

COMPUTER SCIENCE DEPARTMENT FACULTY OF ENGINEERING AND TECHNOLOGY ADVANCED PROGRAMMING COMP231

Instructor :Farid Mohammad



Top 10 Reasons to Learn Java

1. Java's Popularity and High Salary
2. Java is Easy to Learn
3. Java has a Large Community
4. Java has an abundant API
5. Java has multiple Open Source Libraries
6. Java has Powerful Development Tools
7. Java is Free of Cost
8. Java is Platform Independent
9. Java has great Documentation Support
10. Java is Versatile

Reference:

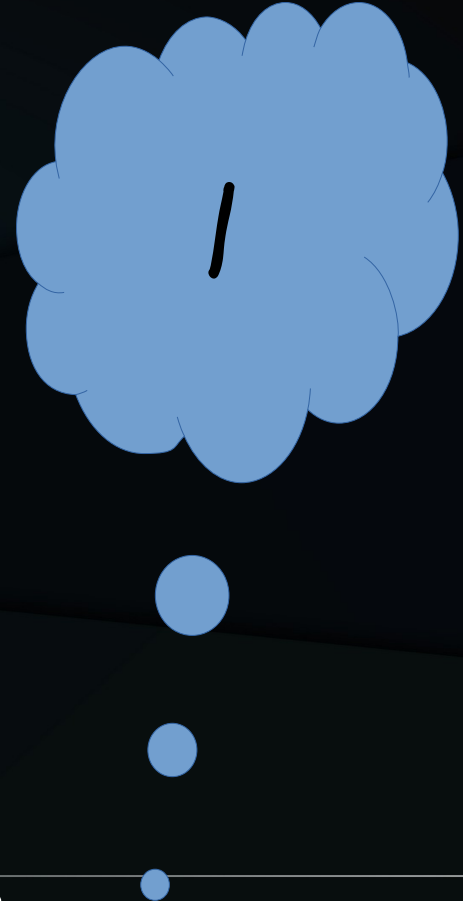
<https://www.geeksforgeeks.org/top-10-reasons-to-learn-java/>



What you'll do in Java

```
import java.awt.*;
import java.awt.event.*;
class Party {
    public void buildInvite() {
        Frame f = new Frame();
        Label l = new Label("Party at
        Tim's");
        Button b = new Button("You bet");
        Button c = new Button("Shoot
        me");
        Panel p = new Panel();
        p.add(l);
        } // more code here...
    }
```

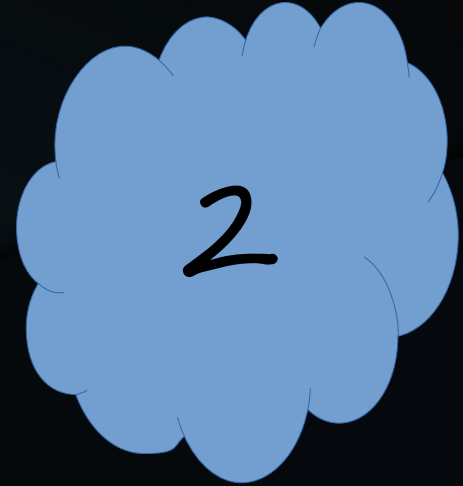
SOURCE



What you'll do in Java

```
%javac Party.java
```

Compiler



What you'll do in Java

Compiled code:
Party.class

Output
(code)



What you'll do in Java

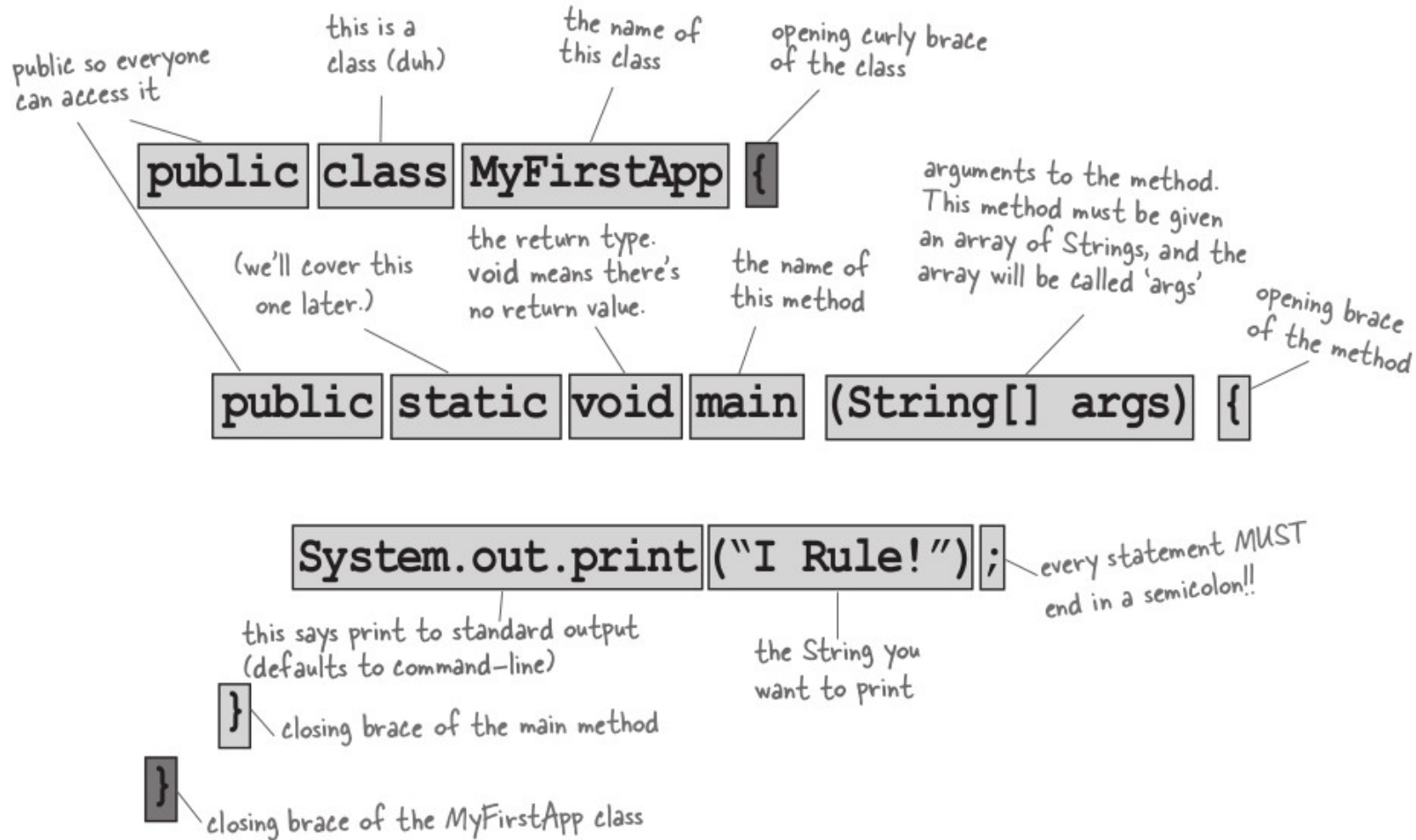
java Party

Run the program by starting the Java Virtual Machine (JVM) with the Party.class file.

Virtual
Machines

4

Anatomy of a class



When the JVM starts running, it looks for the class you give it at the command line

starts looking for main

Next, the JVM runs everything between the curly braces { } of your main

Writing a Simple Program

```
public class ComputeArea {  
  
    public static void main(String[] args) {  
  
        double radius; // Declare radius  
        double area; // Declare area  
  
        // Assign a radius  
        radius = 20; // radius is now 20  
  
        // Compute area  
        area = radius * radius * 3.14159;  
        // Display results  
        System.out.println("The area for the circle of radius " +  
            radius + " is " + area);  
    }  
}
```

- 1- Save as
ComputeArea.java
- 2- Compile by:
Javac ComputeArea.java
- 3- Run by:
java ComputeArea

Reading Input from the Console

use the Scanner class to create an object to read input from System.in

```
Scanner input = new Scanner(System.in);
```

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

```
import java.util.Scanner;  
public class ComputeAreaWithConsoleInput {  
    public static void main(String[] args) {  
        // Create a Scanner object  
        Scanner input = new Scanner(System.in); create a  
        Scanner  
        // Prompt the user to enter a radius  
        System.out.print("Enter a number for radius: ");  
        double radius = input.nextDouble();  
        // Compute area
```

Named Constants

- The value of a variable may change during the execution of a program, but a named constant, or simply constant, represents permanent data that never changes
- Syntax:
final datatype CONSTANTNAME = value;
- Ex: `final double PI = 3.14159; // Declare a constant`

Numeric Types

TABLE 2.1 Numeric Data Types

<i>Name</i>	<i>Range</i>	<i>Storage Size</i>	
byte	-2^7 to $2^7 - 1$ (-128 to 127)	8-bit signed	byte type
short	-2^{15} to $2^{15} - 1$ (-32768 to 32767)	16-bit signed	short type
int	-2^{31} to $2^{31} - 1$ (-2147483648 to 2147483647)	32-bit signed	int type
long	-2^{63} to $2^{63} - 1$ (i.e., -9223372036854775808 to 9223372036854775807)	64-bit signed	long type
float	Negative range: $-3.4028235E + 38$ to $-1.4E - 45$ Positive range: $1.4E - 45$ to $3.4028235E + 38$	32-bit IEEE 754	float type
double	Negative range: $-1.7976931348623157E + 308$ to $-4.9E - 324$ Positive range: $4.9E - 324$ to $1.7976931348623157E + 308$	64-bit IEEE 754	double type

Operator Precedence

Here is an example of how an expression is evaluated:

$$3 + 4 * 4 + 5 * (4 + 3) - 1$$

(1) inside parentheses first

$$3 + 4 * 4 + 5 * 7 - 1$$

(2) multiplication

$$3 + 16 + 5 * 7 - 1$$

(3) multiplication

$$3 + 16 + 35 - 1$$

(4) addition

$$19 + 35 - 1$$

(5) addition

$$54 - 1$$

(6) subtraction

$$53$$

Augmented Assignment Operators

TABLE 2.4 Augmented Assignment Operators

<i>Operator</i>	<i>Name</i>	<i>Example</i>	<i>Equivalent</i>
<code>+=</code>	Addition assignment	<code>i += 8</code>	<code>i = i + 8</code>
<code>-=</code>	Subtraction assignment	<code>i -= 8</code>	<code>i = i - 8</code>
<code>*=</code>	Multiplication assignment	<code>i *= 8</code>	<code>i = i * 8</code>
<code>/=</code>	Division assignment	<code>i /= 8</code>	<code>i = i / 8</code>
<code>%=</code>	Remainder assignment	<code>i %= 8</code>	<code>i = i % 8</code>

Numeric Type Conversions

```
int i = 1;  
byte b = i; // Error because explicit casting is required
```

Fix:

```
byte b = (byte)i;
```

Division example:

```
int num = 5;  
int denom = 7;  
double d = num / denom;
```

the value of d is 0.0

FIX

```
double d = ((double) num) / denom;
```

Strings

- Java strings are sequences of **Unicode** characters

- `String e = ""; // an empty string`

```
String greeting = "Hello";
```

- **Substrings**

```
String greeting = "Hello";
```

```
String s = greeting.substring(0, 3);
```

- **Concatenation**

```
String expletive = "Expletive";
```

```
String PG13 = "deleted";
```

```
String message = expletive + PG13;
```

Logical Operators

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

```
int age=24;
```

!(age > 18) is false

(age > 28) && (weight <= 140) is true

(age > 34) || (weight >= 150) is false

(age > 34) ^ (weight > 140) is false , because (age > 34) and (weight > 140) are both false .

Truth table summarized:

!: true if false

&&: true if both are true

||: true if one is true

^: true if one is false and one is true

TABLE 3.3 Boolean Operators

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

TestBooleanOperators.java

```
import java.util.Scanner;
public class TestBooleanOperators {
    public static void main(String[] args) {
        // Create a Scanner
        Scanner input = new Scanner(System.in);
        // Receive an input
        System.out.print("Enter an integer: ");
        int number = input.nextInt();
        if (number % 2 == 0 && number % 3 == 0)
            System.out.println(number + " is divisible by 2 and 3.");

        if (number % 2 == 0 || number % 3 == 0)
            System.out.println(number + " is divisible by 2 or 3.");

        if (number % 2 == 0 ^ number % 3 == 0)
            System.out.println(number +
                " is divisible by 2 or 3, but not both."); exclusive or
        }
    }
}
```

boolean Data Type

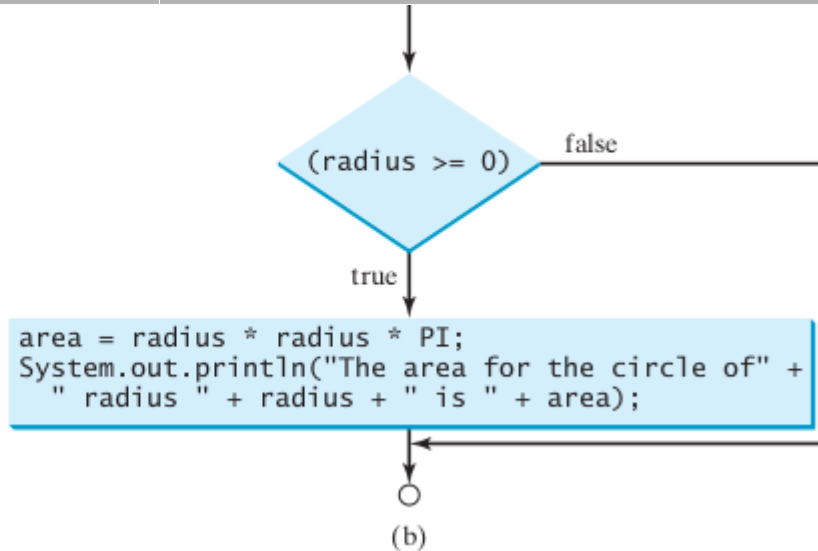
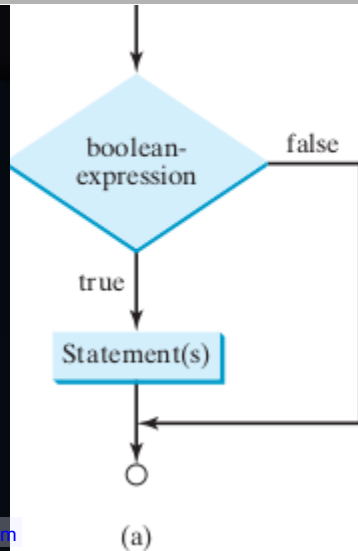
- The boolean data type declares a variable with the value either true or false
- boolean isLarge=(radius < 0) **false**
- boolean isZero=(radius==0) **false**
- Boolean isNotZero=(radius!=0) **true**

If statement

- An if statement is a construct that enables a program to specify alternative paths of execution

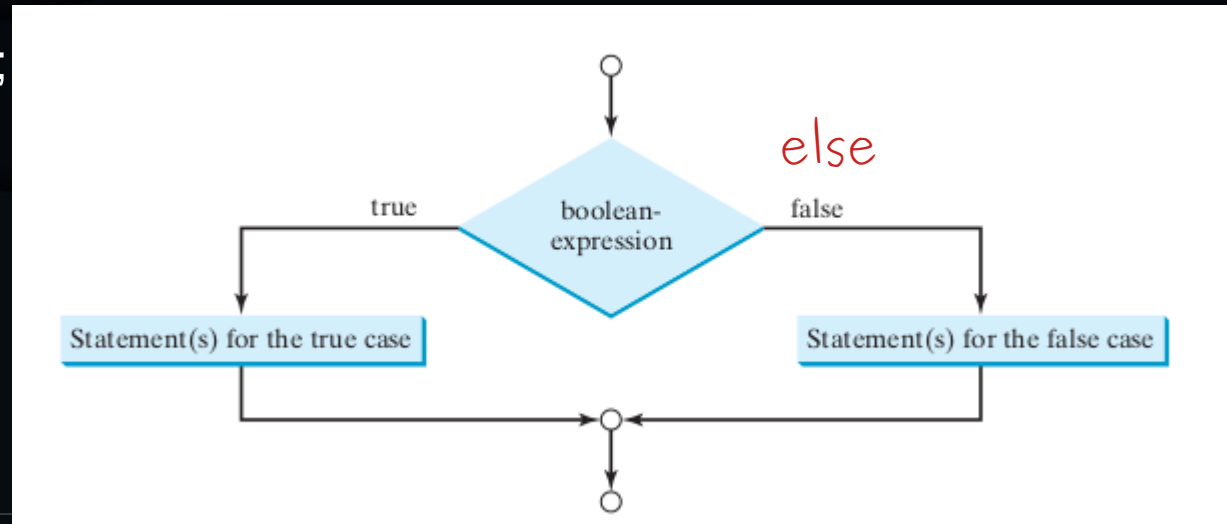
```
if (boolean-expression) {  
    statement(s);  
}
```

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

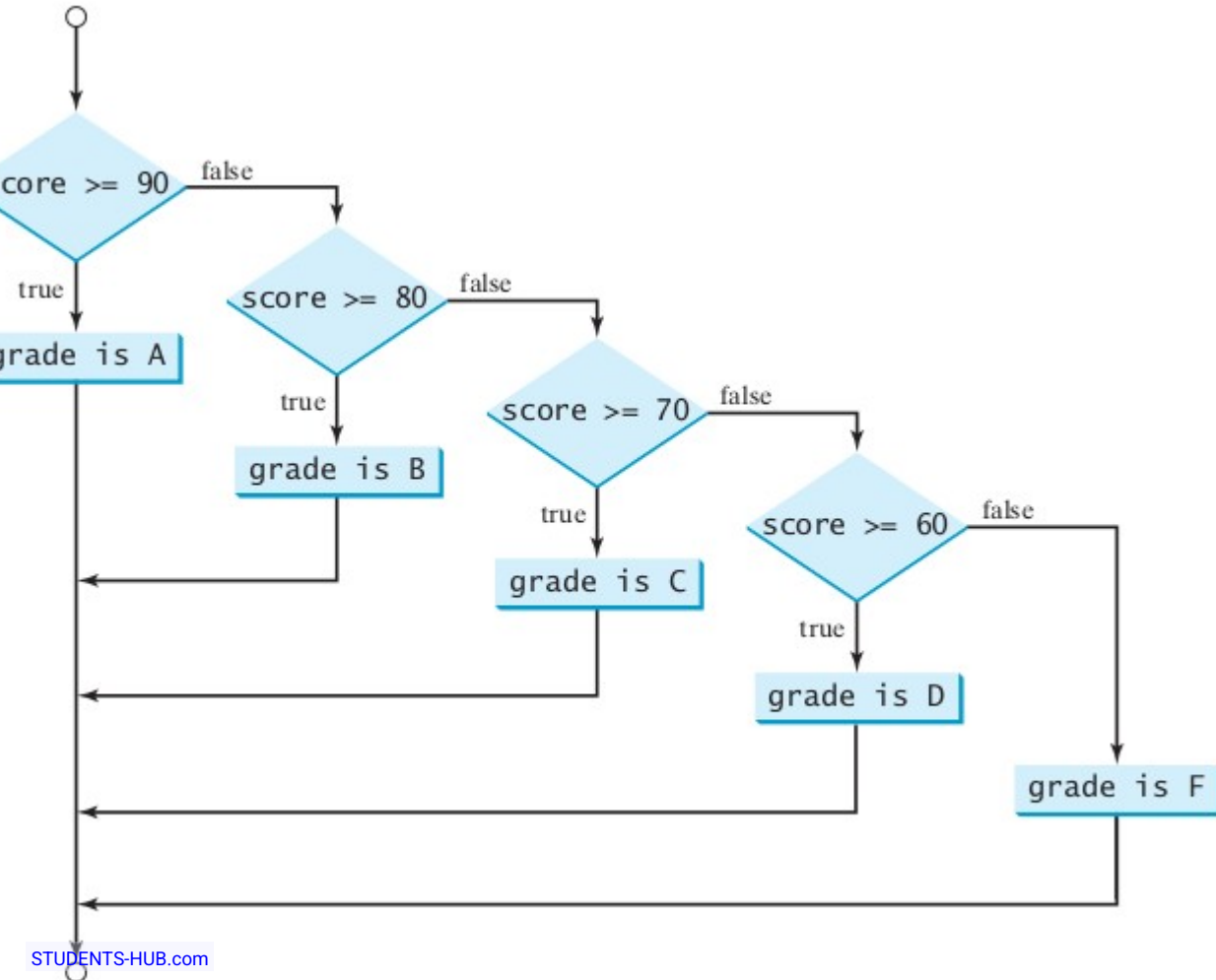


if-else Statements

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



If inside If



```
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");
```

Generating Random Numbers

You can use **Math.random()** to obtain a random double value between 0.0 and 1.0 ,
Excluding 1.0

Generating a random number between zero and 10:

```
int randInt1=Math.random()*10;
```

Generating a random integer number in a range:

```
(int)(Math.random() * ((max - min) + 1)) + min
```

Ex between 6 and 15:

```
(int)(Math.random() * ((9) + 1)) + 6
```

switch Statements

```
int tax=3;  
switch (tax) {  
    case 0: compute tax for single filers;  
        break;  
    case 1: compute tax for married jointly or qualifying widow(er);  
        break;  
    case 2: compute tax for married filing separately;  
        break;  
    case 3: compute tax for head of household;  
        break;  
    default: System.out.println("The default action");  
}
```

3.14 Conditional Expressions

- Short if statement=replace if
- Ex: `x=1; y = (x > 0) ? 10 : -1;`
value of y is 10