303.2.2
Conditional Statements / Flow Control

Control Statement

Learning Objective:

By the end of this lesson, learners should be able to Demonstrate and utilize the Control Statement in Java using If, If-else and switch statements

Outline

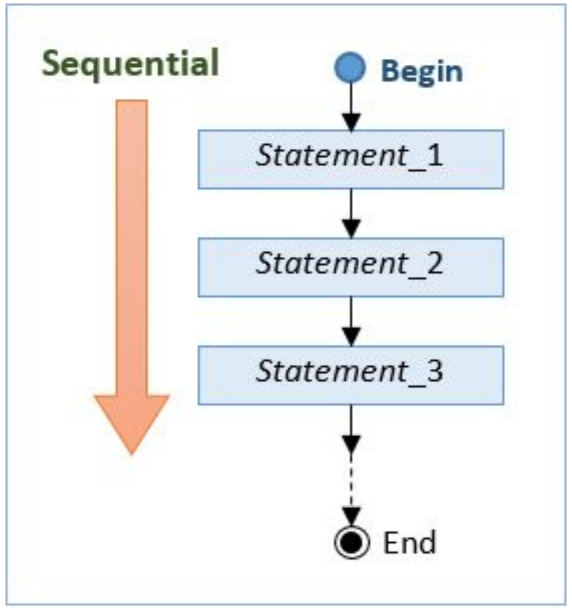
- ❑ Topic 1: Introduction to Control flow statements
- ❑ Topic 1a:Overview - One-way if Statements
- ❑ Topic 1b: Overview - The Two-Way if Statement
- ❑ Topic 1c: Multi-Way if-else Statements
  - ➢ Nested VS Chained - if/else Statements
  - ➢ Use Correct Indentation
  - ➢ Use Curly Braces When Needed
  - ➢ Errant Semicolon
- ❑ Knowledge ChecK
- ❑ Topic 2: Overview - Switch Case Statement

Overview of Control Statements

W will learn and demonstrate about program flow control. We will use several keywords that enable us to control the flow of a Java program. In Java language there are several keywords that are used to alter the flow of the program. Statements can be executed multiple times or only under a specific condition. The if, else, and switch statements are used for testing conditions

Topic 1: Introduction to Control flow statements

- In Java, there are a number of ways we can control the **flow of the program**. Control flow statements, change or break the flow of execution by implementing decision making statements.
- All control flow statements are associated with a business condition – when true, the code block executes; when false it is skipped.
- The decision making statements in Java are:
  - One-way ***if*** Statements
  - Two-Way ***if -else*** Statement
  - Multi-Way ***if-else*** Statements
  - **switch** statement



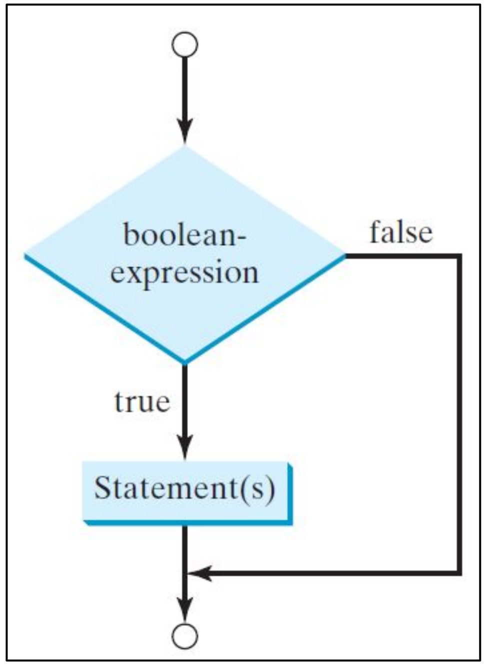
Topic 1a:Overview - One-way *if* Statements

- Practically all Programming languages have some sort of ***if statement***. The ***if statement*** is a **one-way** selection statement.
- You can use the logical operators ***&&, ==, | |, ! (NOT)*** within control expressions in an if statement.

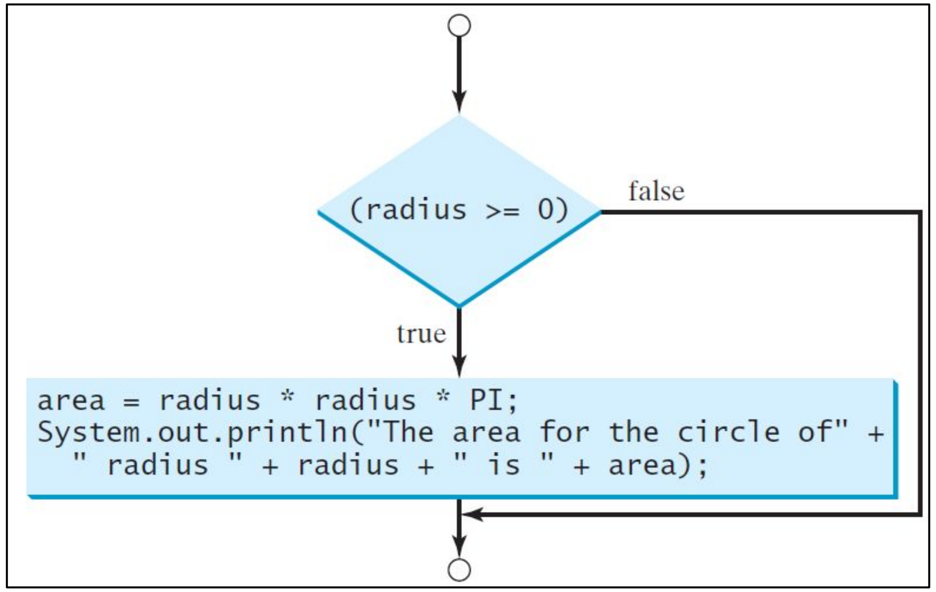
Please view our **Wiki** documentation for more about One-way ***if***-Statements

Basic syntax

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



Example A: One-way *if* Statements



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

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

int radius = 10;
if (radius >= 0) {
  double area = radius * radius * PI;
  System.out.println("The area of a circle of "
    + "radius " + radius + " is " + area);
  } // close if body
  } // close main method body
} // close call body
```

- In this example `(radius >= 0)` is a ***control expression***. The ***control expression*** must be placed in a set of ***parentheses***. If the ***control expression*** is **true** then the statements in the body of the if statement execute. If the ***control expression*** is **false**, then the statements in the body of the if statement are not executed.

Example B: One-way *if* Statements

```
public class OnewayDemo{
   public static void main(String[] args) {
int mark = 80;
if (mark >= 80) {
   System.out.println("Well Done!");
   System.out.println("Keep it up!");
  }  // close if body
System.out.println("Life goes on!");
  } // close main method body
} // close call body
```

Output:

```
Well Done!
Keep it up!
Life goes on!
```

- All if statements contain a ***control expression*** to determine if the statements in the body of the if statement execute or not. In the example above, `(mark >= 80)` is the ***control expression***. The ***control expression*** must be placed in a set of parentheses. If the ***control expression*** is **true** then the statements in the body of the if statement execute. If the ***control expression*** is **false**, then the statements in the body of the if statement are not executed.

One-way *if* Statements - Parentheses are **Required**

- The boolean expression following **if** must be enclosed in parentheses:

```
if i > 0 {
  System.out.println("i is positive");
}
```

(a) Wrong

```
if (i > 0) {
  System.out.println("i is positive");
}
```

(b) Correct

```
if (i > 0) {
  System.out.println("i is positive");
}
```

(a)

Equivalent

```
if (i > 0)
  System.out.println("i is positive");
```

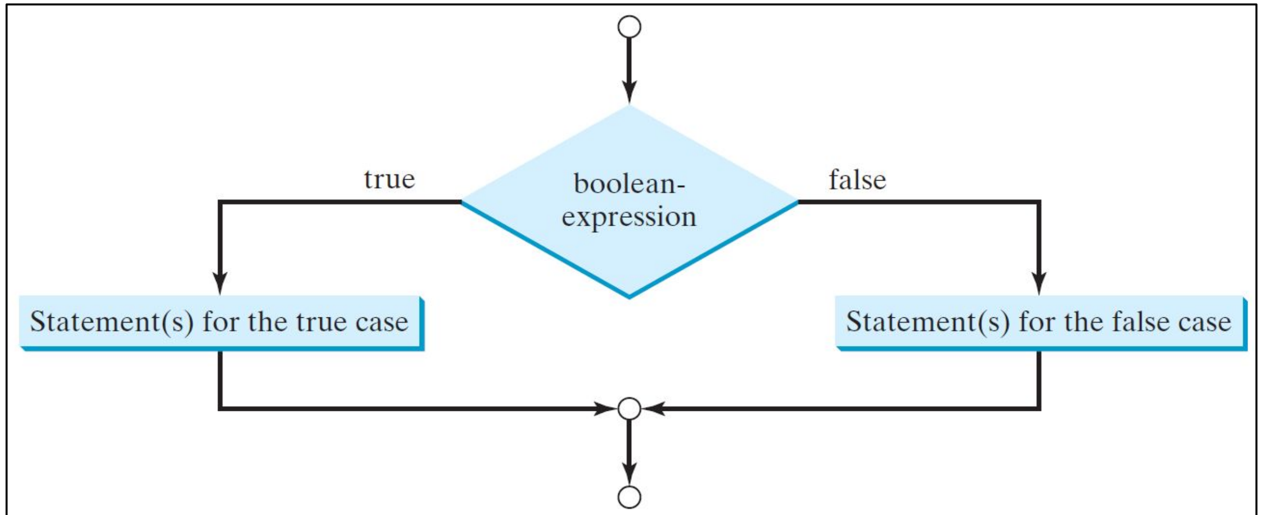
(b)

- When there is only one statement to execute in the conditional, the curly braces are optional.
- If there are multiple statements to execute as a block, the curly braces are essential.

Topic 1b: Overview - The Two-Way *if* Statement

- The ***if else*** statement is a ***two-way*** selection statement since either the block of code after the **"*if*"** part will be executed or the block of code after the **"*else*"** part will be executed.

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


true
boolean-expression
false
Statement(s) for the true case
Statement(s) for the false case

An *if-else* Example

```
public class ifelseDemo {
   public static void main(String[] args) {
 int radius = 10;
 if (radius >= 0) {
     double area = radius * radius * 3.14159;
     System.out.println("The area of a circle of radius" +
          radius + " is " + area);
 } // close if body
 else {
   System.out.println("A negative radius is not possible.");
    } // close else body
   } // close main body
 } // close class body
```

Output

```
The area of a circle of  radius 10 is 314.159
```

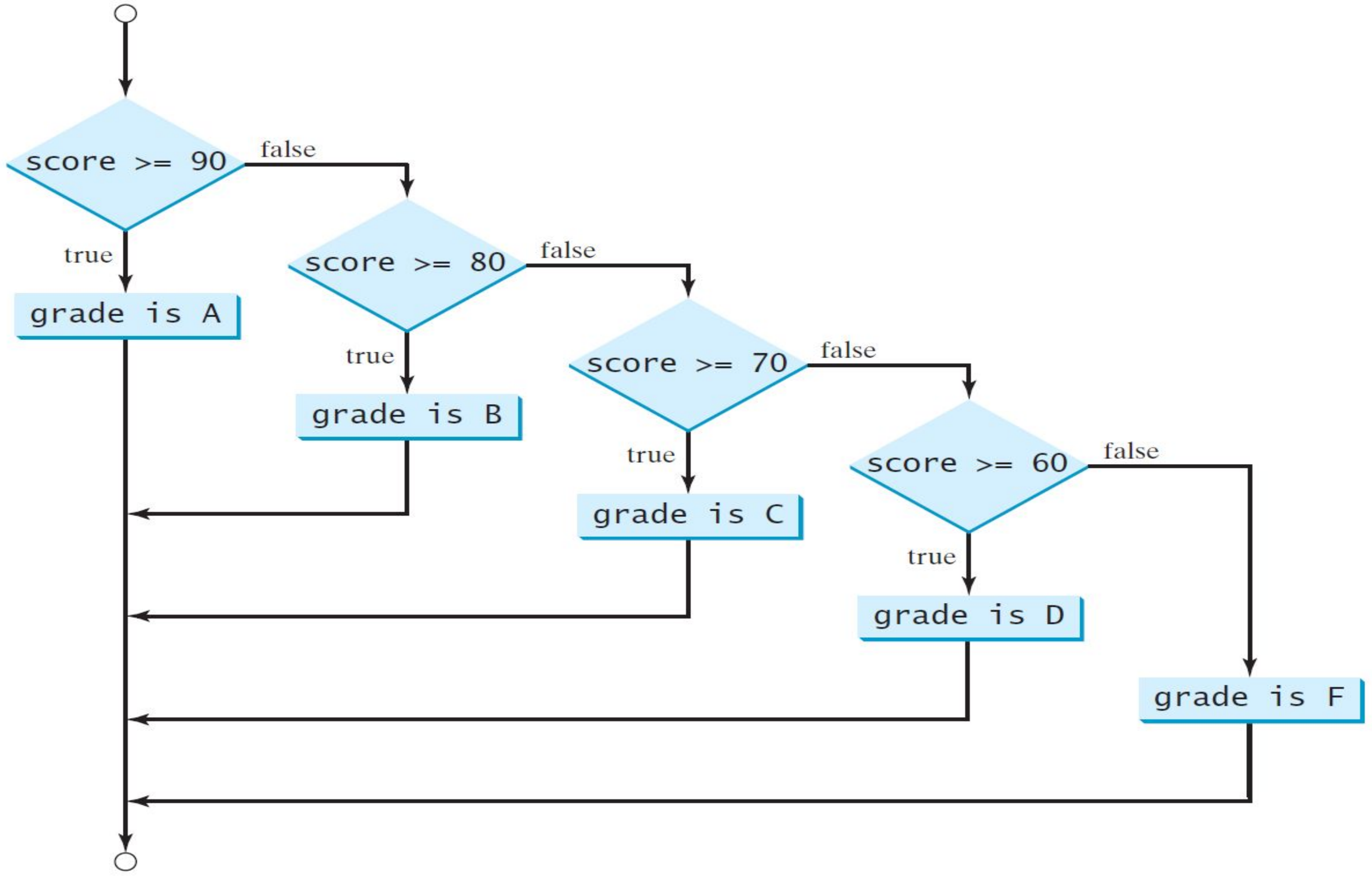
Topic 1c: Multi-Way *if-else* Statements

This program contains an if statement that must choose from among three alternatives. The general form of a ***multi-way if-else*** statement is:

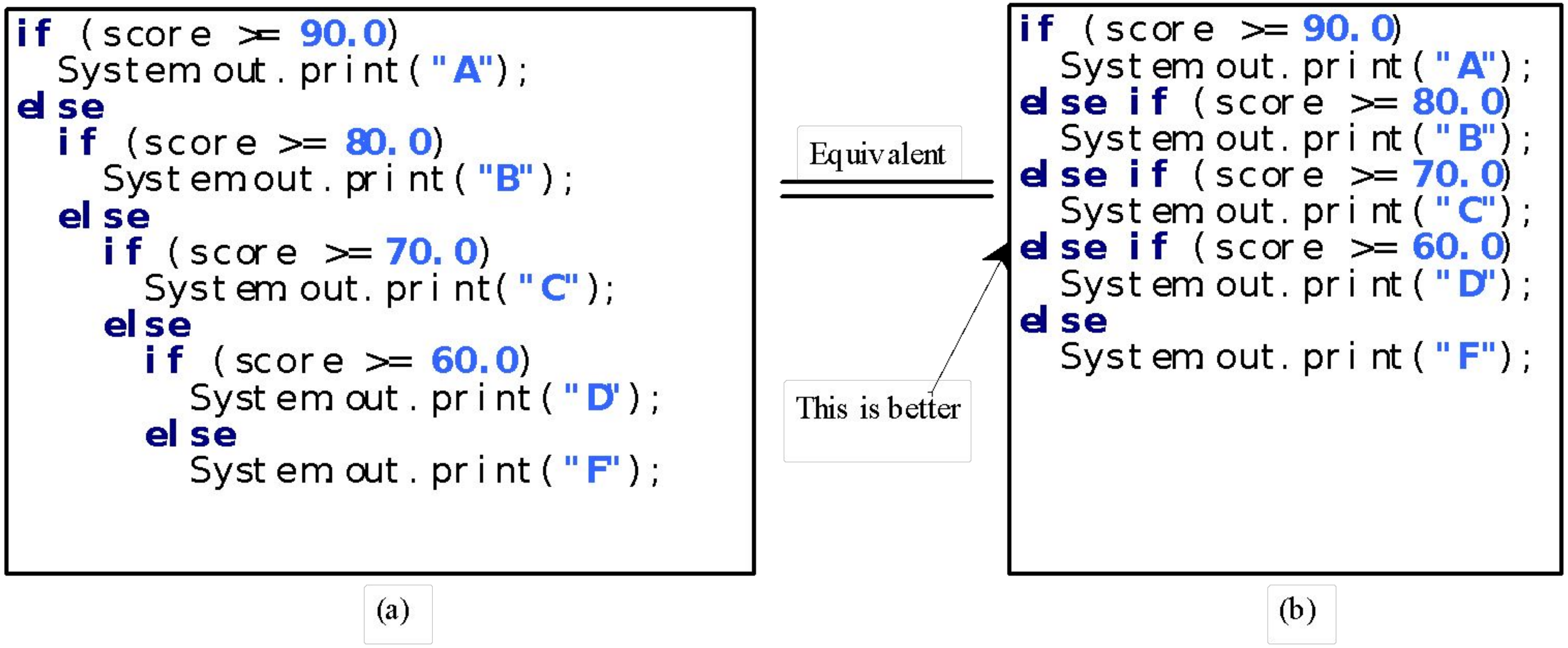
```
if ( condition-expression-1 ) {
      statement(s);
  }
  else if ( condition-expression-2 ) {
       statement(s);
  }
  else if ( condition-expression-3 ) {
       statement(s);
  }
  else {
       statement(s);
  }
```

You can use as many else if lines as you need, and the final **else** is optional.

Multi-Way ***if-else*** Statements illustration



Nested **VS** Chained - ***if/else*** Statements



```
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");
```

```
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");
```

Both statements are equivalent. The chained statements (b) are generally easier to read.

Trace ***if-else*** Statement – First Conditional

Suppose score is 71.0.

The condition is false.

```
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");
```

Trace *if-else* Statement – Second Conditional

Suppose score is 71.0.

The condition is false.

```
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");
```

Trace *if-else* Statement – Third Conditional

Suppose score is 71.0.

The condition is true.

```
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");
```

Trace *if-else* Statement – Expression is True

Suppose score is 71.0.

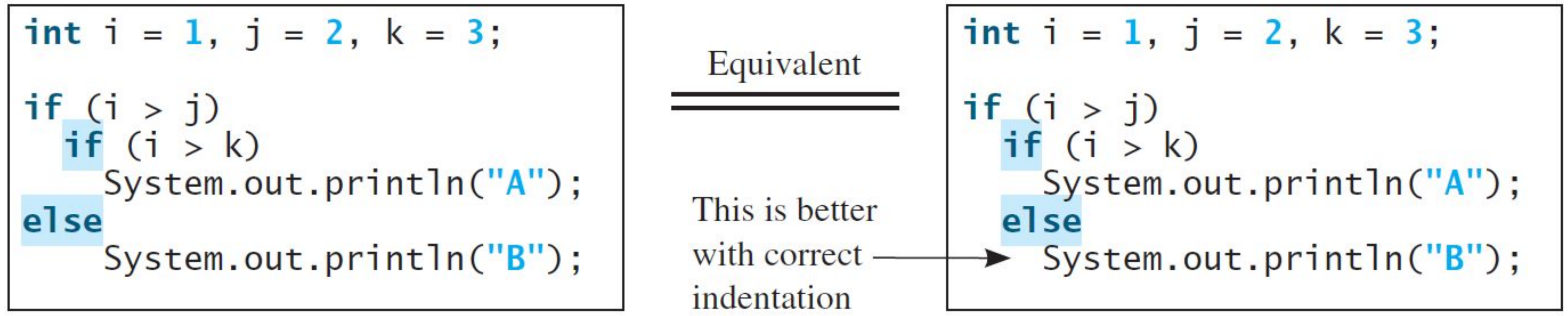
C will print on console.

Exit the ***if*** statement.

```
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");
```

Use Correct Indentation

The ***else*** clause matches the most recent ***if*** clause in the same block.



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

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

(a)

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

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

(b)

What is the output of this code?

Use Curly Braces When Needed

- Nothing is printed from the preceding statement.
- To force the ***else*** clause to match the first ***if*** clause, you must add a pair of braces:

Notice how we can declare (and initialize) multiple variables of the same type in just one line.

```
int i = 1, j = 2, k = 3;
if (i > j) {
   if (i > k)
   System.out.println("A");
} else System.out.println("B");
```

- This statement prints: B

Errant Semicolon

- Adding a semicolon (;) at the end of an ***if*** clause is a common mistake.

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

- The block of code within the curly braces will execute, even when a negative radius is entered.
- This mistake is hard to find because it is neither a compiler error nor a runtime error.
- It is a logic error.

Tip #1

No need to use ***if*** to assign a boolean:

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

(a)

Equivalent

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

(b)

Tip #2

Unnecessary comparison to Boolean Literal:

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

(a)

Equivalent

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

(b)

Knowledge Check

What does the following code print when x has been set to -5?

```
if (x < 0)
{
   System.out.println("x is negative");
}
else if (x == 0)
{
   System.out.println("x is zero");
}
else
{
   System.out.println("x is positive");
}
```

A. x is negative

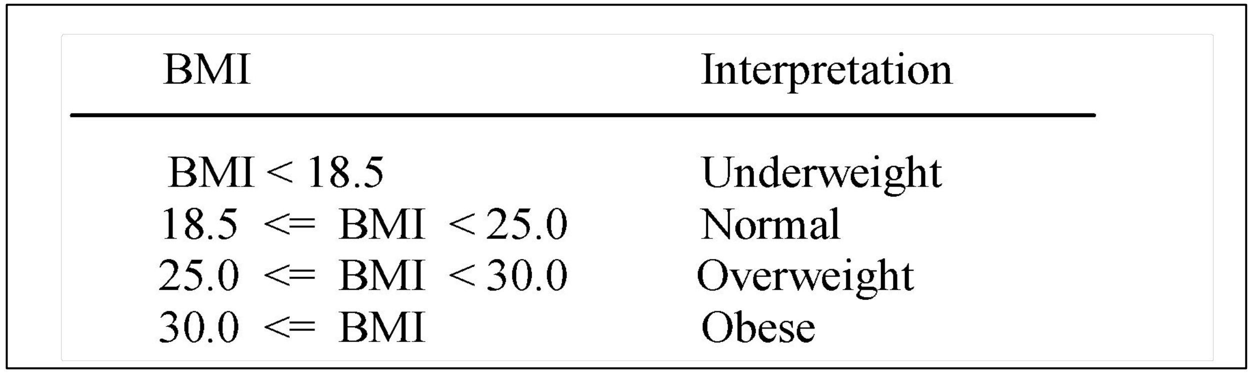
B. x is zero

C. x is positive

Problem: Calculate Body Mass Index

- Body Mass Index (BMI) is a measure of health on weight.
- It is calculated by taking an individual's weight in kilograms and dividing by the square of their height in meters.
  - *BMI = (Weight in Kg) / (Height in Meters * Height in Meters)*

    For example,
  - *weight = 75 kg, height = 1.5 m*
  - *BMI = 75 / (1.5*1.5) = 33.33*
- The interpretation of BMI for people 16 years or older is as follows:



BMI	Interpretation
BMI < 18.5	Underweight
18.5 <= BMI < 25.0	Normal
25.0 <= BMI < 30.0	Overweight
30.0 <= BMI	Obese

Solution : Calculate Body Mass Index

```
public class BMI {
   public static void main(String args[]) {
       Scanner sc = new Scanner(System.in);
       System.out.print("Input weight in kilogram: ");
       double weight = sc.nextDouble();
       System.out.print("\nInput height in meters: ");
       double height = sc.nextDouble();
       // calculate bmi
       double BMI = weight / (height * height);
       // check range
       if(BMI < 18.5)
           System.out.println("Under weight");
       else if(BMI >= 18.5 && BMI < 25)
           System.out.println("Normal");
       else if(BMI >=  25 && BMI < 30)
           System.out.println("Overweight");
       else
           System.out.println("Obese");
       System.out.print("\nThe Body Mass Index (BMI) is " + BMI + " kg/m2");
   }
}
```

Problem: Determining Leap Year?

- This program first prompts the user to enter a year as an int and checks if it is a leap year.
- A year is a leap year if it is divisible by 4 but not by 100, or it is divisible by 400.

Solution: Determining Leap Year?

```
public class LeapYear {
    public static void main(String[] args){
        int year;
        System.out.println("Enter an Year :: ");
        Scanner sc = new Scanner(System.in);
        year = sc.nextInt();
        if (((year % 4 == 0) && (year % 100!= 0)) || (year%400 == 0))
            System.out.println("Specified year is a leap year");
        else
            System.out.println("Specified year is not a leap year");
    }
```

Topic 2: Overview - **Switch** Case Statement

- A ***Switch Case*** statement is used when we have a number of options and we may need to perform a different task for each choice.
- The ***switch case*** statement is a **branch** statement. The case is a keyword that is used with the *switch* statement. It performs the execution of statement/statements when the value of the expression is matched with the case value, and the code of the particular statements is ended by **break** keyword

The syntax of the **switch** statement in Java is:

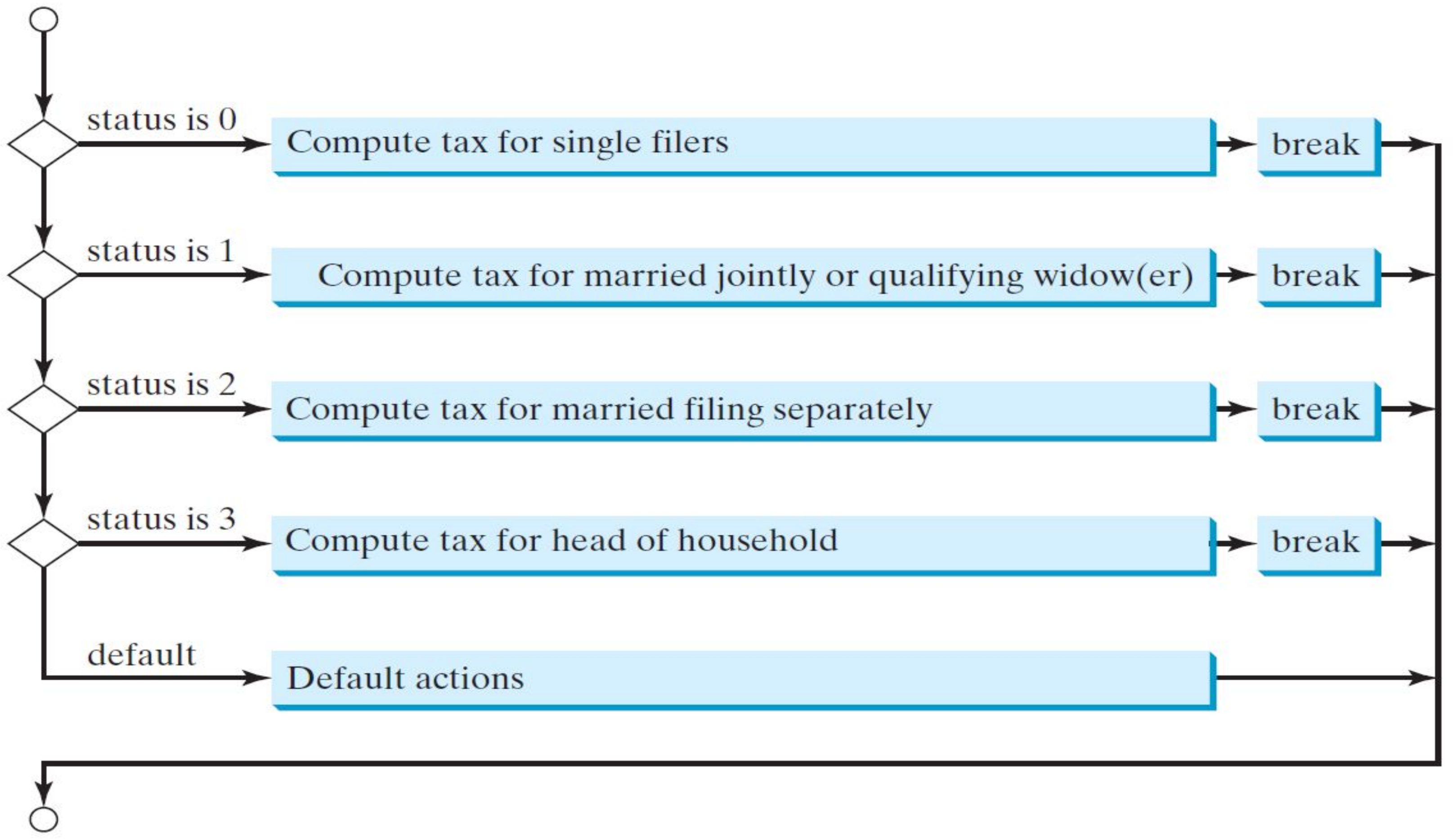
```
switch (expression / variable ) {
  case value1:
    // code
    break;

  case value2:
    // code
    break;

  ...
  ...

  default:
    // default statements
    }
```

switch Statement Flowchart


status is 0
Compute tax for single filers
break
status is 1
Compute tax for married jointly or qualifying widow(er)
break
status is 2
Compute tax for married filing separately
break
status is 3
Compute tax for head of household
break
default
Default actions

Trace *switch* Statement: Evaluate Switch Expression

Suppose day is 2:

```
switch (day) {
  case 1:
  case 2:
  case 3:
  case 4:
  case 5: System.out.println("Weekday"); break;
  case 0:
  case 6: System.out.println("Weekend");
}
```

Trace ***switch*** Statement: Match Case Label

Match case 2.

```
switch (day) {
  case 1:
  case 2:
  case 3:
  case 4:
  case 5: System.out.println("Weekday"); break;
  case 0:
  case 6: System.out.println("Weekend");
}
```

Trace ***switch*** Statement: Fall-Through if no *Break*

Fall-through to case 3.

```
switch (day) {
   case 1:
   case 2:
   case 3:
   case 4:
   case 5: System.out.println("Weekday"); break;
   case 0:
   case 6: System.out.println("Weekend");
}
```

TEKsystems Own change

Trace *switch* Statement: Fall-Through Again

Fall through to case 4.

```
switch (day) {
  case 1:
  case 2:
  case 3:
  case 4:
  case 5: System.out.println("Weekday"); break;
  case 0:
  case 6: System.out.println("Weekend");
}
```

Trace ***switch*** Statement: Execute

Fall-through and execute case 5.

```
swi       (day) {
  ca  e 1:
  c  se 2:
  c  se 3:
  case 4:
  case 5: System.out.println("Weekday"); break;
  case 0:
  case 6: System.out.println("Weekend");
}
```

Trace *switch* Statement: Break

Encounter break.

```
switch (day) {
  case 1:
  case 2:
  case 3:
  case 4:
  case 5: System.out.println("Weekday"); break;
  case 0:
  case 6: System.out.println("Weekend");
}
```

Trace *switch* statement: End *switch* Statement

Exit the *switch* statement.
Execute the first line after the *switch* statement.

```
switch (day)
  case 1:
  case 2:
  case 3:
  case
  case 5: System.out.println("Weekday"); break;
  case 0:
  case 6: System.out.println("Weekend");
}
```

switch Statement Example 1

When control reaches the *switch* statement, the program evaluates (grade / 10) (the *switch expression*).

Then it attempts to match the *switch* expression value to one of the case labels. If a match is found, the statement(s) following that case label are executed.

If no match is found, then the default statement(s) (if there is a default case) will execute.

Every case ends with a break statement. This transfers control to the first statement after the switch structure, and break is *required*.*

```
public static void main(String args[])
{
   String output;
   // grade is a value 0 - 100
   int grade = 50;
   switch (grade / 10)
   {
       case 10: output = "Wow!; ";   break;
       case 9: output = "Excellent!"; break;
       case 8: output = "Very Good!"; break;
       case 7: output = "Good!"; break;
       default: output = "Keep trying!";
break;
   }
   System.out.println(output);
}
```

About the Case Labels

- The **switch expression** is evaluated at runtime and may be any variable or expression.
- The **case labels** are evaluated at compile-time and must be constants or literals.
- This example uses String literals. Previous examples used **int** literals.
- You can also use **char** literals and Enum types.
- The **break** keyword terminates the ***switch*** statement.
- If you omit the break statement, execution will fall through to the next case.

This example contains an unintended fall-through. **Can you find it?**

```java
private static int processArgument(String arg) {
  int result;
  switch(arg) {
    case "a": result = 1; break;
    case "b": // fall through
    case "c": result = 2; break;
    case "d":
    case "e":
    case "f": result = 3;
    case "g": result = 4; break;
    default: result = -1;
  }
  return result;
}
```

default case in Java switch-case

The switch statement also includes an optional default case. It is executed when the expression doesn't match any of the cases. For example,

```
public class switchcaseDemo{
  public static void main(String[] args) {
      int expression = 9;

    switch(expression) {
      case 2:
        System.out.println("Small Size");
        break;

      case 3:
        System.out.println("Large Size");
        break;

      // default case
      default:
        System.out.println("Unknown Size");
    }
  }
 }
```

In this example, we have created a **switch-case** statement.
Here, the value of `expression` doesn't match with any of the cases.
Hence, the code inside the **default** case is executed.

```
default:
    System.out.println("Unknown Size);
```

Output

```
Unknown Size
```

switch Statement: char Example

```java
public static void main(String[] args)
{
   //Declaring a variable for switch expression
   char alphabet = 'b';
   switch(alphabet) //Switch expression
   {
       //Case statements
       case 'a':
           System.out.println("This is character 'a' ");
           break;
       case 'b':
           System.out.println("This is character 'b' ");
           break;
       case 'c':
           System.out.println("This is character 'c' ");
           break;
       //Default case statement
       default: System.out.println("Please enter valid input");
   }
}
```

Here is an example of switching on a char. Notice how each case label is now a char literal.

switch Statement: enum Example

In the below example, we will explore how Enum keyword works along with Switch case statements when Enum is declared outside main class.

```
enum Cars {
       BMW,
       JEEP,
       AUDI,
       VOLKSWAGON,
       NANO,
       FIAT;
   }
   public static void main(String args[]) {
       // Declaring Enum variable
       Cars c;
       c = Cars.AUDI;
       // Switch keyword
       switch (c) {
           // Case statements
           case BMW:
               System.out.println("You choose BMW !");
               break;
           case JEEP:
               System.out.println("You choose JEEP !");
               break;
```

```
   case AUDI:
     System.out.println("You choose AUDI !");
       break;
   case VOLKSWAGON:
     System.out.println("You choose VOLKSWAGON !");
     break;
   case NANO:
     System.out.println("You choose NANO !");
     break;
   case FIAT:
     System.out.println("You choose FIAT !");
    default:
     System.out.println("NEW BRAND'S CAR.");
       break;
       }
   }
}
```

switch Statement Summary

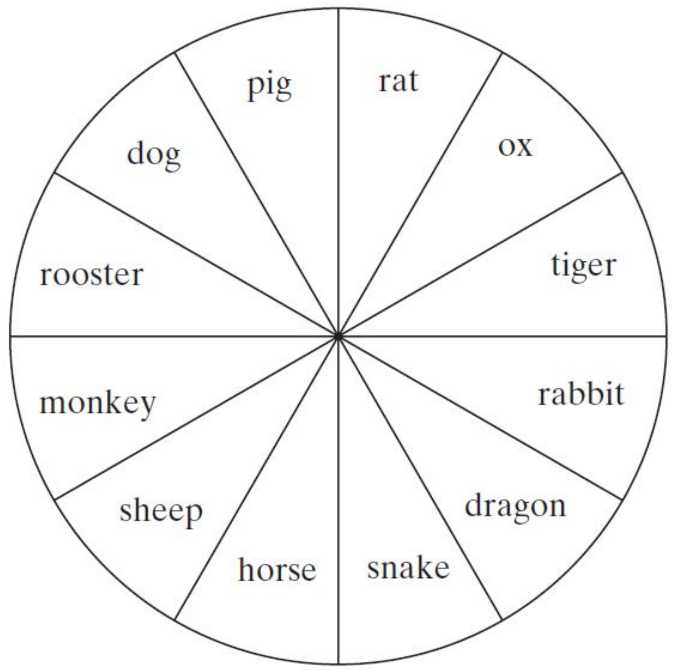
Essential parts of the **switch** statement:

- **switch expression** - this is evaluated at run-time and must evaluate to an integer type, a string, a char, or an enum.
- **case labels** - these must be literals or constants, and their type must match the type of the switch expression.
- **break statements** - break (or return) statements must be used to prevent accidental fall-through.
- **default case** - the default case is executed if no other case matches the value produced by the switch expression.

```
switch (switch-expression) {
   case value1:  statement(s)1;
                 break;
   case value2: statement(s)2;
                 break;
   …
   case valueN: statement(s)N;
                 break;
   default: statement(s)-for-default;
}
```

Problem: Chinese Zodiac

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



`year % 12` =

- 0: monkey
- 1: rooster
- 2: dog
- 3: pig
- 4: rat
- 5: ox
- 6: tiger
- 7: rabbit
- 8: dragon
- 9: snake
- 10: horse
- 11: sheep

Note that **year % 12** determines the Zodiac sign. 1900 is the year of the rat because **1900 % 12** is **4**. Listing 3.10 gives a program that prompts the user to enter a year and displays the animal for the year

Solution: Chinese Zodiac

```
public static void main(String[] args) {
    int year = 2022;
   switch (year % 12)
    {    case 0: System.out.println("monkey"); break;
         case 1: System.out.println("rooster"); break;
         case 2: System.out.println("dog"); break;
         case 3: System.out.println("pig"); break;
         case 4: System.out.println("rat"); break;
         case 5: System.out.println("ox"); break;
         case 6: System.out.println("tiger"); break;
         case 7: System.out.println("rabbit"); break;
         case 8: System.out.println("dragon"); break;
         case 9: System.out.println("snake"); break;
         case 10: System.out.println("horse"); break;
         case 11: System.out.println("sheep");
    }
}
```

Practice Assignment

Complete this assignment 303.2.2 - Practice Assignment - Control Flow (Conditional) Statements. You can find this assignment on Canvas, under the Assignment section.

Use your office hours to complete this assignment. If you have technical questions while performing the practice assignment, ask your instructors for assistance.

Note: It is not mandatory assignments. This assignment does not count toward the final grade

Summary

We use conditional statements to control the flow of programs. In Java, *if/else* statements are used to control program flow based on a certain set of conditions. Additionally, Java offers a feature called the *switch* statement, which will evaluate an expression against multiple cases. Normally, *switch* will be used in a scenario where there is a need to perform the action on certain conditions, and conditions are many. In the case of only 2-3 conditions, things can be worked out with *if-else -if* statements.

References

https://math.hws.edu/javanotes/c3/s5.html

https://www.javatpoint.com/control-flow-in-java

Questions?

End of Module