print it out.
- If the integer `n` is less than 50, assign "try again!" to a `String` called `status` and then print it out.
- If the integer `n` is greater than 90, increase a `double` called `pay` by 3%.

Practice (Optional)

Create a program to teach a first grade child how to learn subtractions. The program **randomly** generates two single-digit integers number1 and number2 **with number1 \geq number2** and displays a question such as “What is 9 – 2?” to the student. After the student types the answer, the program displays whether the answer is correct.

- **Hint:** *use `Math.random()` to obtain a random double value between 0 and 9 inclusive.*

Algorithm

1. Generate two random single-digit integers
2. Make sure number 1 is larger than number (**How?**:
if number1 < number2, swap number1 with number2)
3. Get student's answer to the question: “what is number1 – number2?”
4. Grade the answer and display the result

TIPS

(1) Forgetting Necessary Braces

- The braces can ONLY be omitted if the block contains a single statement.

```
double radius = 5, area;  
if (radius >= 0)  
    area = radius * radius * 3.14;  
    System.out.println("The area " + " is " + area);  
  
//WRONG!
```

```
double radius = 5, area;  
if (radius >= 0){  
    area = radius * radius * 3.14;  
    System.out.println("The area " + " is " + area);  
}  
  
//CORRECT!
```

TIPS

(2) Wrong Semicolon at the if Line (Logic Error)

- The following two code segments are equivalent.

```
if (radius >= 0);    //LOGICAL ERROR
{
    area = radius * radius * 3.14;
    System.out.println("The area " + " is " + area);
}
```

```
double radius = 5, area;
if (radius >= 0) {};
{
    area = radius * radius * 3.14;
    System.out.println("The area " + " is " + area);
}
```

TIPS

(3) Avoiding Duplicate Code in Different Cases

- The following two code segments are equivalent.

```
if (inState) {  
    tuition = 5000;  
    System.out.println("The tuition is " + tuition);  
} else {  
    tuition = 15000;  
    System.out.println("The tuition is " + tuition);  
}
```

```
if (inState) {  
    tuition = 5000;  
} else {  
    tuition = 15000;  
}  
System.out.println("The tuition is " + tuition);
```

TIPS

(4) Redundant Testing of Boolean Values

- The following two code segments are equivalent.

```
if (even == true)
    System.out.println(
        "It is even.");
```

(a)

Equivalent

```
if (even)
    System.out.println(
        "It is even.");
```

(b)

Clicker Question

What is the output?

```
int x = 10;  
if (x <= 10)  
    System.out.print("A") ;  
else  
    System.out.print("B") ;  
    System.out.print("C") ;
```

A. A

B. B

C. ABC

D. AB

E. AC

Clicker Question

What is the output?

```
int x = 10;  
if (x <= 10)  
    System.out.print("A") ;  
else{  
    System.out.print("B") ;  
    System.out.print("C") ;  
}
```

A. A

B. B

C. ABC

D. AB

E. AC

Nesting If Statements

Nesting if Statement, cont'd

We **nest** `if` statements for more complicated decisions.

- Verify that you use blocks appropriately to group your code!
- Example:

```
if (age > 16) {  
    System.out.print("Adult ");  
    if (gender.equals("male")) {  
        System.out.println("fast ");  
    }else{  
        System.out.println("great ");  
    }  
    System.out.println("driver!");  
}else{  
    System.out.println("Too young to drive.");  
}
```

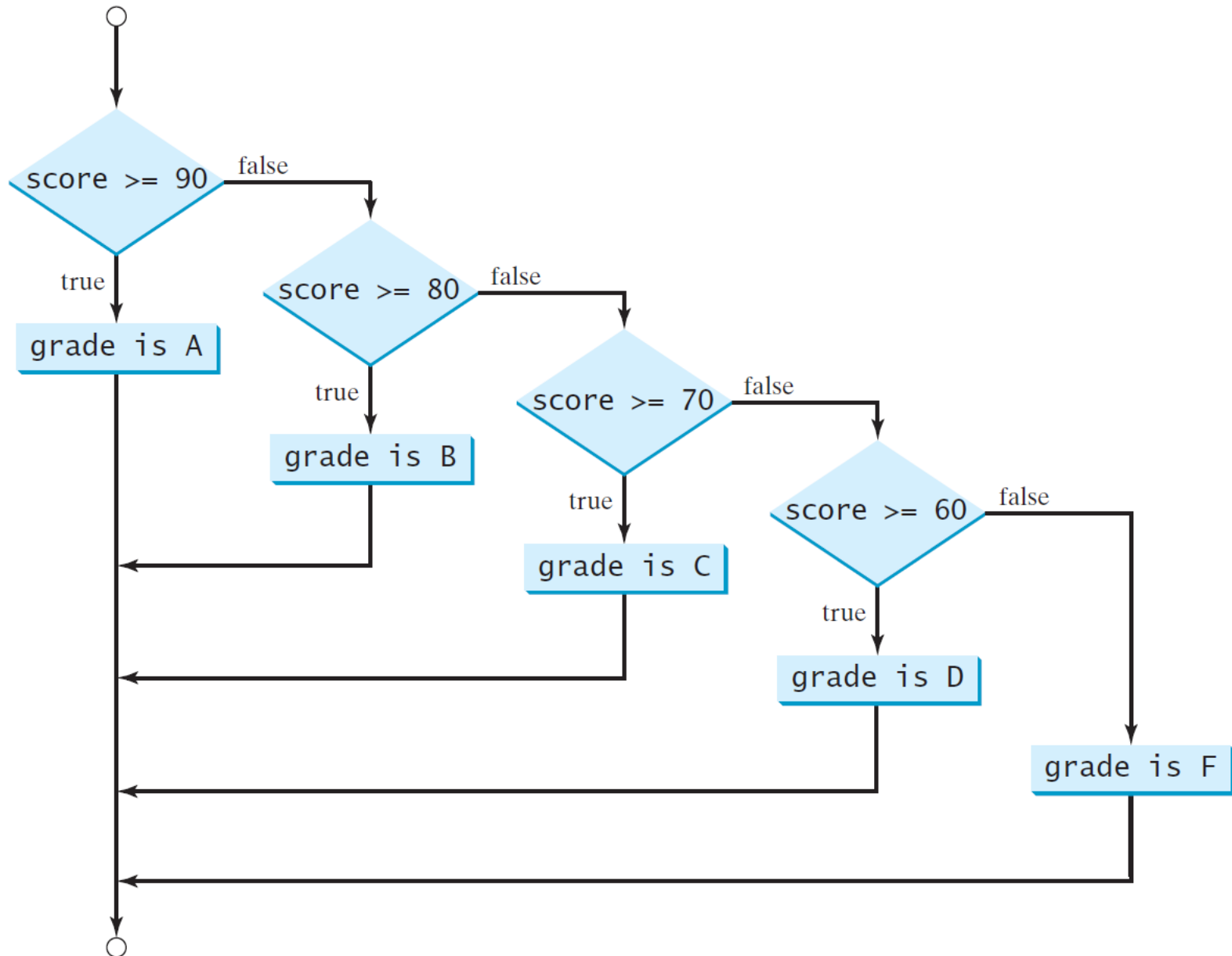
Nesting if Statement, cont'd

```
if (score >= 90.0)
    System.out.print("A");
else
    if (score >= 80.0)
        System.out.print("B");
    else
        if (score >= 70.0)
            System.out.print("C");
        else
            if (score >= 60.0)
                System.out.print("D");
            else
                System.out.print("F");
```

Equivalent

```
if (score >= 90.0)
    System.out.print("A");
else if (score >= 80.0)
    System.out.print("B");
else if (score >= 70.0)
    System.out.print("C");
else if (score >= 60.0)
    System.out.print("D");
else
    System.out.print("F");
```

Nesting if Statement, cont'd



(5) Dangling else Ambiguity

```
int i = 1, j = 2, k = 3;

if (i > j)
    if (i > k)
        System.out.println("A");
else
    System.out.println("B");
```

(a)

Equivalent

This is better
with correct
indentation

```
int i = 1, j = 2, k = 3;

if (i > j)
    if (i > k)
        System.out.println("A");
    else
        System.out.println("B");
```

(b)

Clicker Question

What is the output?

```
int num=12;  
if (num >= 8)  
    System.out.print("A") ;  
    if (num == 10)  
        System.out.print("B") ;  
else  
    System.out.print("C") ;
```

A. A

B. B

C. C

D. AB

E. AC

Logical Operators

Logical Operators

*The logical operators **!**, **&&**, **||**, and **^** can be used to create a **compound Boolean expression**.*

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

NOTE: In Java, the expression **1 <= numberOfDaysInAMonth <= 31** is **INCORRECT**, because **1 <= numberOfDaysInAMonth** is evaluated to a **boolean** value, which cannot be compared with **31**. The correct expression in Java is

(1 <= numberOfDaysInAMonth) && (numberOfDaysInAMonth <= 31)

Truth Tables for Logical Operators

p	!p
true	false
false	true

p ₁	p ₂	p ₁ && p ₂	p ₁ p ₂	p ₁ ^ p ₂
false	false	false	false	false
false	true	false	true	true
true	false	false	true	true
true	true	true	true	false

Clicker Question

What is the output?

```
int x = 10, y = 20;  
boolean result = !(x != 10) && (y == 20);  
System.out.println(result);
```

A. true

B. false

Clicker Question

What is the output?

```
int x = 10, y = 20;  
boolean result = (x >= y) || (y <= x);  
System.out.println(result);
```

A. true

B. false

Clicker Question

What is the output?

```
int x = 10, y = 20;
if (x >= 5) {
    System.out.print("bigx ");
    if (y >= 10)
        System.out.print("bigy ");
} else if (x == 10 || y == 15)
    if (x < y && x != y)
        System.out.print("not equal");
```

A. bigx

B. bigy

C. bigx not equal

D. bigx bigy not equal

E. bigx bigy

Lecture Activity 4 Task 3

Let's say a person can get a driving license if they satisfy exactly two conditions:

- Be at least 17 years old.
- Be a current resident of BC.

Write a Java program to check the age (input is an integer) and ask a Yes/No question about whether their living address is in BC (input is character Y or N).

- The program should then inform the user whether they can get a driving license or not.
- The program should either display to the user why they were denied a driving license (if applicable) or congratulate them for their license.

Practice (Optional)

Write a program that prompts the user to enter a number and then checks whether that number is divisible by 2 **and** 3, whether a number is divisible by 2 **or** 3, and whether a number is divisible by 2 **or** 3 **but not both**.

Algorithm:

1. Prompt the user to enter a number
2. Check the number and display the result

Hint: use AND (&), OR (|), and XOR (^) operators

The & and | Operators

&& and ||:

- Shortcut operators
- The parts of an expression containing && or || operators are evaluated only until it's known whether the condition is true or false .

& and |

- the & and | operators **always evaluate both of their operands.**

Practice (Optional)

■ What is the value of x after these expressions?

- `int x = 1;`
`if ((x > 1) & (x++ < 10));` `//x = 2`

- `int x = 1;`
`if ((x > 1) && (x++ < 10));` `//x = 1`

- `int x = 1;`
`if ((1 == x) | (10 > x++));` `//x = 2`

- `int x = 1;`
`if ((1 != x) || (10 > x++));` `//x = 2`

Switch Statement

Control Structures

```
graph TD; A[Control Structures] --> B[Sequential]; A --> C[Selection]; A --> D[Repetition]; C --- E["• If-else<br/>• Switch<br/>• Conditional operator"]; D --- F["• while<br/>• do-while<br/>• for"]
```

Sequential

Selection

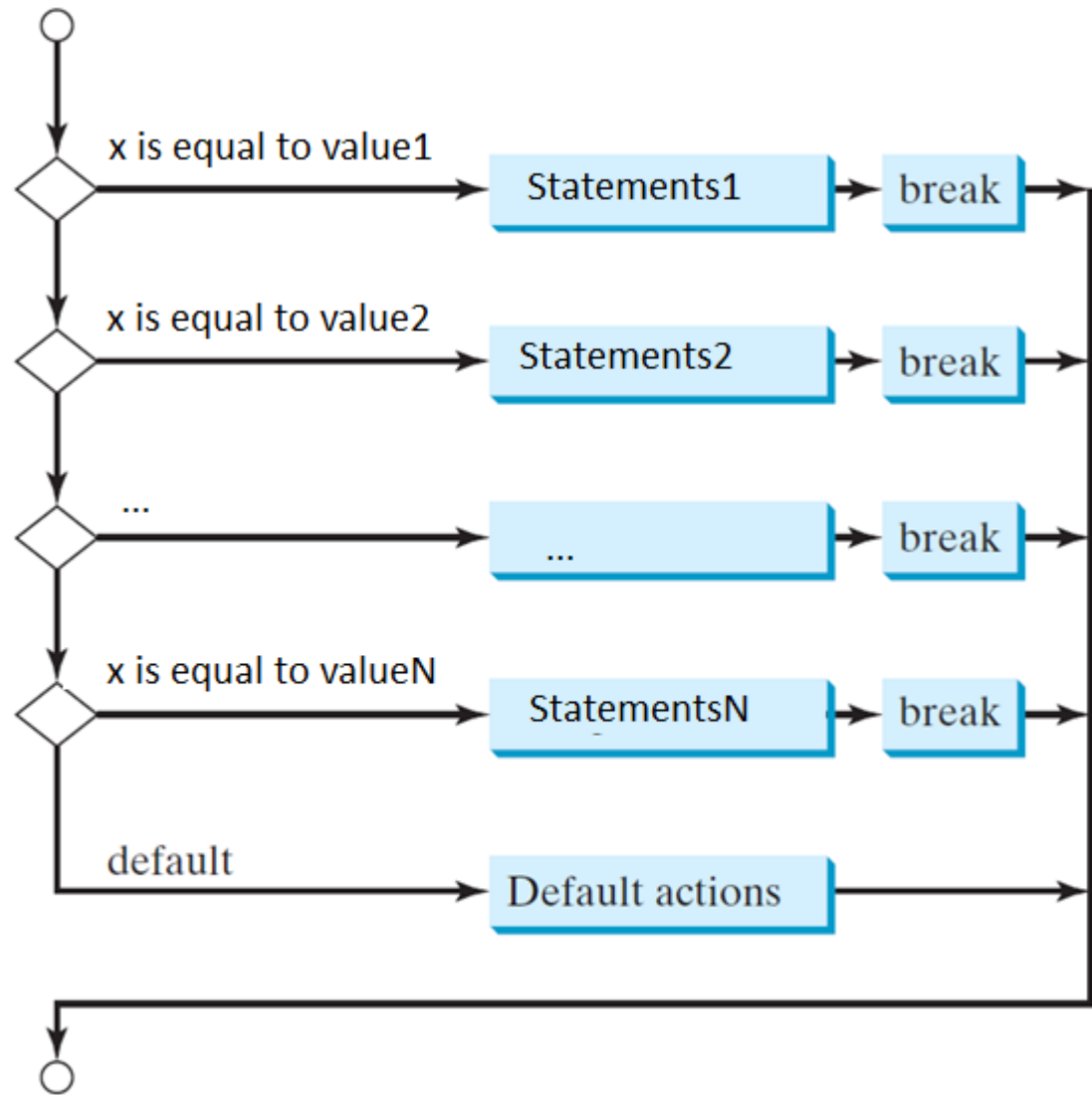
- If-else
- Switch
- Conditional operator

Repetition

- while
- do-while
- for

switch Statement

```
switch (x) {  
  case value1:  
    //statements1;  
    break;  
  case value2:  
    //statements2;  
    break;  
  ...  
  case valueN:  
    //statementsN;  
    break;  
  default:  
    //default actions;  
}
```



switch Statement

The switch-expression (x in the figure) :

- must yield a value of char, byte, short, or int type (also String in JDK7 and beyond)
- must always be enclosed in parentheses.

The values:

- value1, ..., and valueN must have the same data type as the value of the switch-expression.

The case statements

- are executed when the value in the case statement matches the value of the switch-expression.

The keyword **break**:

- It is optional, but it should be used at the end of each case in order to terminate the remainder of the switch statement. If the break statement is not present, the next case statement will be executed.

The default case

- It is optional, and it can be used to perform actions when none of the specified cases matches the switch-expression.

```
switch (x) {  
    case value1:  
        //statements1;  
        break;  
    case value2:  
        //statements2;  
        break;  
    ...  
    case valueN:  
        //statementsN;  
        break;  
    default:  
        //default actions;  
}
```

Clicker Question

What is the output?

```
int num = 2;  
switch (num) {  
    case 2: System.out.print("two "); break;  
    case 1: System.out.print("one "); break;  
    case 3: System.out.print("three "); break;  
    default: System.out.print("other "); break;  
}
```

- A. one
- B. two
- C. three
- D. other

Clicker Question

What is the output?

```
int num = 2;  
switch (num) {  
    case 1: System.out.print("one ");  
    case 3: System.out.print("three "); break;  
    case 2: System.out.print("two ");  
    default: System.out.print("other ");  
}
```

- A. one
- B. one two three
- C. one three two other
- D. two other
- E. one three

Clicker Question

What is the output?

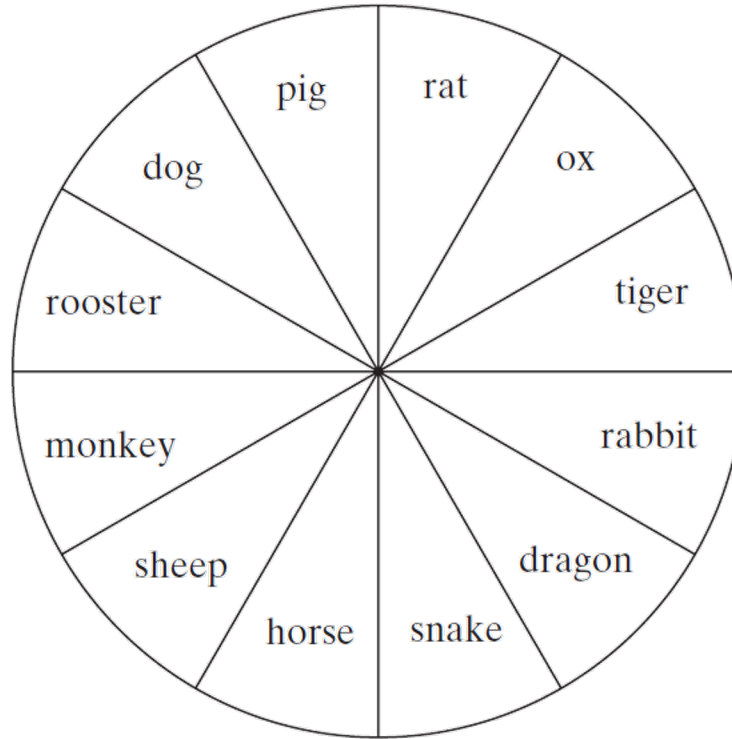
```
int day = 2;  
switch(day) {  
    case 1:  
    case 2:  
    case 3:  
    case 4:  
    case 5: System.out.print("weekday"); break;  
    case 0:  
    case 6: System.out.print("weekend"); break;  
    default: System.out.print("unknown day");  
}
```

- A. weekday
- B. weekend
- C. unknown day
- D. None of the above

Practice (Optional)



Write a program that prompts the user to enter a year and displays the animal for the year.



$$\text{year} \% 12 = \left\{ \begin{array}{l} 0: \text{monkey} \\ 1: \text{rooster} \\ 2: \text{dog} \\ 3: \text{pig} \\ 4: \text{rat} \\ 5: \text{ox} \\ 6: \text{tiger} \\ 7: \text{rabbit} \\ 8: \text{dragon} \\ 9: \text{snake} \\ 10: \text{horse} \\ 11: \text{sheep} \end{array} \right.$$

Algorithm:

1. Prompt the user to enter a year
2. Calculate $\text{year} \% 12$
3. Use conditional statements to display the correct animal.

Extended switch statement (Java 15)

Java 12 -15 included improvements to the switch statement.

Two new forms:

1. Simplified syntax.

- i.e. Combining multiple cases; similar to old switch but more compact

2. As an expression

- i.e. it returns a value, e.g. `variable = switch () {...}`

Note: These improvements were in ***preview mode*** prior Java 15.

- i.e. they are disabled by default, and not guaranteed to be supported in future Java releases.

Extended switch statement, cont'd

1) Simplified syntax (Java 12):

When to use? To simplify several cases referring to the same code.

```
switch (day) {  
    case 2:  
    case 3:  
    case 4:  
    case 5:  
    case 6:  
        System.out.println("weekday");  
        break;  
    case 1:  
    case 7:  
        System.out.println("nice");  
        System.out.println("weekend");  
        break;  
    default:  
        System.out.println("invalid");  
}
```



New Simplified Syntax (Java 15)

Using **:** and **break**

```
switch (day) {  
    case 2,3,4,5,6: System.out.println("weekday");  
                   break;  
    case 1,7:      System.out.println("nice");  
                   System.out.println("weekend");  
                   break;  
    default:       System.out.println("invalid");  
}
```

Extended switch statement, cont'd

2) Switch expression with yield

switch returns (yields) a value.

When to use? to obtain a value based on a condition

```
String dayType;  
switch (day) {  
    case 2:  
    case 3:  
    case 4:  
    case 5:  
    case 6:  
        System.out.println("X");  
        dayType = "weekday";  
        break;  
    case 1:  
    case 7:  
        dayType = "weekend";  
        break;  
    default:  
        dayType = "invalid";  
}
```

New Syntax (Java 15)

As an expression with **:** and **yield**

```
String dayType = switch (day) {  
    case 2,3,4,5,6 :  
        System.out.println("X");  
        yield "weekday";  
    case 1,7:  
        yield "weekend";  
    default:  
        yield "invalid";  
};
```

yield ends the case (like the break) and returns a value

Must include a default case and a semicolon

Aside: using -> with switch

You can replace the colon (:) with (->). In this case:

1. For simplified switch syntax: you don't have to break.

Using -> **WITHOUT** break

```
switch (day) {  
  case 2,3,4,5,6 -> System.out.println("weekday");  
  case 1,7      ->{ System.out.println("nice");  
                  System.out.println("weekend");  
                }  
  default      -> System.out.println("invalid");  
}
```

Use {} for
multiple
statements

No break

2. For switch expressions, you don't have to write yield or break

```
String dayType = switch (day) {  
  case 2,3,4,5,6 -> "weekday";  
  case 1,7      -> "weekend";  
  default      -> "invalid";  
};
```

No yield or break

Conditional Expression (? :)

Control Structures

```
graph TD; A[Control Structures] --> B[Sequential]; A --> C[Selection]; A --> D[Repetition]; C --> E[• If-else]; C --> F[• Switch]; C --> G[• Conditional operator]; D --> H[• while]; D --> I[• do-while]; D --> J[• for];
```

Sequential

Selection

- If-else
- Switch
- Conditional operator

Repetition

- while
- do-while
- for

Conditional Expressions

A conditional expression evaluates an expression based on a condition. Its syntax is:

boolean-expression ? expression1 : expression2;

Example1:

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

is equivalent to:

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


Conditional Expressions

Example2:

```
System.out.println((num % 2 == 0) ? num + "is even" : num + "is odd");
```

is equivalent to:

```
if (num % 2 == 0)
    System.out.println(num + "is even");
else
    System.out.println(num + "is odd");
```

TIPS

Simplifying Boolean Variable Assignment

```
if (number % 2 == 0)
    even = true;
else
    even = false;
```

(a)

Equivalent

```
boolean even
    = number % 2 == 0;
```

(b)

Clicker Question

What is the displayed on the screen?

```
int x = 5, y = 2;  
x > y? "larger": "smaller";
```

- A. larger
- B. smaler
- C. larger smaller
- D. Error

Clicker Question

What is the displayed on the screen?

```
int x = 5, y = 2;  
String s = x > y? "larger": "smaller";  
System.out.println(s);
```

- A. larger
- B. smaler
- C. larger smaller
- D. Error

Practice (Optional)

Rewrite the following **if** statements using the conditional operator.

```
if (x % 2 == 0)
```

```
    s = "even";
```

```
else
```

```
    s = "odd";
```

```
if (y >= 10)
```

```
    x = 100;
```

```
else
```

```
    x = -100;
```

Practice (Optional)

Rewrite the following conditional expressions using **if-else** statements


- `score = (x > 10) ? 3 * scale : 4 * scale;`
- `System.out.println((number % 3 == 0) ? i : j);`

Write conditional expression that returns **-1** or **1** randomly.

Operator Precedence

When two operators share an operand, the operator with the higher *precedence* goes first.

- For example, $1 + 2 * 3$ is treated as $1 + (2 * 3)$ since multiplication has a higher precedence than addition.



<code>var++</code> and <code>var--</code> (Postfix) <code>+</code> , <code>-</code> (Unary plus and minus), <code>++var</code> and <code>--var</code> (Prefix) (type) (Casting) <code>!</code> (Not)	Unary Operations
<code>*</code> , <code>/</code> , <code>%</code> (Multiplication, division, and remainder) <code>+</code> , <code>-</code> (Binary addition and subtraction)	Mathematical Operators
<code><</code> , <code><=</code> , <code>></code> , <code>>=</code> (Relational) <code>==</code> , <code>!=</code> (Equality)	Relational Operators
<code>^</code> (Exclusive OR) <code>&&</code> (AND) <code> </code> (OR)	Logical Operators
<code>=</code> , <code>+=</code> , <code>-=</code> , <code>*=</code> , <code>/=</code> , <code>%=</code> (Assignment operator)	Assignment

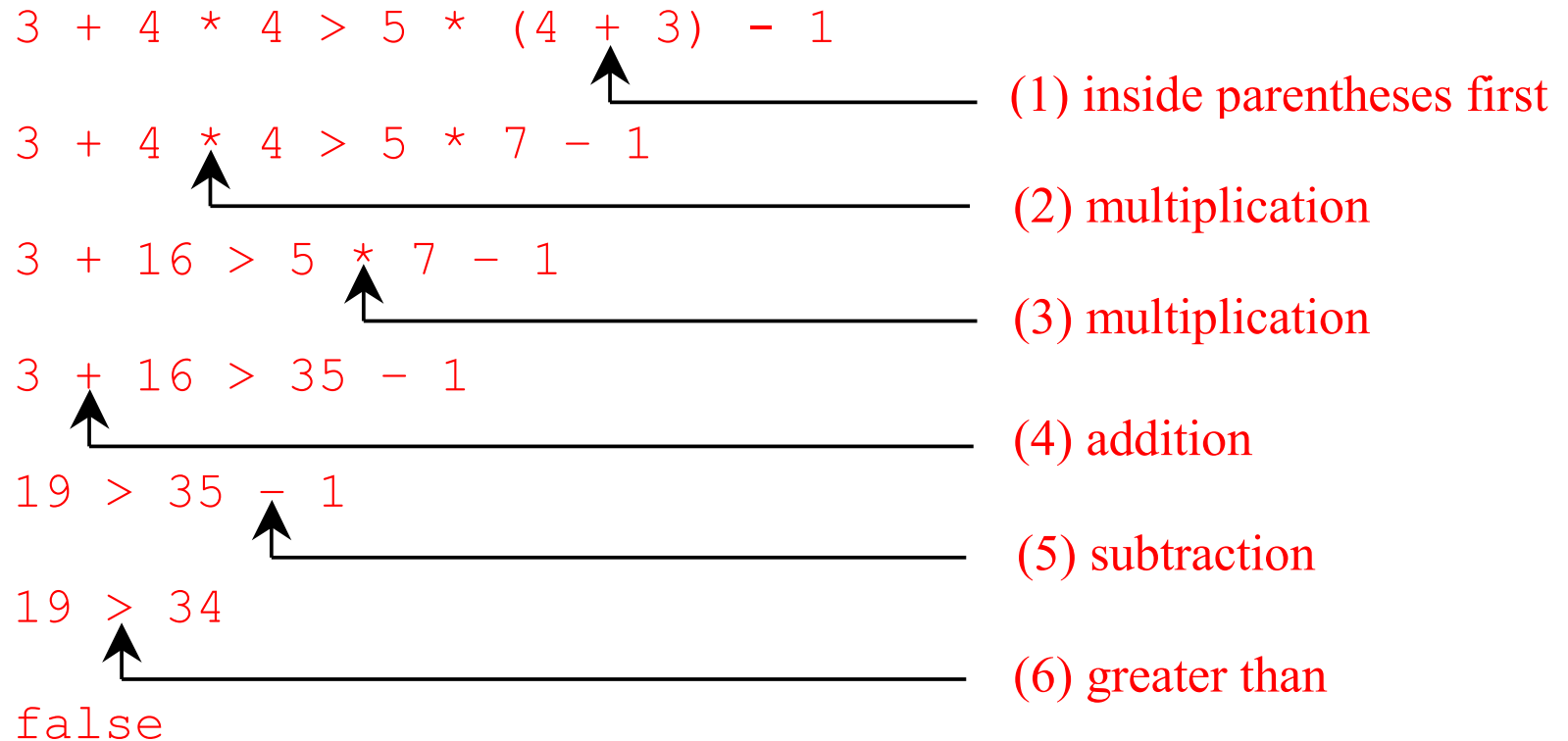
Operator Precedence and Associativity

Notes:

- The expression in the **parentheses is always evaluated first**. When evaluating an expression without parentheses, the operators are applied according to the precedence rule and the associativity rule.
- If operators with the same precedence are next to each other, their associativity determines the order of evaluation.
 - **All binary operators** except assignment operators are **left-associative**.
 $a - b + c - d$ is equivalent to $((a - b) + c) - d$
 - **Assignment operators** are **right-associative**.
 $a = b += c = 5$ is equivalent to $a = (b += (c = 5))$

Example

Applying the operator precedence and associativity rule, the expression $3 + 4 * 4 > 5 * (4 + 3) - 1$ is evaluated as follows:



Practice (Optional)

What is the value of this expression?

■ $3 + 4 * 4 > 5 * (4 + 3) - 1 \ \&\& \ (4 + 3 > 5) ;$

Answer: apply the precedence rules as follows:

■ Parentheses $3 + 4 * 4 > 5 * 7 - 1 \ \&\& \ (4 + 3 > 5)$

$3 + 4 * 4 > 5 * 7 - 1 \ \&\& \ (7 > 5)$

$3 + 4 * 4 > 5 * 7 - 1 \ \&\& \ \text{true}$

■ Mathematical $19 > 34 \ \&\& \ \text{true}$

■ Relational $\text{false} \ \&\& \ \text{true}$

■ Logical false

Practice (Optional)

Assuming that **x** is **1**, what is the value of.

- **(true) && (3 > 4)**
- **!(x > 0) && (x > 0)**
- **(x > 0) || (x < 0)**
- **(x != 0) || (x == 0)**
- **(x >= 0) || (x < 0)**
- **(x != 1) == !(x == 1)**