Chapter 3

Dr. Sami Albouq

You Will Learn

- To declare boolean variables and write Boolean expressions using relational operators
- To implement selection control using one-way if statements.
- To implement selection control using two-way if-else statements
- To implement selection control using nested if and multi-way if statements
- To avoid common errors and pitfalls in if statements.
- To program using selection statements for a variety of examples

Flow of Control

- Flow control in Java refers to the process of directing the flow of program execution based on certain conditions. It allows the programmer to control the order in which statements are executed and determine which statements should be executed or skipped depending on specific conditions.

- These decisions are based on boolean expressions (or conditions) that evaluate to true or false
- The order of statement execution is called the flow of control

Flow of Control

```
public class Test{
 public static void main (String []args){
        Statemnt 1
        Statemnt 2
        Statemnt 3
        .
        .
        .
        Statemnt n..
 }
}
```

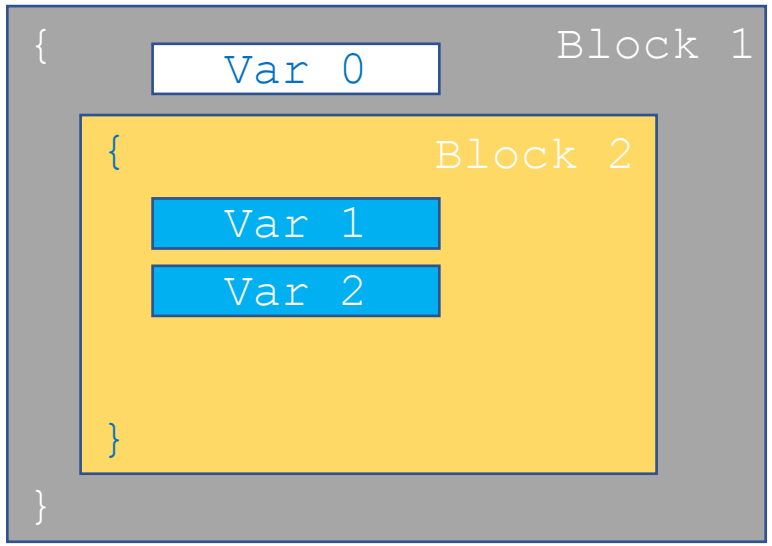
How can I execute or skip this statement ?

```
public class Test{
 public static void main (String []args){

    System.out.println(”The restaurant is”);
    System.out.println(“open”);
    System.out.println(“close”);

  }
}
```

During Ramadan, the restaurant is close

Quick Info About Block in Java

- In Java, a block is a group of zero or more statements enclosed in curly braces {}. A block can be used anywhere in Java where a single statement is allowed.
- A block in Java is used to group statements together to form a single unit. This is often done to define the scope of local variables or to group statements that need to be executed together.



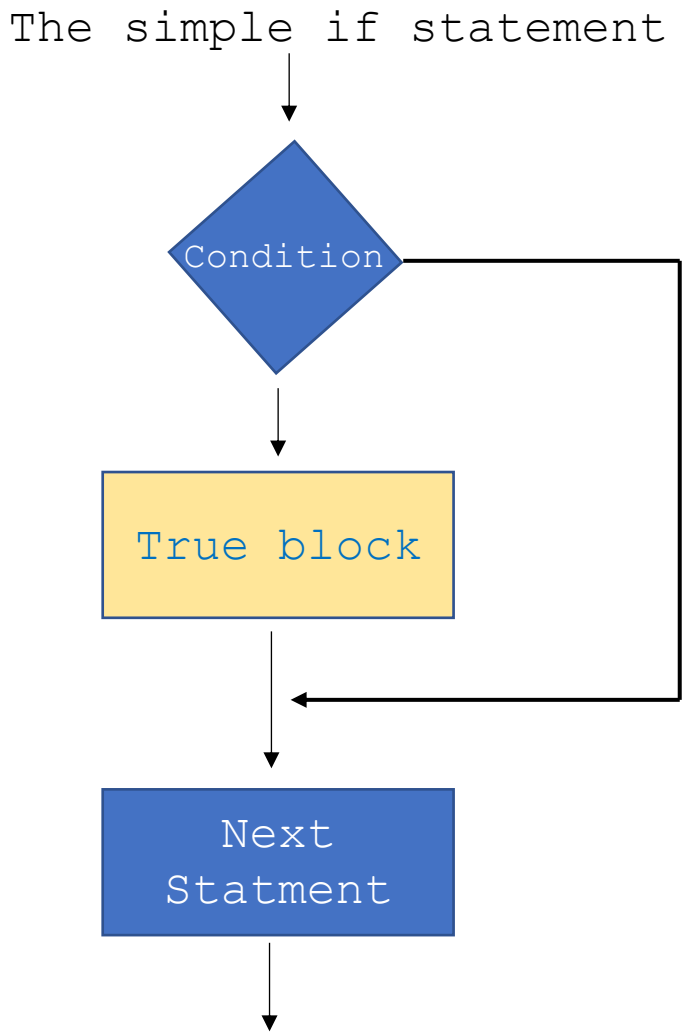

```
{
    int x = 1 ;

    {
      int y = 3;
      x = y;   // acceptable
      y = x  // acceptable

    }                    <-- End of accessibility
    y = x ;              <-- Incorrect
}                        <-- End of accessibility
```

Any variable is declared in this block cannot be access outside the block e.g, variable y. Note x can be used in both blocks.

if Statement

The "if" statement in Java is a conditional statement that allows a programmer to specify a block of code to be executed if a certain condition is true. The basic syntax of an if statement is as follows:

```
if (Boolean expression) {

   // code to be executed if condition is true

}
```



if Statement

If statement block

if is a Java **reserved word** and starts with a small letter

```
if (Boolean expression) {
```

The *condition* must be a boolean expression. It must evaluate to either true or false.

```
    // code to be executed if condition is true
```

If the *condition* is true, the *statement* is executed.
If it is false, the *statement* is skipped.

```
}
```

Boolean Expression

- Boolean expressions are those expressions which return either true or false. Thus, the return type is boolean.
- some other programming languages represent the return value of boolean expressions as integral values (0 or 1)
- Boolean expressions use the below operators:
  - Relational Operators
  - Logical Operators

Boolean Relational Operators

- Boolean relational operators are used to compare two values or expressions and return a Boolean value (either true or false) based on the comparison.

Relational Operators	Type (number of operands)	Meaning
<	binary	is less than
<=	binary	is less than or equal to
>	binary	is greater than
>=	binary	is greater than or equal to
==	binary	is equal to
!=	binary	is not equal to

Boolean Relational Operators

- If int variable num1 holds the value 5:

1. Greater than operator (>):
Returns true if the value on the left is greater than the value on the right, otherwise false.

Example:
- num1 > 3 evaluates to true
- 2 > num1 evaluates to false

2. Greater than or equal to operator (>=):
Returns true if the value on the left is greater than or equal to the value on the right, otherwise false.

Example:
- 5 >= num1 evaluates to true
- 2 >= num1 evaluates to false

3. Less than operator (<):
Returns true if the value on the left is less than the value on the right, otherwise false.

Example:
- 3 < num1 evaluates to true
- num1 < 2 evaluates to false

4. Less than or equal to operator (<=):
Returns true if the value on the left is less than or equal to the value on the right, otherwise false.

Example:
- 3 <= num1 evaluates to true
- num1 <= 2 evaluates to false

Boolean Relational Operators

- If int variable num1 holds the value 5:

5. Equal to operator (==):
Returns true if the value on the left is equal to the value on the right, otherwise false.

Example:
- 5 == num1 evaluates to true
- 2 == num1 evaluates to false

6. Not equal to operator (!=):
Returns true if the value on the left is not equal to the value on the right, otherwise false.

Example:
- 3 != num1 evaluates to true
- num1 != 2 evaluates to true

Logical operators

- Logical operators are used to combine multiple boolean expressions or values and return a single boolean value.
- Operands must be boolean expressions!

Logical Operators	Type (number of operands)	Meaning
!	Unary	NOT
&&	Binary	AND
||	Binary	OR

Truth Tables

- A truth table is a table used in logic to show all possible combinations of inputs and their corresponding outputs in a logical expression.

a	b	a && b	a || b	!a
true	true	true	true	false
true	false	false	true	false
false	true	false	true	true
false	false	false	false	true

Logical operators

o If int variable num1 holds the value 5:

1. NOT operator (!):
Returns the opposite boolean value of the expression. If the expression is true, it returns false, and vice versa.

Example:
- !(num1 > 3) evaluates to false
- !(2 < num1) evaluates to false
- !(num1 < 3) evaluates to true

2. AND operator (&&):
Returns true if both the expressions on the left and right side are true, otherwise false.

Example:
- (num1 > 3) && (2 < 4) evaluates to true
- (num1 > 3) && (4 < 2) evaluates to false

3. OR operator (||):
Returns true if either of the expressions on the left or right side is true, otherwise false.

Example:
- (num1 > 3) || (2 < 4) evaluates to true
- (num1 < 3) || (4 < 2) evaluates to false

Logical operators

Logical operators can also be used in combination with relational operators to create more complex expressions

Example:

- ((5 > 3) || (4 < 2)) && !(5 == 6) evaluates to true
- ((5 < 3) && (4 > 2)) || (7 != 7) evaluates to true

Note use the equality operators only with primitive types and object references, not to compare object data! More on this later…

Negation of Equality and Relational Operators

Expression	!(Expression)
a == b	a != b
a != b	a == b
a < b	a >= b
a <= b	a > b
a > b	a <= b
a >= b	a < b

Basic Boolean Expression Examples

If the total price of the items in the shopping cart is greater than $60, you get a $10 discount.

To convert this into a boolean expression, we can use relational operators to represent the conditions and the conclusion.

```
customer_total_price   = 50.0
MIN_TOTAL_PRICE        = 60.0
get_discount_in$       = 10.0

(customer_total_price > MIN_TOTAL_PRICE )
 => You get a get_discount_in$ ($10.0).
```

Relational Operators

==	equal to
!=	not equal to
<	less than
>	greater than
<=	less than or equal to
>=	greater than or equal to

Basic Boolean Expression Examples

If the total number of items in the shopping cart is greater or equal to 12, and the total price is greater than $60, you get a $10 discount.

To convert this into a boolean expression, we can use relational and logical operators to represent the conditions and the conclusion.

```
customer_total_price   = 50.0
customer_item_count    = 12.0
MIN_TOTAL_PRICE        = 60.0
MIN_TOTAL_ITEMS        = 12
discount_in$           = 10.0
```

Logical Operators

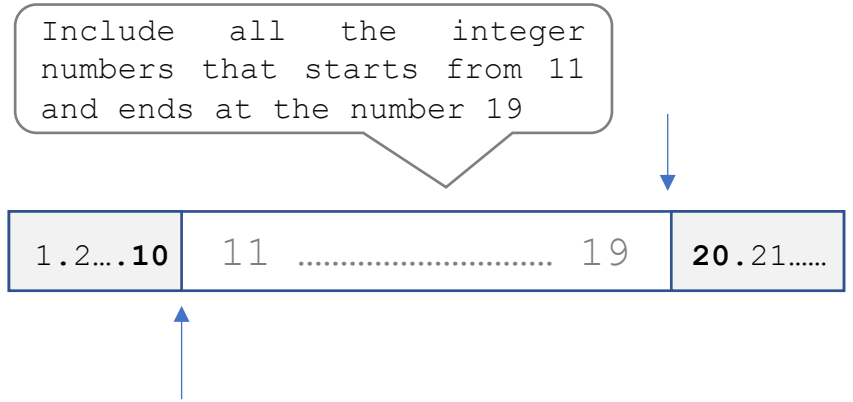
!	Logical NOT
&&	Logical AND
||	Logical OR

```
(customer_item_count >= MIN_TOTAL_ITEMS && customer_total_price > MIN_TOTAL_ PRICE )
=> You get a discount_in$ ($10.0).
```

Basic Boolean Expression Examples

If the total number of items in the shopping cart is between 10 and 20, you get a $10 discount.

```
MAX_TOTAL_ITEMS     = 20
MIN_TOTAL_ITEMS     = 10

customer_item_count = 12
```




```
(customer_item_count  > MIN_TOTAL_ITEMS  && customer_item_count < MAX_TOTAL_ITEMS
)
=> The costumer gets a 10$S.
```

Basic Boolean Expression Examples

If the total number of items in the shopping cart is between 10 and 20, and the total price is greater than $60, you get a $10 discount.

Show me you work now

Basic Boolean Expression Examples

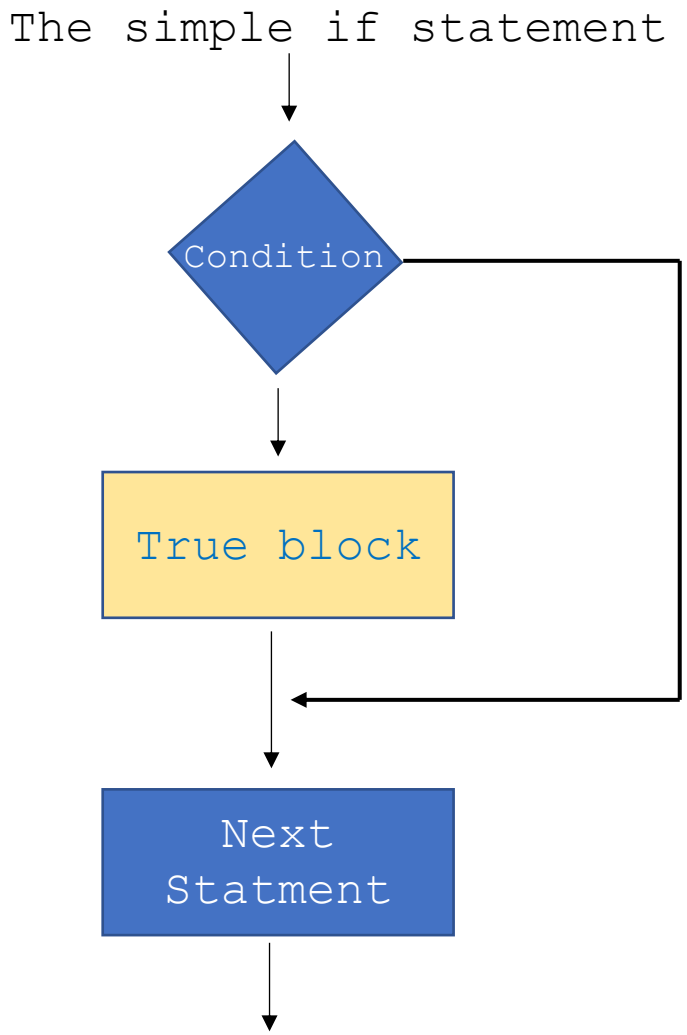
If the total number of items in the shopping cart is between 10 and 20, and the total price is greater than $60, you get a 10% discount.

```
MAX_TOTAL_ITEMS     = 20
MIN_TOTAL_ITEMS     = 10
MIN_TOTAL_PRICE     = 60.0

customer_item_count  = 12
customer_total_price = 50.0

(customer_item_count  > MIN_TOTAL_ITEMS  && customer_item_count < MAX_TOTAL_ITEMS &&
customer_total_price > MIN_TOTAL_PRICE )
=> The costumer gets a $10.
```

if Statement

The "if" statement in Java is a conditional statement that allows a programmer to specify a block of code to be executed if a certain condition is true. The basic syntax of an if statement is as follows:

```
if (Boolean expression) {

   // code to be executed if condition is true

}
```



If Statement

- One of the most common mistakes is using the assignment operator instead of testing equality.

Mistake:

```
if (x = 5);
  statement1;
```

Correct:

```
if (x == 5)
  statement1;
```

- Although logical AND (&&) has a higher precedence than logical OR (||), it is a good idea to use parentheses to aid readability.

```
if ( ((expression) && (expression)) || (expression))
  statement1;
```

Higher Precedence Example

```
System.out.println( false || true && false );

System.out.println( true && false || false );
```

Both lines of code use logical operators || (OR) and && (AND). In Java, && has higher precedence than ||.

1. System.out.println(false || true && false); prints false because && is evaluated first, resulting in false || false, which is false.

2. System.out.println(true && false || false); also prints false because && is evaluated first, resulting in false || false, which is false.

if Statement

If the total price of the items in the shopping cart is greater than $60, you get a $10 discount.

To convert this into a boolean expression, we can use relational operators to represent the conditions and the conclusion.

```
double         customer_total_price   = 50.0;
final  double MIN_TOTAL_PRICE         = 60.0;
double         get_discount_in$       = 10.0;

if(customer_total_price > MAX_TOTAL_PRICE )
 System.out.println(customer_total_price - discount_in$ );
```

if Statement

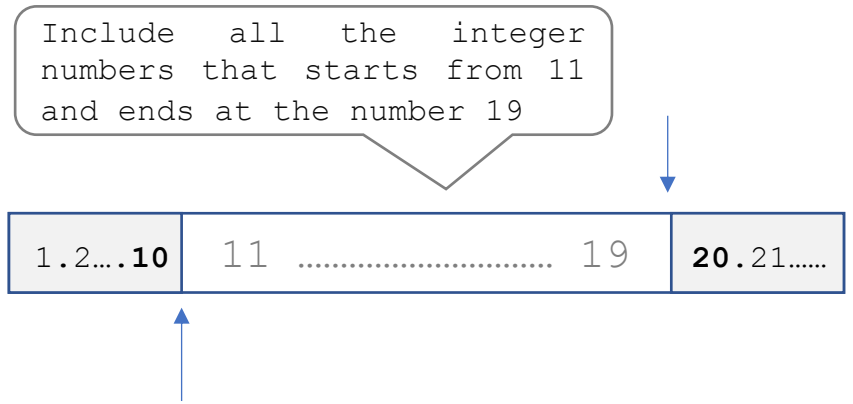
If the total number of items in the shopping cart is greater or equal to 12, and the total price is greater than $60, you get a 10$ discount.

To convert this into a boolean expression, we can use relational and logical operators to represent the conditions and the conclusion.

```
double       customer_total_price = 50.0;
int          customer_item_count  = 12;

final double MIN_TOTAL_PRICE     = 60.0;
final int    MIN_TOTAL_ITEMS     = 12;
double       discount_in$        = 10.0;

if(customer_item_count >= MIN_TOTAL_ITEMS && customer_total_price > MIN_TOTAL_ PRICE )
 System.out.println(customer_total_price - discount_in$ );
```

if Statement

If the total number of items in the shopping cart is between 10 and 20, you get a $10 discount.




```
final int MAX_TOTAL_ITEMS       = 20;
final int MIN_TOTAL_ITEMS       = 10;
double    customer_total_price  = 20.0;
int       customer_item_count   = 12;
double    discount_in$          = 10.0;

if(customer_item_count  > MIN_TOTAL_ITEMS  && customer_item_count < MAX_TOTAL_ITEMS )
 System.out.println(customer_total_price - discount_in$);
```

if Statement Block

```
final int MAX_TOTAL_ITEMS       = 20;
final int MIN_TOTAL_ITEMS       = 10;

int       customer_item_count   = 12;
double    discount_in$          = 10.0;
double    customer_total_price    = 20.0;

if(customer_item_count  > MIN_TOTAL_ITEMS  && customer_item_count < MAX_TOTAL_ITEMS )
 System.out.println(customer_total_price  - 10.0);
System.out.println("You got a gift .. !");
```

The second print statement is not part of the if statement but is in the main method. To add it to the if statement, put it inside a curly brace {} block so that it only runs if the condition is met.

if Statement Block

```
final int MAX_TOTAL_ITEMS      = 20;
final int MIN_TOTAL_ITEMS      = 10;

int         customer_item_count   = 12;
double        discount_in$          = 10.0;
double     customer_total_price     = 20.0;

if(customer_item_count  > MIN_TOTAL_ITEMS  && customer_item_count < MAX_TOTAL_ITEMS )
{
 System.out.println(customer_total_price  - 10.0);
 System.out.println("You got a gift .. !");
}
```

if Statement

If the total number of items in the shopping cart is between 10 and 20, and the total price is greater than $60, you get a $10 discount.

```
final int    MAX_TOTAL_ITEMS     = 20;
final int    MIN_TOTAL_ITEMS     = 10;
final double MIN_TOTAL_PRICE     = 60.0;

int          cutomer_item_count  = 12;
double       cutomer_total_price = 50.0;

if(cutomer_item_count  > MIN_TOTAL_ITEMS  && cutomer_item_count < MAX_TOTAL_ITEMS &&
cutomer_total_price > MIN_TOTAL_PRICE )
System.out.println(coutomer_total_price - 10.0);
```

if Statement

if the number is greater than 0, print “is positive”; otherwise, “is negative”

```
int num1 = 1;
```

if Statement

if the number is greater than 0, print “is positive”; otherwise, “is negative”

```
int num1 = 1; // this could be an input form the keyboard [int num1 = input.nextInt();]

if (num1 > 0)
 System.out.println(“is positive”); //Display: is positive
```

if Statement

if the number is greater than 0, print “is positive”; otherwise, “is negative”

```
int num1 = -2;

if (num1 > 0)
 System.out.println(“is positive”);
```

How about negative numbers?

The evaluation is false, nothing will be printed.

if Statement

if the number is greater than 0, print “is positive”; otherwise, “is negative”

```
int num1 = -2;

if (num1 > 0)
System.out.println(“is positive”);

if (num1 <= -1)
 System.out.println(“is negative”); // Display is negative
```

if Statement

if the number is greater than 0 and even, print “even-positive”;
otherwise, “is negative”

if Statement

if the number is greater than 0 and even, print “even-positive”;
otherwise, “is negative”

```
int num1 = 4;

if (num1 > -1 && num1 % 2 == 0)
System.out.println(“even-positive”);


if (num1 <= -1)
 System.out.println(“is negative”);
```

Nested if Statement

if the number is greater than 0 and even, print “even-positive”;
otherwise, “is negative”

```
int num1 = -2;

if (num1 > 0)
{
 // the number is positive
 // we need to check if it’s even?
}

if (num1 <= -1)
 System.out.println(“is negative”);
```

Nested if Statement

if the number is greater than 0 and even, print “even-positive”;
otherwise, “is negative”

```
int num1 = -2;

if (num1 > 0)
{
    if (num1 % 2 == 0)
    System.out.println(“even-positive”);
}

if (num1 <= -1)
 System.out.println(“is negative”);
```

Nested if statement

Nested if statement

A nested if statement is a conditional statement that is contained within another conditional statement. It allows you to create more complex logical conditions by checking multiple conditions in a specific order.

if-else statements

- If the boolean expression evaluates to true, statement1 is executed. If false, statement2 is executed.
- The else clause is optional.
- If more than one statement is to be executed, the statements must be grouped in a block.

```
if (boolean-expression)
  statement1;
else
  statement2;
```

if-else statements

- When executing multiple statements:
- Statement indentation and placement of curly braces should enhance readability.

```
if (boolean-expression)
{
        statement1;
        statement2;
        statement3;
}
else
{
        statement4;
        statement5;
        statement6;
}
```

if-else Statement

if the number is greater than 0, print “positive”; otherwise, “negative”

Both ways are correct

```
int num1 = -2;

if (num1 > 0)
   System.out.println(“is positive”);
else
   System.out.println(“is negative”);
```

```
int num1 = -2;

if (num1 > 0)
   System.out.println(“is positive”);

if (num1 <= -1)
   System.out.println(“is negative”);
```

if-else Statement

if the number is greater than 0, print “positive”; otherwise, “negative”

```
int num1 = -2;

if (num1 > 0)
  System.out.println(“is positive”);
else
  System.out.println(“is negative”);
```

If the evaluation is true, execute the statement that is inside the if block only

When the if statement evaluation is false, execute this else block only

if-else Statement

To apply for admission to the Islamic University, the applicant must meet certain requirements, including being between the ages of 18 and 25."

```
public class Apply
 public static void main (String[] args)
    {
       final int MAX_AGE = 25;
       final int MIN_AGE = 18;
       Scanner scan = new Scanner (System.in);
       System.out.print ("Enter your age: ");        // get the input form the keyboard
       int age = scan.nextInt();                     // store a student's age in a variable to use it later
       System.out.println ("You entered: " + age); // confirm the entry from the user

       if (age <= MAX_AGE && age >= MIN_AGE)
          System.out.println ("You can apply.");
        else
          System.out.println ("You can not apply");
    }
 }
```

Nested if-else Statement

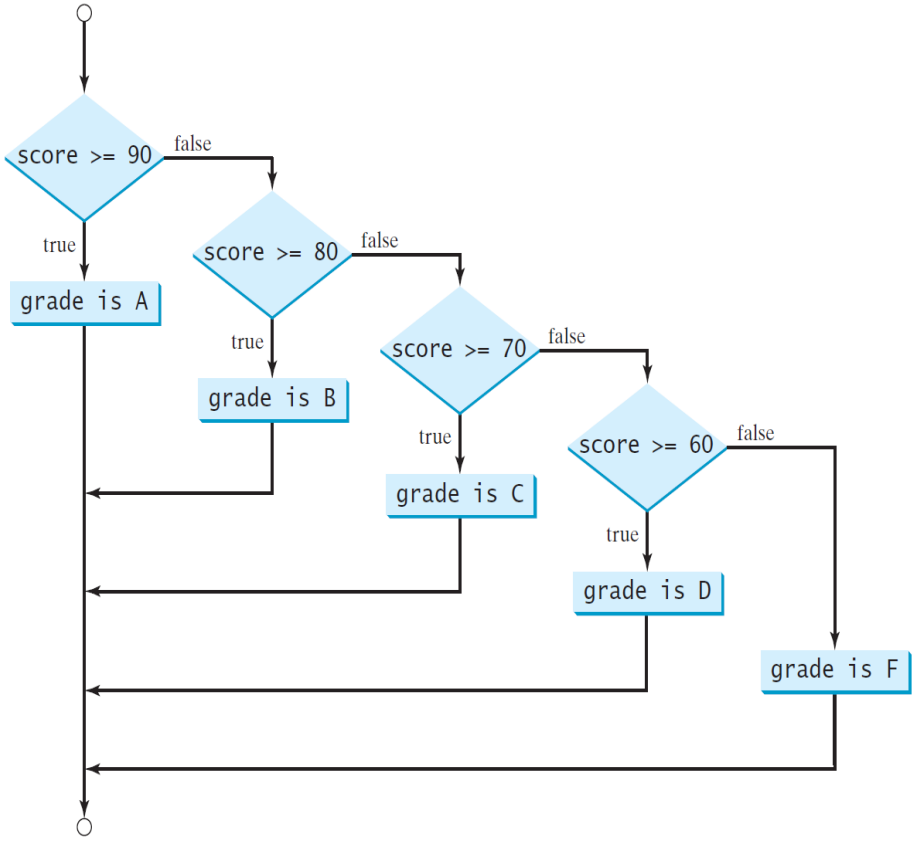
- If statements can be nested. When doing so, it is recommended that curly braces and indentation be used to clearly show the structure of the code.

```
if (boolean-expression)
{
        if (boolean-expression)
        {
                statement1;
        }
        else
        {
                statement2;
        }
}
else
{
        statement3;
}
```

```
if (boolean-expression)
{
        statement1
}
else
{
        if (boolean-expression)
        {
                statement2;
        }
        else
        {
                statement3;
        }
}
```

Nested if-else Statement

```
int score = input.nextInt();
if (score >= 90)
  System.out.println("A");
else
  if (score >= 80)
     System.out.println("B");
  else
     if (score >= 70)
       System.out.println("C");
     else
       if (score >= 60)
         System.out.println("D");
       else
         System.out.println("F");
```



if-else if Statement

- "if-else if" is a conditional statement that allows you to check multiple conditions and execute different blocks of code based on those conditions.

- The "if-else if" statement works by first evaluating the first condition using the "if" statement. If that condition is false, it moves on to the next condition using the "else if" statement. This process continues until either a condition is true, in which case the corresponding code block is executed, or until all conditions have been evaluated and none of them are true, in which case a default code block can be executed using an "else" statement.

if-else if Statement

```
if (condition1)
{
    // code to execute if condition1 is true
}
 else if (condition2)
{
    // code to execute if condition2 is true
}
 else if (condition3)
{
    // code to execute if condition3 is true
}
 else {
    // code to execute if none of the conditions are true
}
```

if-else if Statement

if-else if

```
int score = input.nextInt();
if (score >= 90)
  System.out.println("A");
else if (score >= 80)
  System.out.println("B");
else if (score >= 70)
 System.out.println("C");
else if (score >= 60)
  System.out.println("D");
else
  System.out.println("F");
```

if-else

```
int score = input.nextInt();
if (score >= 90)
  System.out.println("A");
else
  if (score >= 80)
     System.out.println("B");
  else
     if (score >= 70)
       System.out.println("C");
     else
       if (score >= 60)
         System.out.println("D");
       else
         System.out.println("F");
```

if-else if Statement

In this example, the "if-else if" statement checks whether the number is negative, zero, or positive, and prints a different message depending on which condition is true.

```
int num = 10; // this could be [int num = input.nextInt();]
if (num < 0)
{
  System.out.println("The number is negative.");
}
else if (num == 0)
{
   System.out.println("The number is zero.");
}
else
{
    System.out.println("The number is positive.");
}
```

Switch Statement

In Java, a switch statement is a type of conditional statement that allows you to execute different blocks of code based on the value of a variable or expression.

```
switch (variable or expression) {
    case value1:
        // code to execute if the variable or expression equals value1
        break;
    case value2:
        // code to execute if the variable or expression equals value2
        break;
    case value3:
        // code to execute if the variable or expression equals value3
        break;
    default:
        // code to execute if none of the values match the variable or expression
        break;
}
```

Switch Statement

- Often a break *statement* is used as the last statement in each case's statement list
- A break statement causes control to transfer to the end of the switch statement
- If a break statement is not used, the flow of control will continue into the next case
- Sometimes this may be appropriate, but often we want to execute only the statements associated with one case

Switch Statement

- A switch statement can have an optional default case
- The default case has no associated value and simply uses the reserved word default
- If the default case is present, control will transfer to it if no other case value matches
- If there is no default case, and no other value matches, control falls through to the statement after the switch

Switch Statement

- The expression of a switch statement must result in an integral type, meaning an int or a char
- It cannot be a boolean value, a floating point value (float or double), or another integer type
- The implicit boolean condition in a switch statement is equality
- You cannot perform relational checks with a switch statement

Switch Statement

```
int number = input.nextInt();

switch(number) {
    case 1:
        System.out.println("The number is one.");
        break;
    case 2:
        System.out.println("The number is two.");
        break;
    case 3:
        System.out.println("The number is three.");
        break;
    case 4:
        System.out.println("The number is four.");
        break;
    case 5:
        System.out.println("The number is five.");
        break;
    default:
        System.out.println("The number is not in the range of 1 to 5.");
        break;
}
```

Practical Questions

1. Write a Java program that takes three integer inputs from the user and finds the maximum among them. Print the maximum value.

2. Write a Java program that takes two integer inputs from the user and checks whether their sum is greater than 100. If the sum is greater than 100, print "The sum is greater than 100." Otherwise, print "The sum is not greater than 100."

Practical Questions

1. What will be the output of the following Java code segment?

```
int x = 5;
int y = 10;
if (x < y) {
    System.out.println("x is less than y");
} else {
    System.out.println("x is greater than or equal to y");
}
```

2. What will be the output of the following Java code segment?

```
boolean isSunny = true;
if (isSunny) {
    System.out.println("It's a sunny day!");
} else {
    System.out.println("It's not a sunny day.");
}
```

Practical Questions

1. What will be the output of the following Java code segment?

```
int age = 25;
if (age >= 18 && age < 60) {
    System.out.println("You are eligible for voting.");
} else {
    System.out.println("You are not eligible for voting.");
}
```

2. What will be the output of the following Java code segment?

```
int marks = 85;
if (marks >= 90) {
    System.out.println("You scored an A grade.");
} else if (marks >= 80) {
    System.out.println("You scored a B grade.");
} else {
    System.out.println("You scored a C grade.");
}
```

Practical Questions

3. What will be the output of the following Java code segment?

```
int age = 25;
if (age >= 18 && age < 60) {
    System.out.println("You are eligible for voting.");
} else {
    System.out.println("You are not eligible for voting.");
}
```

4. What will be the output of the following Java code segment?

```
int marks = 75;
if (marks >= 90) {
    System.out.println("You scored an A grade.");
} else if (marks >= 70) {
    System.out.println("You scored a B grade.");
} else if (marks >= 50) {
    System.out.println("You scored a C grade.");
} else {
    System.out.println("You scored a D grade.");
}
```

Practical Questions

3. What will be the output of the following Java code segment?

```
int age = 25;
if (age >= 18 && age < 60) {
    System.out.println("You are eligible for voting.");
} else {
    System.out.println("You are not eligible for voting.");
}
```

4. What will be the output of the following Java code segment?

```
int marks = 75;
if (marks >= 90) {
    System.out.println("You scored an A grade.");
} else if (marks >= 70) {
    System.out.println("You scored a B grade.");
} else if (marks >= 50) {
    System.out.println("You scored a C grade.");
} else {
    System.out.println("You scored a D grade.");
}
```

Questions

1. Write a program to check whether a number is negative, positive or zero
2. Write a program to check whether a number is even or odd
3. Write a program that inputs a day number and prints the corresponding day in text.
4. Write a program to calculate the salary as per the following table

Gender	Year of Service	Qualifications	Salary
Male	>= 10	Post - Graduate	15000
	>= 10	Graduate	10000
	< 10	Post - Graduate	10000
	< 10	Graduate	7000
Female	>= 10	Post - Graduate	12000
	>= 10	Graduate	9000
	< 10	Post - Graduate	10000
	< 10	Graduate	6000