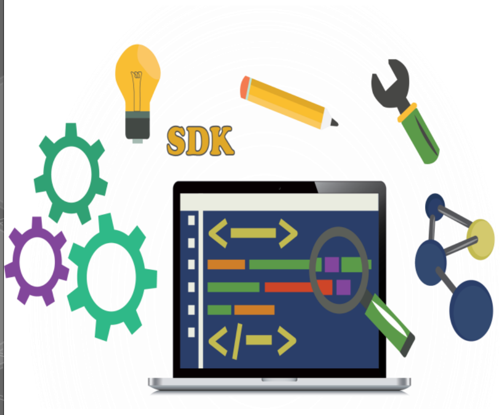
Developing Software: Introduction



YMC
Y-MAX COLLEGE
IT & Business Management Institute

Prepared By:

Dr. Myomyo Thannaing
(Senior Lecture @Y-Max)
M.C.Sc, Ph.D. (IT)

Y-Max College (YMC)

"Our Mission"

"The Advanced Technical Education For Professional Careers, Carried Out In A Real Professional Way."

Please like and share it for us. Warmly welcome to any question concern with IT.

Find us on: facebook ⟷ https://www.facebook.com/ymaxcollege

⟷ www.ymaxcollege.com

Y-Max College (YMC)

OUR CAMPUS

- **Y-Max College (YMC)**
- ခြံအမှတ် ၅၃၁ (ဘီ)၊ မာလာမြိုင်ရိပ်သာလမ်း၊ လှည်းတန်းစင်တာအနီး၊ ကမာရွတ်မြို့နယ်၊ ရန်ကုန်။
- ☎ 09 793354330~31

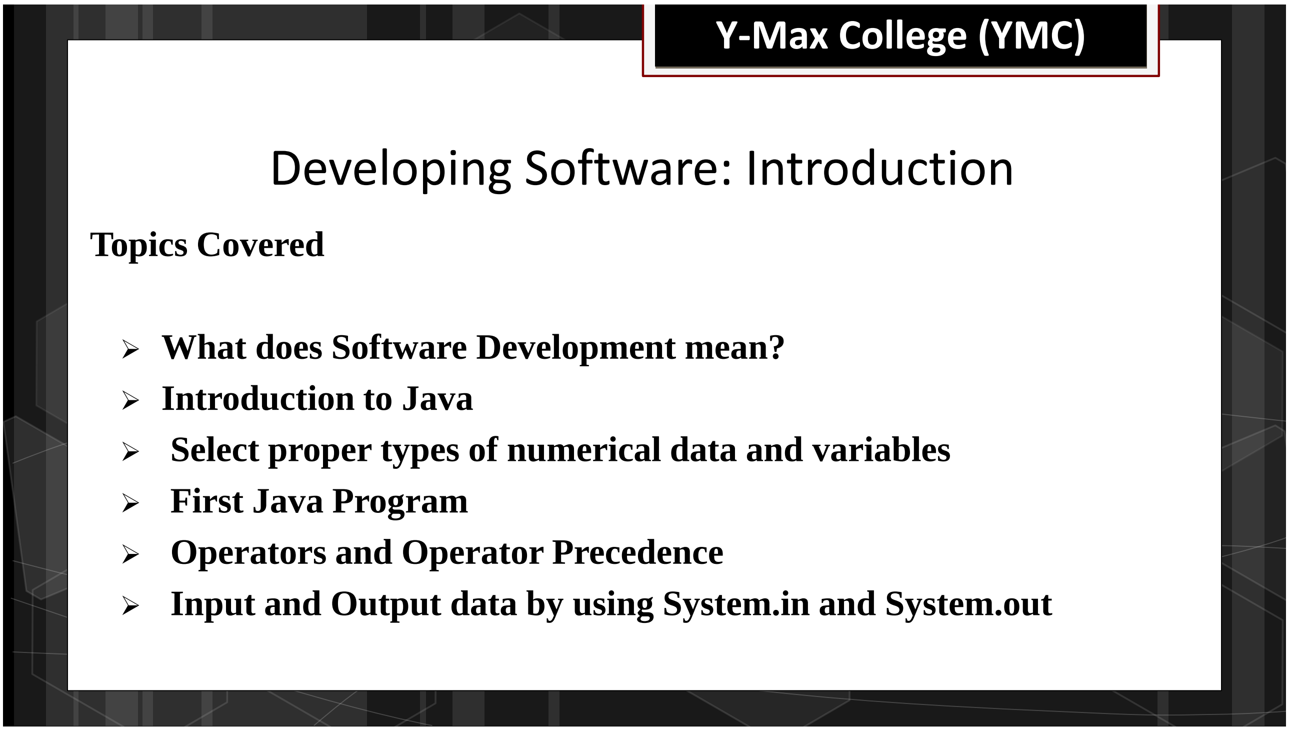
Y-Max College (YMC)

Contact Address

- **Center-1 (ဗိုလ်တစ်ထောင်)**
- အမှတ် (၁၁၇) ၊ ၃လွှာ ၊ ၄၆လမ်း (အထက်ဘလောက်) ၊ ဗိုလ်တစ်ထောင်မြို့နယ် ၊ ရန်ကုန်။
- PH : 09 793354335

- **Center-2 (လှည်းတန်း)**
- တိုက်(၁) ၊ ပထမထပ် ၊ အင်းစိန်လမ်းမကြီး ၊ လှည်းတန်းမီးပွိုင့်အနီး ၊ ကမာရွတ်မြို့နယ် ၊ ရန်ကုန်။
- PH : 09 261319178 , 09 972707373

- **Center-3 (မန္တလေး)**
- အမှတ် (၅၅) ၊ ၇၉လမ်း ၊ ၃၆x၃၇လမ်းကြား ၊ ဟေမာလာမြောက်ရပ် ၊ မဟာအောင်မြေမြို့နယ် မန္တလေးမြို့။
- PH : 09 766883288, 09 766883299

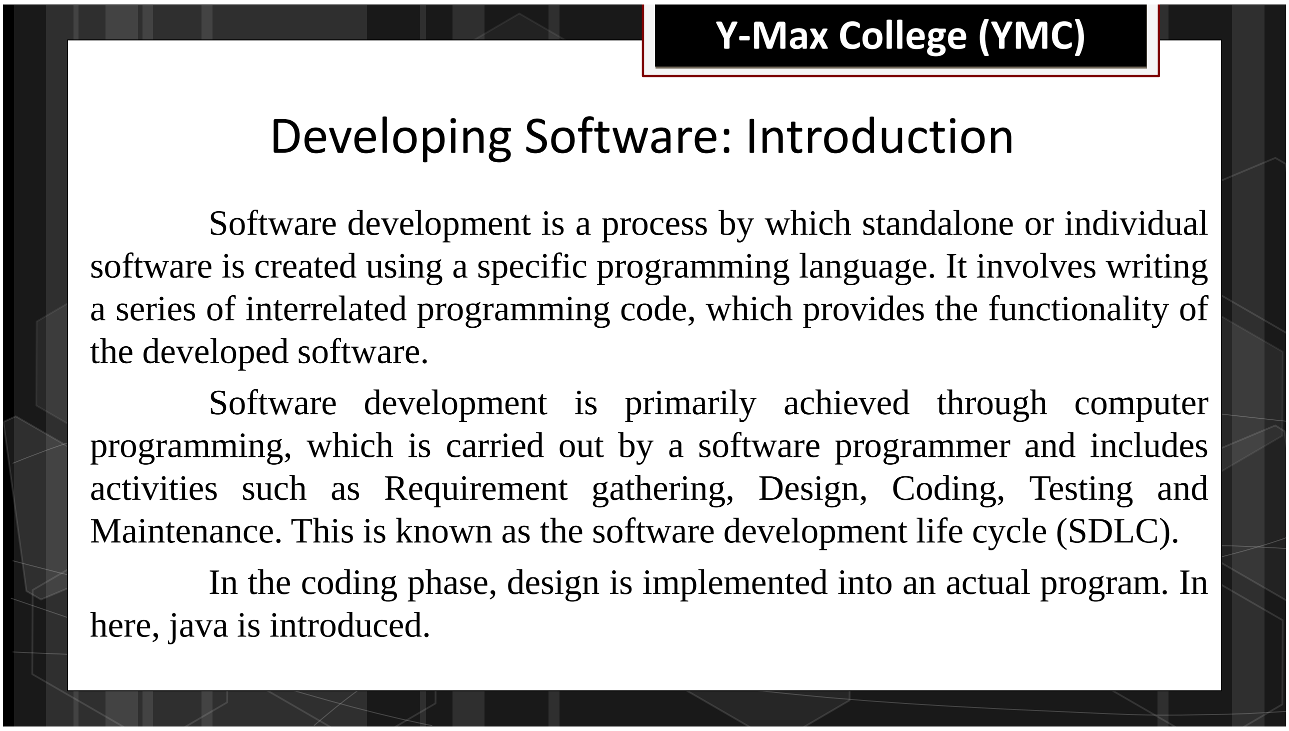


Y-Max College (YMC)

Developing Software: Introduction

Topics Covered

- **What does Software Development mean?**
- **Introduction to Java**
- **Select proper types of numerical data and variables**
- **First Java Program**
- **Operators and Operator Precedence**
- **Input and Output data by using System.in and System.out**



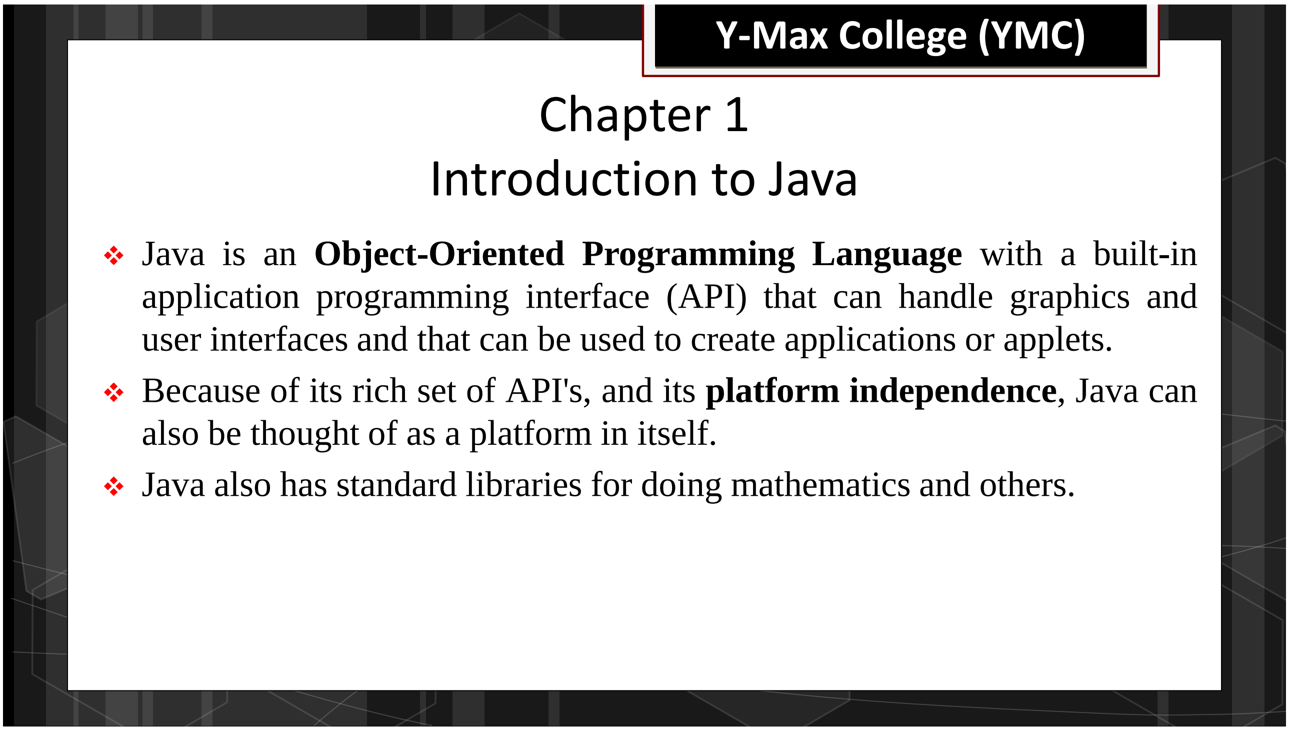
Y-Max College (YMC)

Developing Software: Introduction

Software development is a process by which standalone or individual software is created using a specific programming language. It involves writing a series of interrelated programming code, which provides the functionality of the developed software.

Software development is primarily achieved through computer programming, which is carried out by a software programmer and includes activities such as Requirement gathering, Design, Coding, Testing and Maintenance. This is known as the software development life cycle (SDLC).

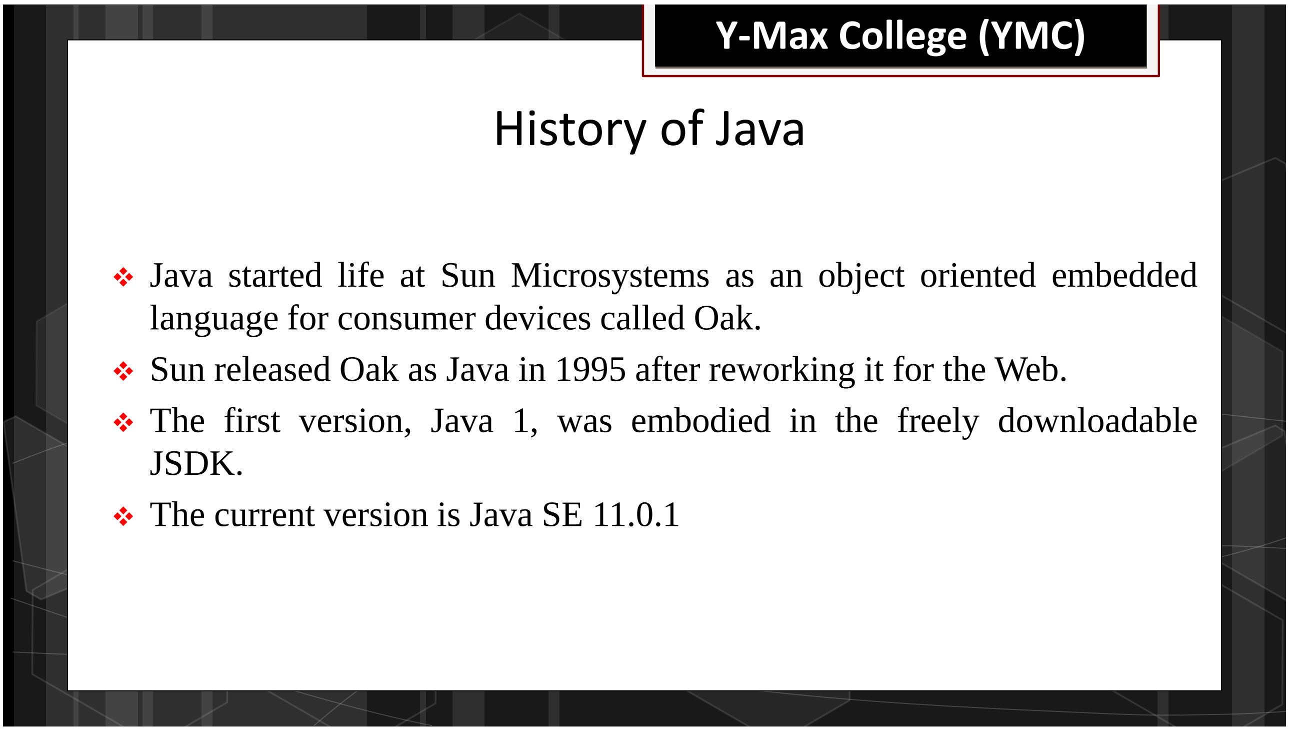
In the coding phase, design is implemented into an actual program. In here, java is introduced.



Y-Max College (YMC)

Chapter 1
Introduction to Java

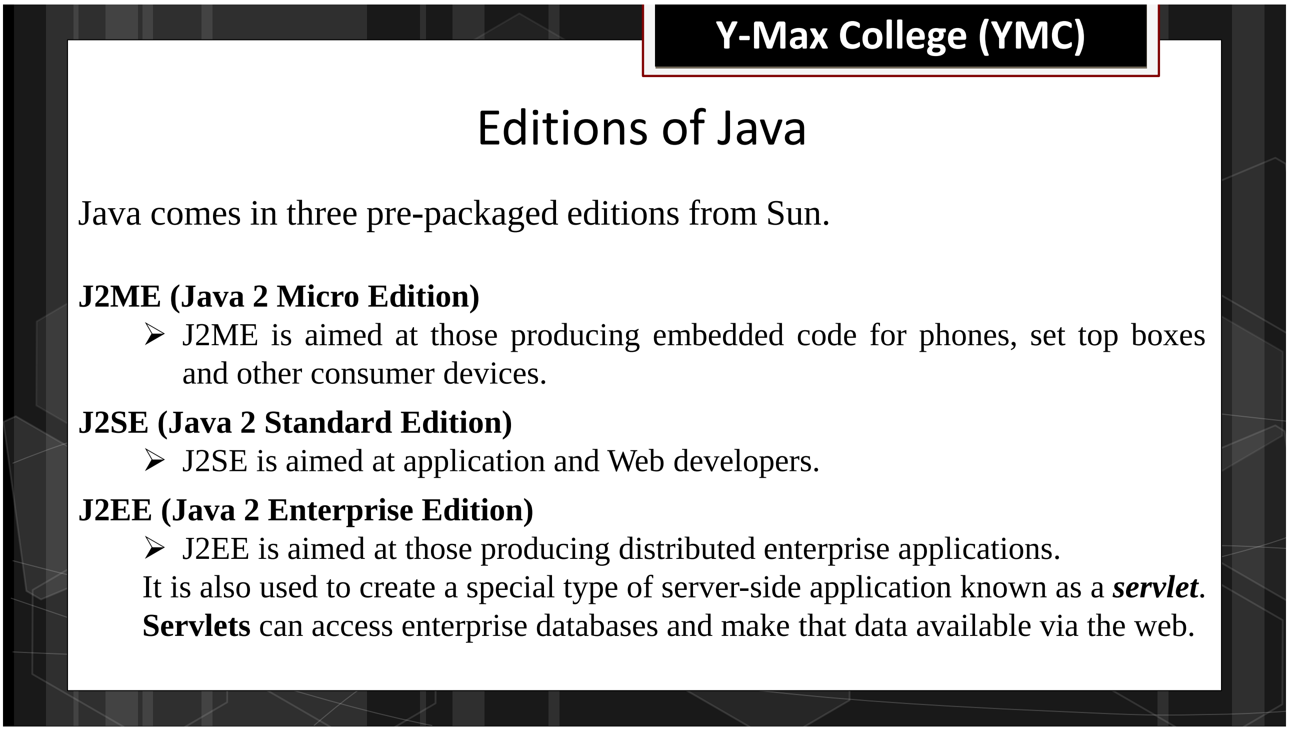
- Java is an **Object-Oriented Programming Language** with a built-in application programming interface (API) that can handle graphics and user interfaces and that can be used to create applications or applets.
- Because of its rich set of API's, and its **platform independence**, Java can also be thought of as a platform in itself.
- Java also has standard libraries for doing mathematics and others.



Y-Max College (YMC)

History of Java

- Java started life at Sun Microsystems as an object oriented embedded language for consumer devices called Oak.
- Sun released Oak as Java in 1995 after reworking it for the Web.
- The first version, Java 1, was embodied in the freely downloadable JSDK.
- The current version is Java SE 11.0.1



Y-Max College (YMC)

Editions of Java

Java comes in three pre-packaged editions from Sun.

J2ME (Java 2 Micro Edition)

- ➢ J2ME is aimed at those producing embedded code for phones, set top boxes and other consumer devices.

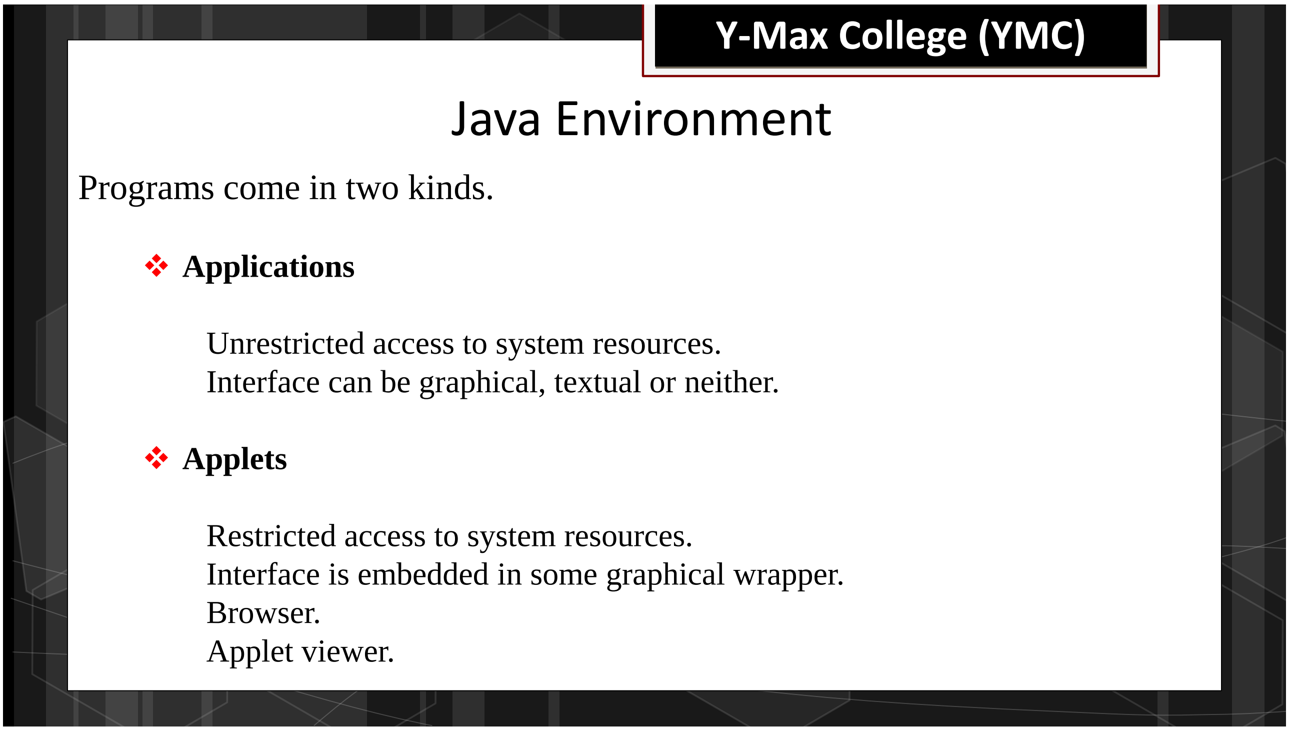
J2SE (Java 2 Standard Edition)

- ➢ J2SE is aimed at application and Web developers.

J2EE (Java 2 Enterprise Edition)

- ➢ J2EE is aimed at those producing distributed enterprise applications.

It is also used to create a special type of server-side application known as a ***servlet***.
Servlets can access enterprise databases and make that data available via the web.



Y-Max College (YMC)

Java Environment

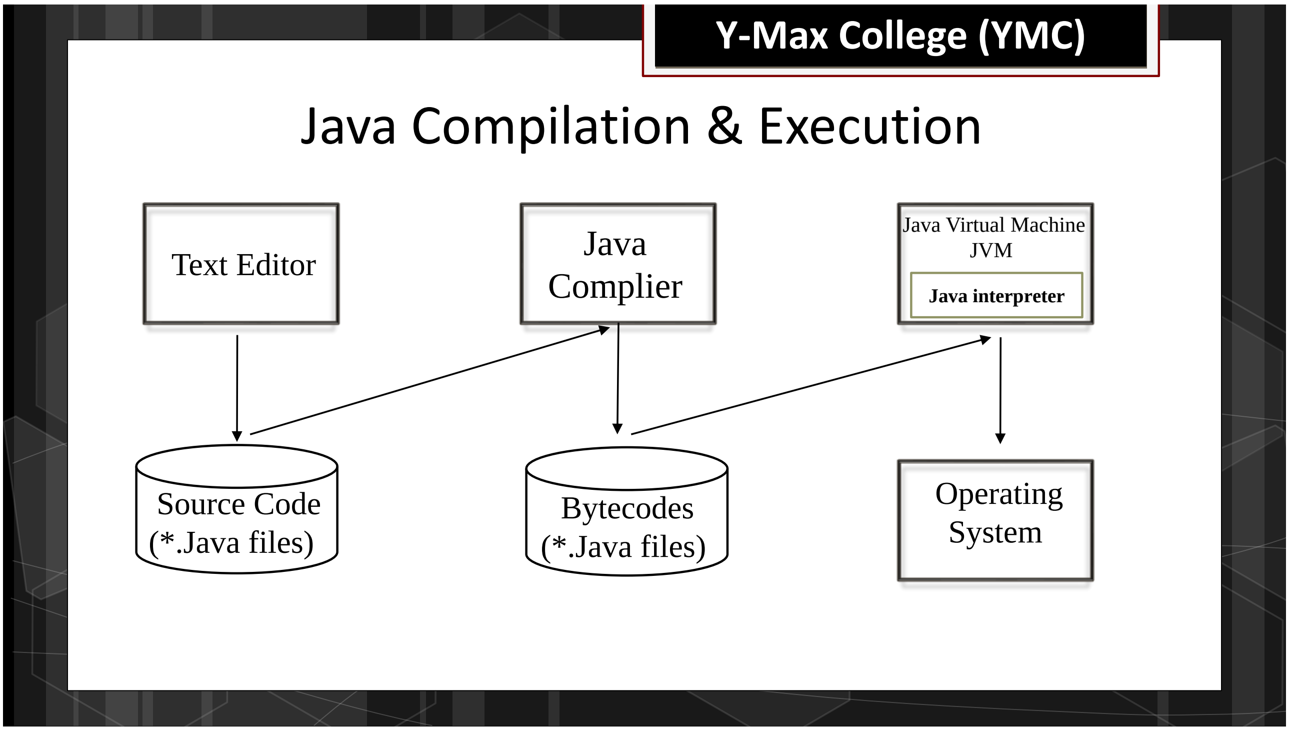
Programs come in two kinds.

- ❖ **Applications**

  Unrestricted access to system resources.
  Interface can be graphical, textual or neither.

- ❖ **Applets**

  Restricted access to system resources.
  Interface is embedded in some graphical wrapper.
  Browser.
  Applet viewer.



Y-Max College (YMC)

Java Compilation & Execution

Text Editor

Java Complier

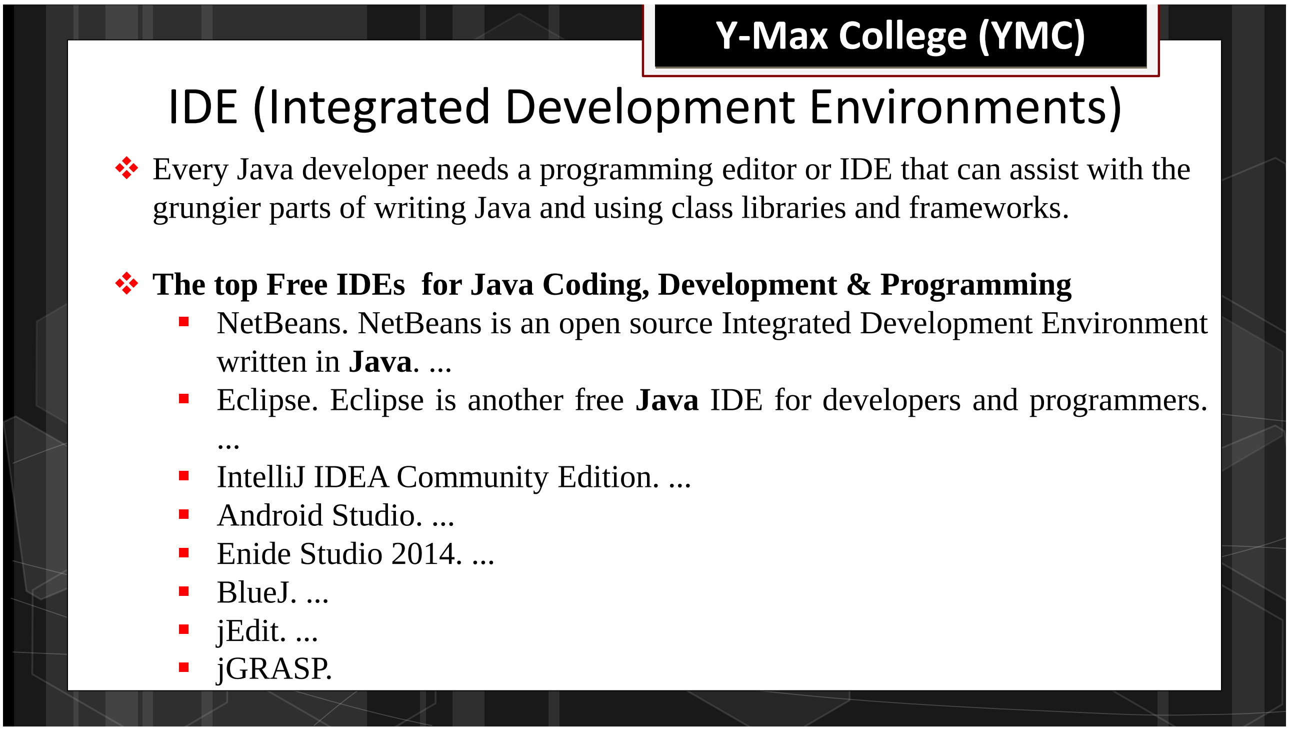
Java Virtual Machine JVM

Java interpreter

Source Code (*.Java files)

Bytecodes (*.Java files)

Operating System



Y-Max College (YMC)

IDE (Integrated Development Environments)

- Every Java developer needs a programming editor or IDE that can assist with the grungier parts of writing Java and using class libraries and frameworks.

- **The top Free IDEs for Java Coding, Development & Programming**
  - NetBeans. NetBeans is an open source Integrated Development Environment written in **Java**. ...
  - Eclipse. Eclipse is another free **Java** IDE for developers and programmers. ...
  - IntelliJ IDEA Community Edition. ...
  - Android Studio. ...
  - Enide Studio 2014. ...
  - BlueJ. ...
  - jEdit. ...
  - jGRASP.

Y-Max College (YMC)

Java Editor

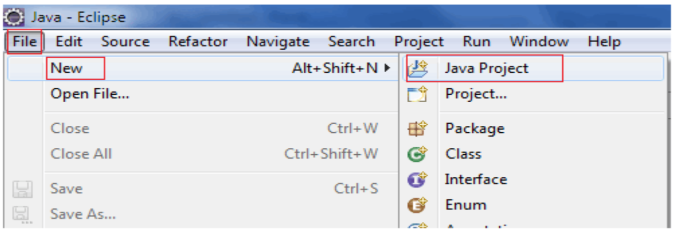
Release	Rename name	Release Year
4.9	2018-09	2018
4.8	Photon	2018
4.7	Oxygen	2017
4.6	Neon	2016
4.5	Mars	2015
4.4	Luna	2014
4.3	Kepler	2013
4.2	Juno	2012

Y-Max College (YMC)

First Java Program

Java Hello World Example using Eclipse IDE

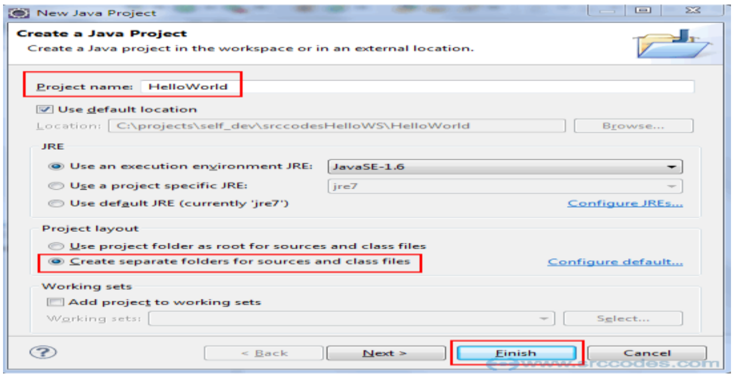
- Create Java Project
  Select from the menu file ⟶ **New** ⟶ **Java Project**



Y-Max College (YMC)

Create Java Project

- Enter “**HelloWorld**” as the Project name. Keep Rest of settings as it is as shown in the following screenshot.



New Java Project

Create a Java Project

Create a Java project in the workspace or in an external location.

Project name: HelloWorld

☑ Use default location

Location: C:\projects\self_dev\srccodesHelloWS\HelloWorld

JRE

● Use an execution environment JRE: JavaSE-1.6

○ Use a project specific JRE: jre7

○ Use default JRE (currently 'jre7')

Project layout

○ Use project folder as root for sources and class files

● Create separate folders for sources and class files

Working sets

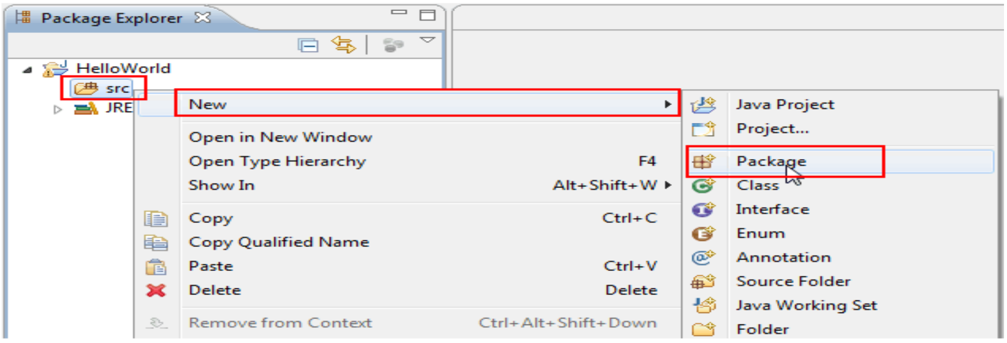
□ Add project to working sets

Working sets:

Y-Max College (YMC)

Create Java Packages

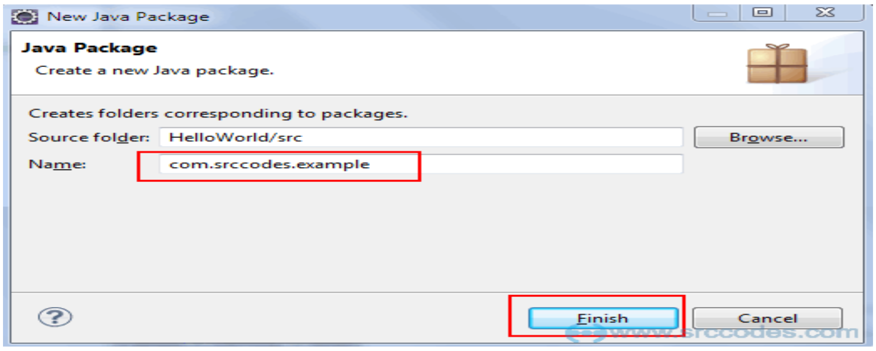
- Right Click on “src” folder and select from the context menu New⟶Package



Y-Max College (YMC)

Create Java Package

- Write ‘**com.srccodes.example**’ in the ‘**Name**’ field and click ‘**Finish**’ button.



New Java Package

Java Package

Create a new Java package.

Creates folders corresponding to packages.

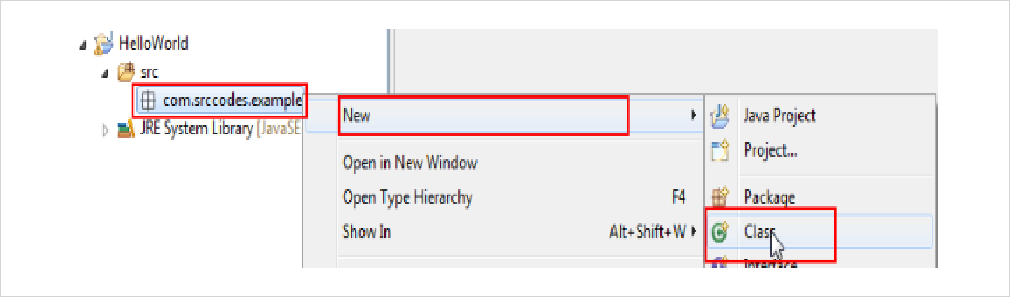
Source folder: HelloWorld/src

Name: com.srccodes.example

Y-Max College (YMC)

Create Java Class

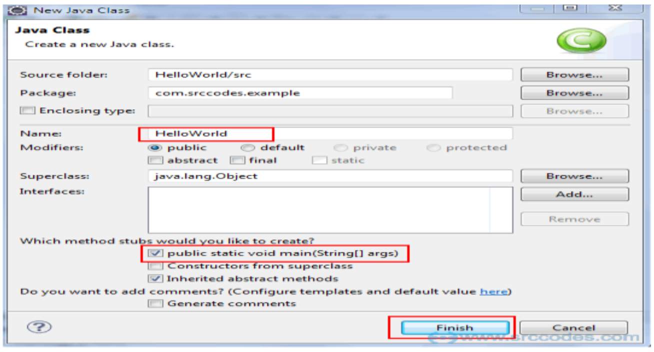
- Write ‘**com.srccodes.example**’ package and select from context menu **New ⟶ Class**



Y-Max College (YMC)

Create Java Class

- Write “**HelloWorld**” in the ‘**Name**’ field and select the check-box for ‘public static void main(String[] args)



New Java Class

Java Class

Create a new Java class.

Source folder: HelloWorld/src

Package: com.srccodes.example

□ Enclosing type:

Name: HelloWorld

Modifiers: ● public ○ default ○ private ○ protected

□ abstract □ final □ static

Superclass: java.lang.Object

Interfaces:

Which method stubs would you like to create?

☑ public static void main(String[] args)

□ Constructors from superclass

☑ Inherited abstract methods

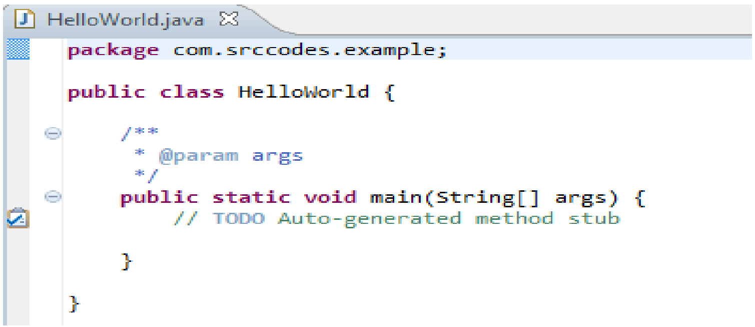
Do you want to add comments? (Configure templates and default value here)

□ Generate comments

Y-Max College (YMC)

Create Java Class

- Eclipse will generate a java class and open the same in the java editor as shown below.



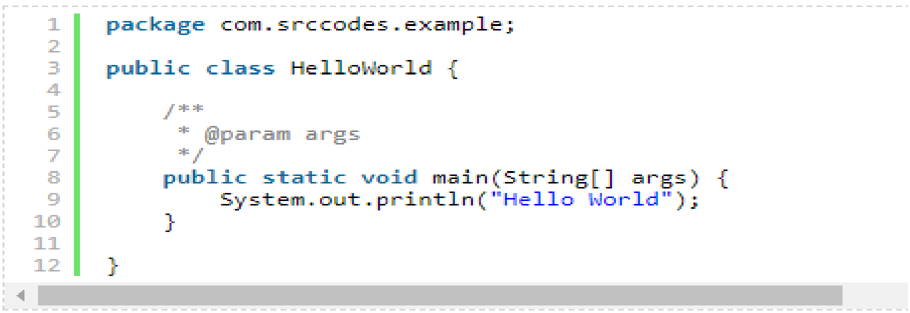
```
package com.srccodes.example;

public class HelloWorld {

	/**
	 * @param args
	 */
	public static void main(String[] args) {
		// TODO Auto-generated method stub

	}

}
```

Y-Max College (YMC)

Create Java Code

- Edit the generated ‘**HelloWorld**’ java class as per the following code.

File : HelloWorld.java



```
package com.srccodes.example;

public class HelloWorld {

    /**
     * @param args
     */
    public static void main(String[] args) {
        System.out.println("Hello World");
    }

}
```

Y-Max College (YMC)

public static void main(String args[])

- **Public**: The keyword “Public” is an access specifier that declares the main method as unprotected.
- **Static**: It says this method belongs to the entire class and NOT a part of any objects of class. The main must always be declared static since the interpreter uses this before any objects are created.
- **Void**: The type modifier that states that main does not return any value.
- A program must include a *method* called **main** where the program starts. The argument to main must always be a string array (containing any command line arguments).

Y-Max College (YMC)

System.out.println(“HelloWorld”);

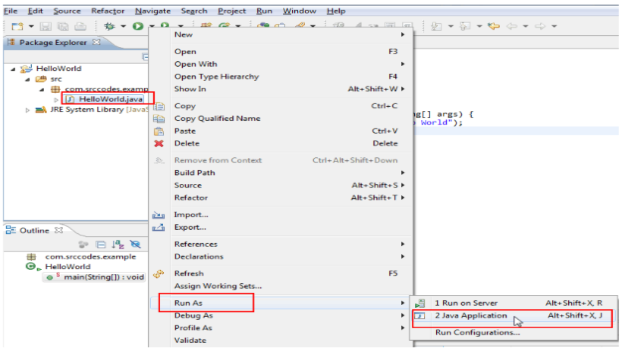
- java.lang.*

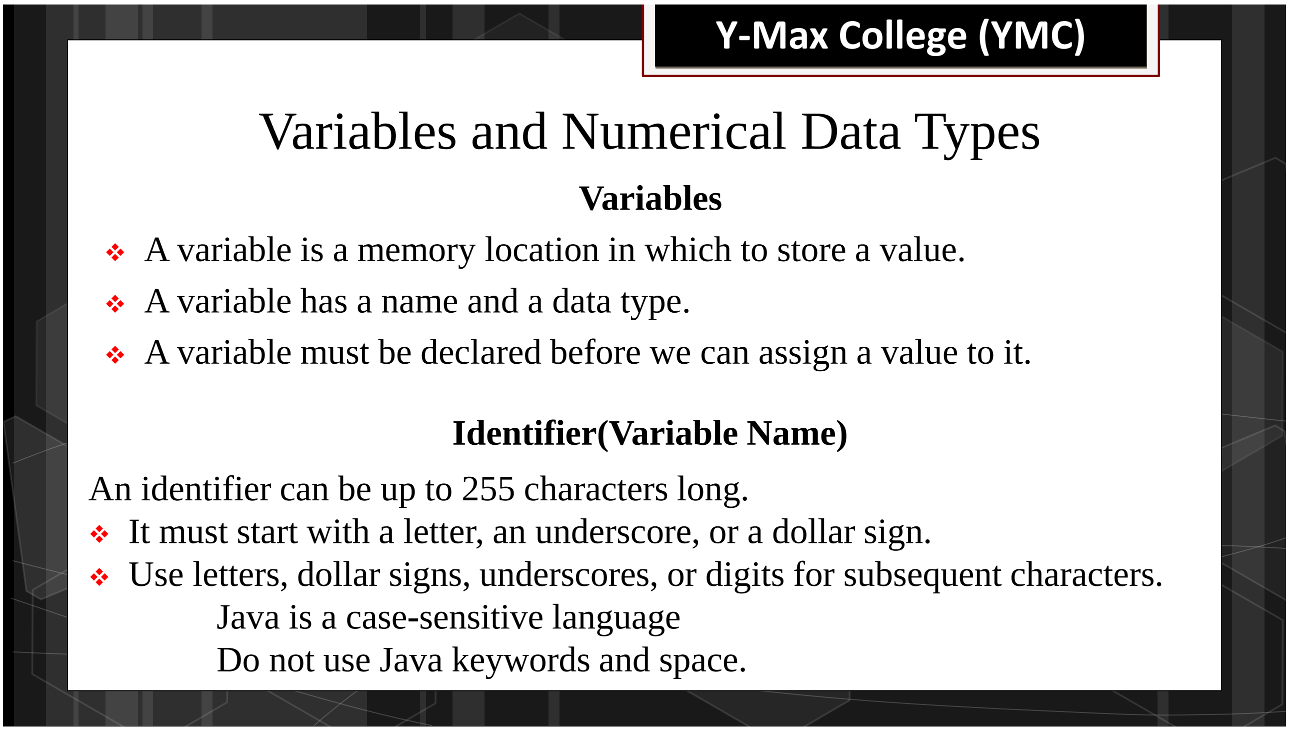
  All classes in “lang” package of java package.
- System is really the java.lang.System class.
- This class has a public static field called out which is an instance of the java.io.PrintStream class. So when we write System.out.println(), we are really invoking the println() method of the “out” field of the java.lang.System class.

Y-Max College (YMC)

Run Your Code

- Right click on ‘HelloWorld.java’ and select from context menu ‘Run As’’ ➡ Java Application’.





Y-Max College (YMC)

Variables and Numerical Data Types

Variables

- A variable is a memory location in which to store a value.
- A variable has a name and a data type.
- A variable must be declared before we can assign a value to it.

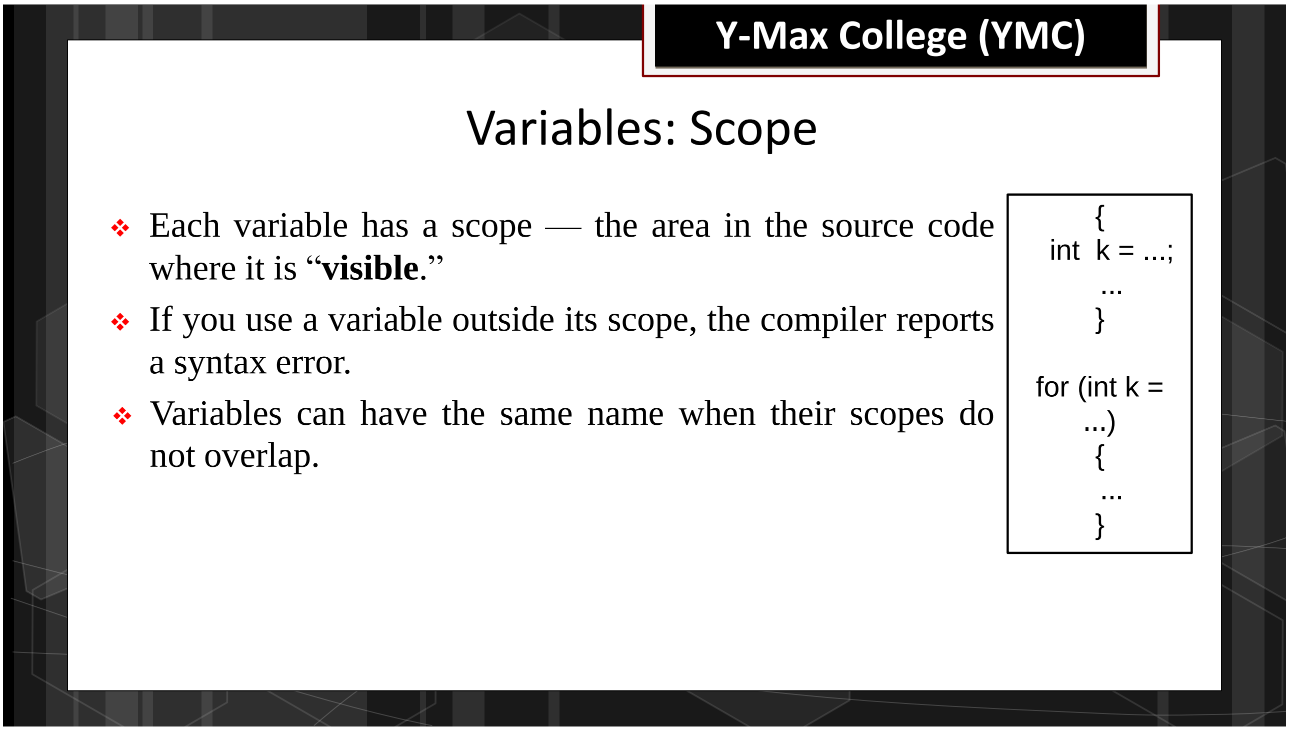
Identifier(Variable Name)

An identifier can be up to 255 characters long.

- It must start with a letter, an underscore, or a dollar sign.
- Use letters, dollar signs, underscores, or digits for subsequent characters.

Java is a case-sensitive language

Do not use Java keywords and space.

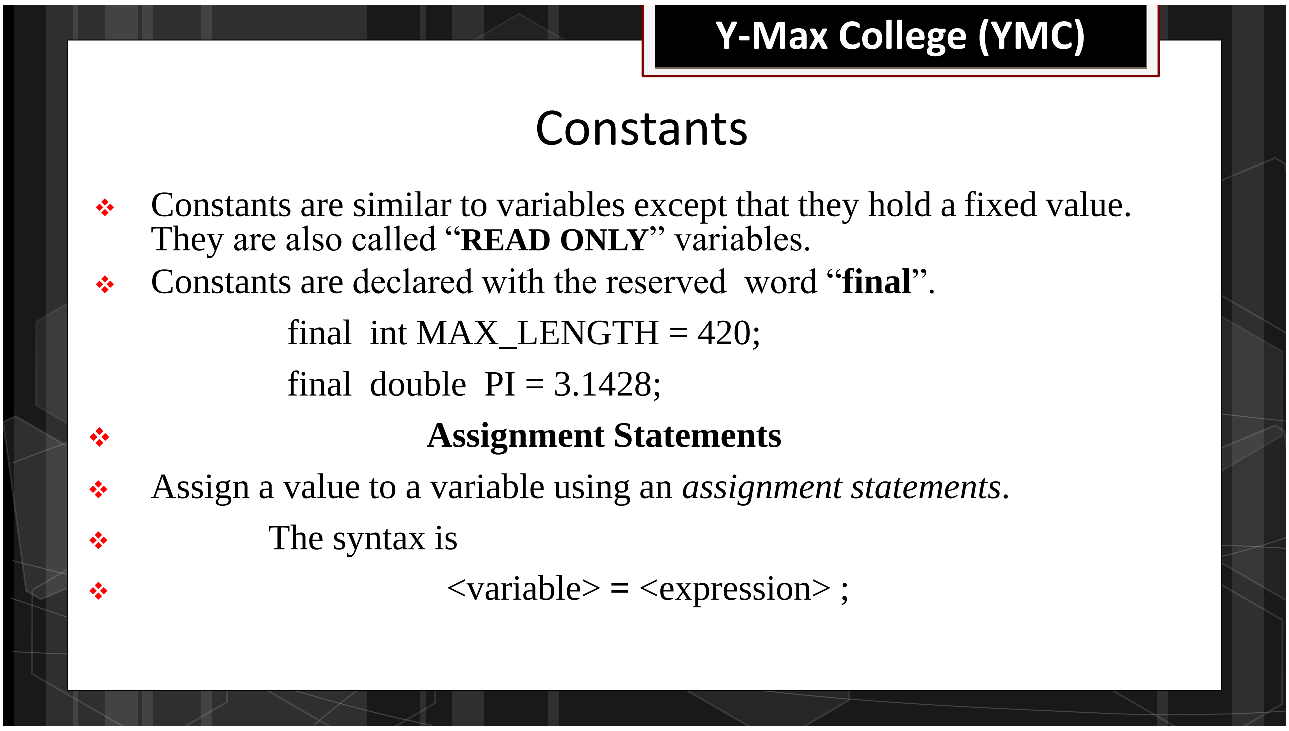


Y-Max College (YMC)

Variables: Scope

- Each variable has a scope — the area in the source code where it is "**visible**."
- If you use a variable outside its scope, the compiler reports a syntax error.
- Variables can have the same name when their scopes do not overlap.

```
{
  int  k = ...;
  ...
}

for (int k =
...)
{
  ...
}
```



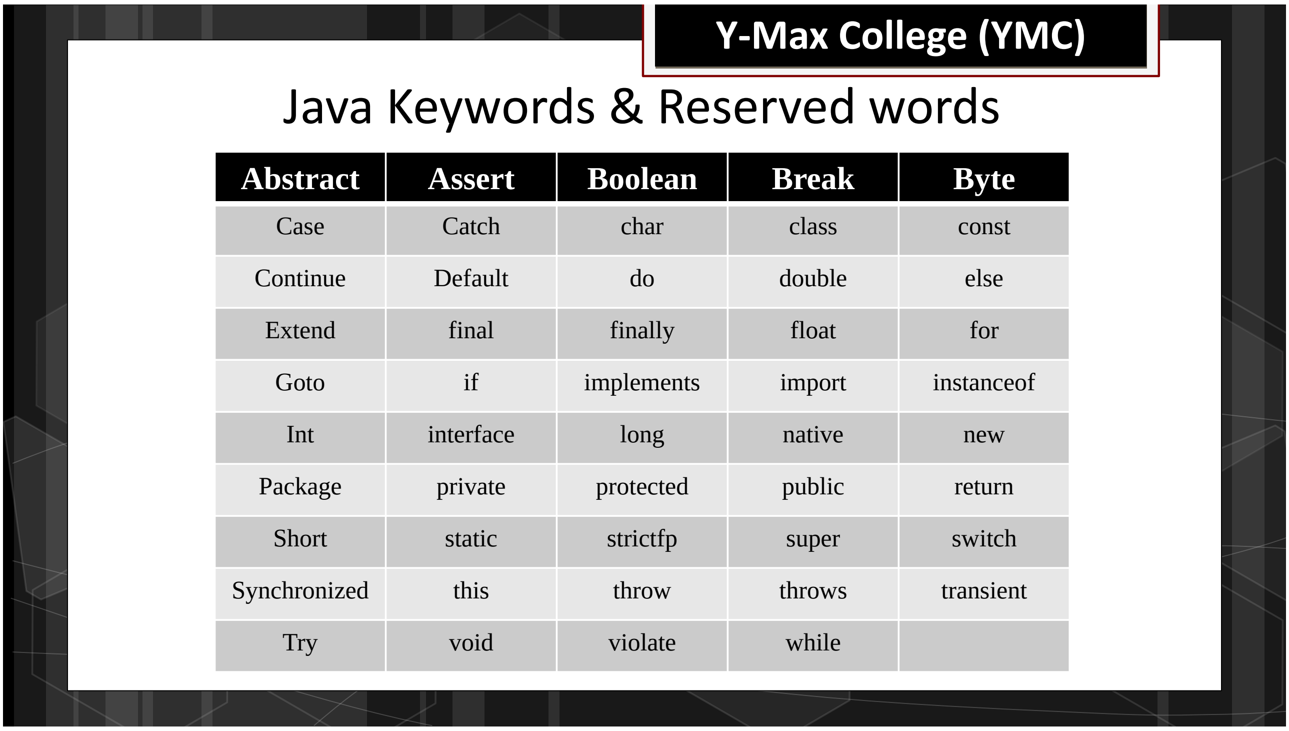
Y-Max College (YMC)

Constants

- Constants are similar to variables except that they hold a fixed value. They are also called “**READ ONLY**” variables.
- Constants are declared with the reserved word “**final**”.

```
final  int MAX_LENGTH = 420;
final  double  PI = 3.1428;
```

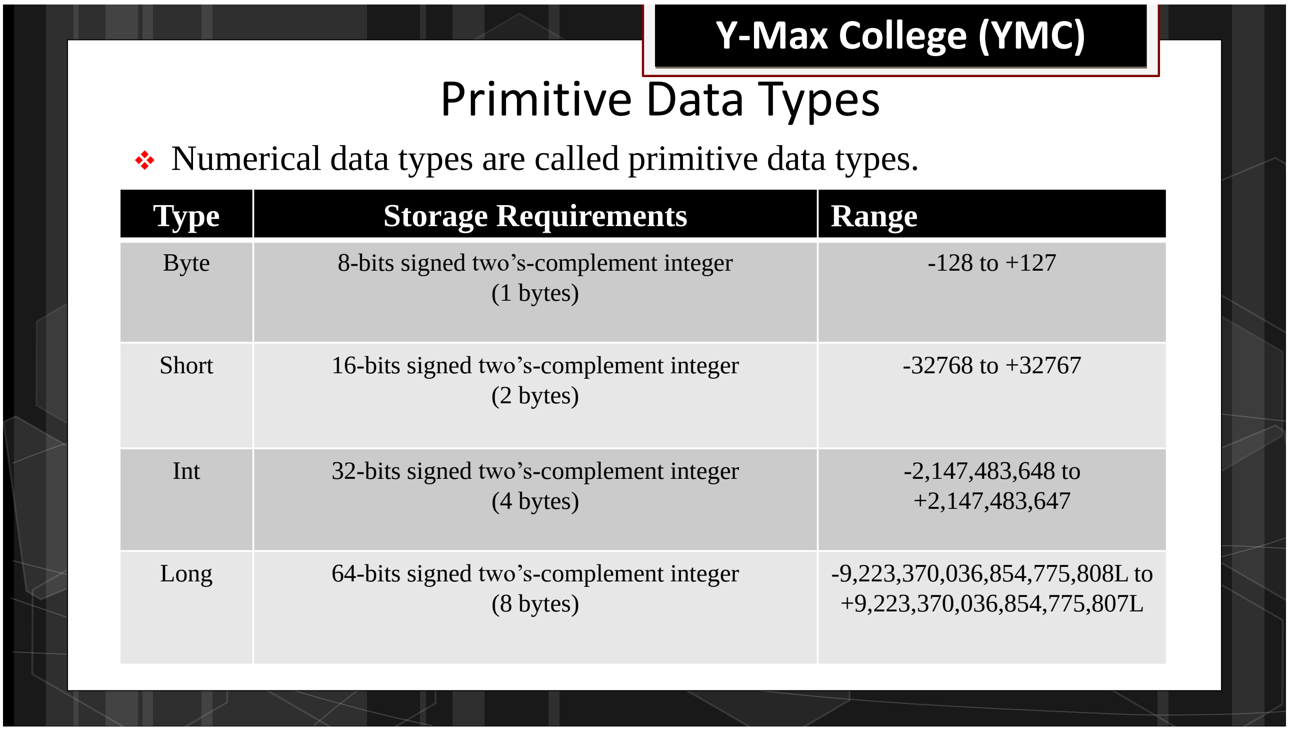
- **Assignment Statements**
- Assign a value to a variable using an *assignment statements*.
- The syntax is
- <variable> = <expression> ;



Y-Max College (YMC)

Java Keywords & Reserved words

Abstract	Assert	Boolean	Break	Byte
Case	Catch	char	class	const
Continue	Default	do	double	else
Extend	final	finally	float	for
Goto	if	implements	import	instanceof
Int	interface	long	native	new
Package	private	protected	public	return
Short	static	strictfp	super	switch
Synchronized	this	throw	throws	transient
Try	void	violate	while	

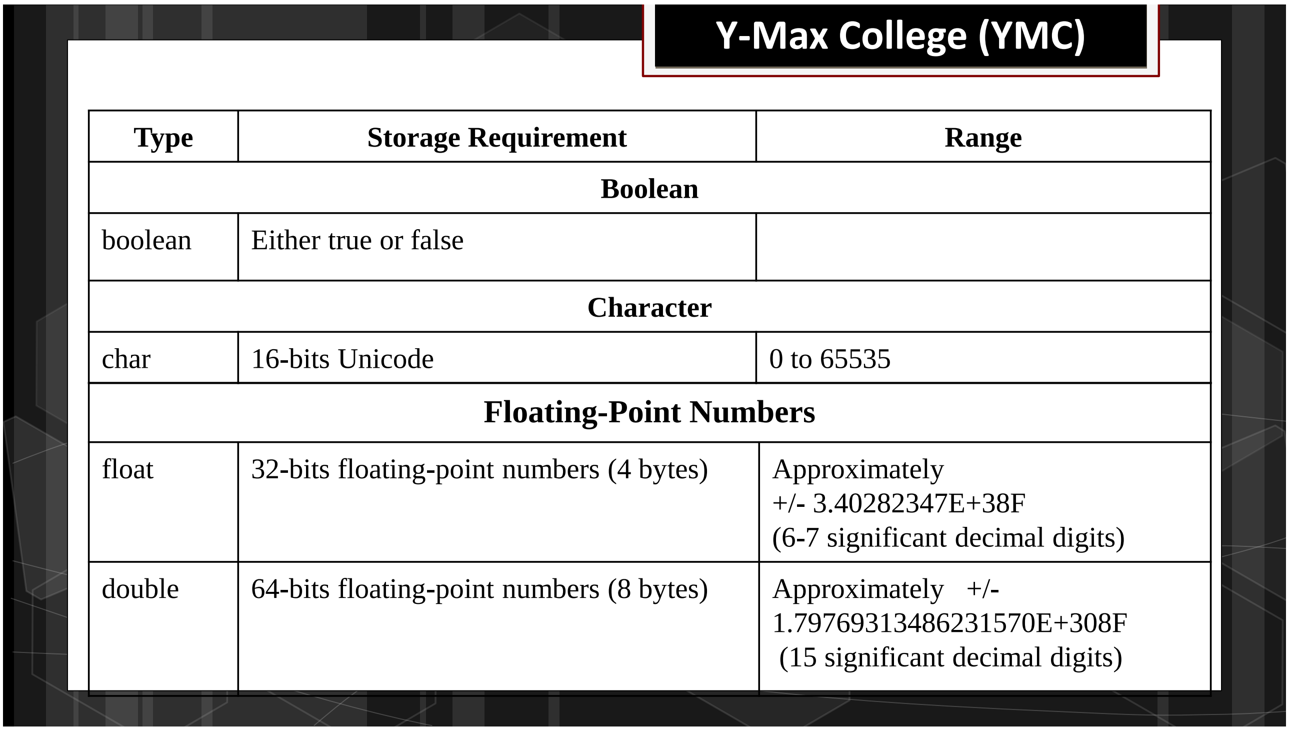


Y-Max College (YMC)

Primitive Data Types

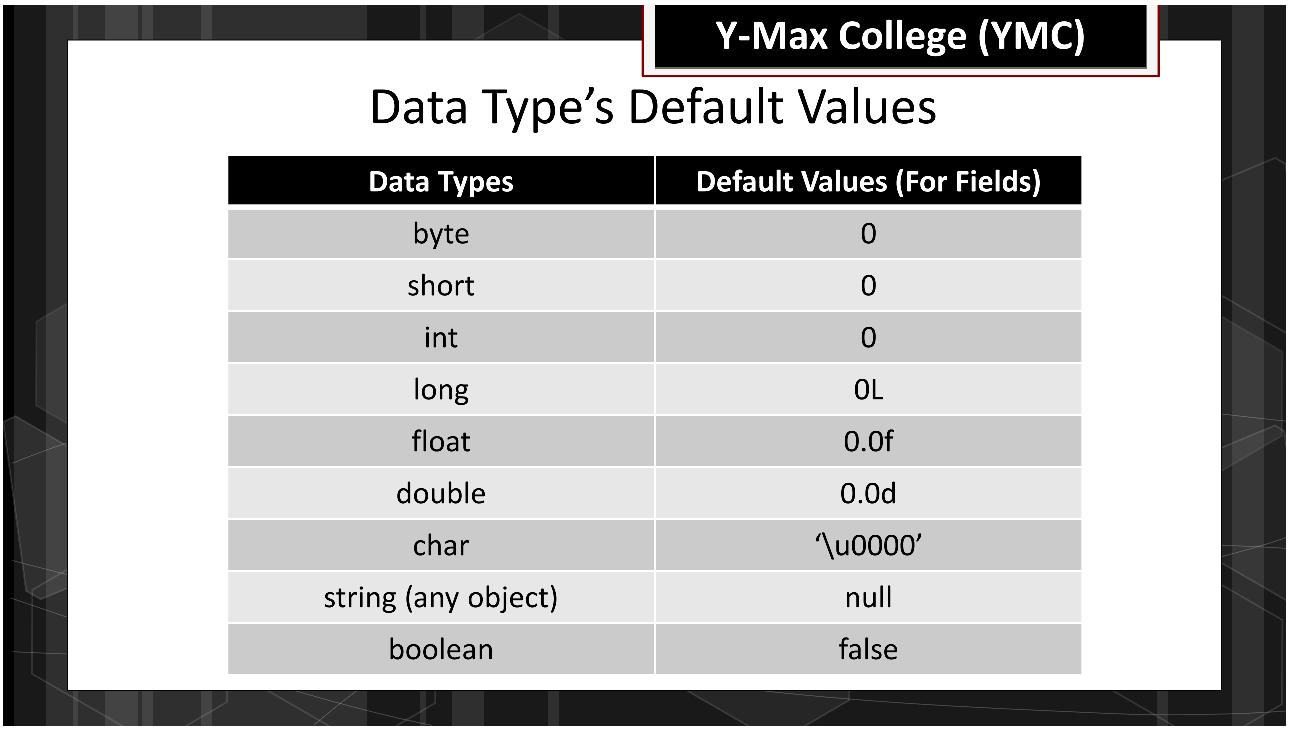
- Numerical data types are called primitive data types.

Type	Storage Requirements	Range
Byte	8-bits signed two's-complement integer (1 bytes)	-128 to +127
Short	16-bits signed two's-complement integer (2 bytes)	-32768 to +32767
Int	32-bits signed two's-complement integer (4 bytes)	-2,147,483,648 to +2,147,483,647
Long	64-bits signed two's-complement integer (8 bytes)	-9,223,370,036,854,775,808L to +9,223,370,036,854,775,807L



Y-Max College (YMC)

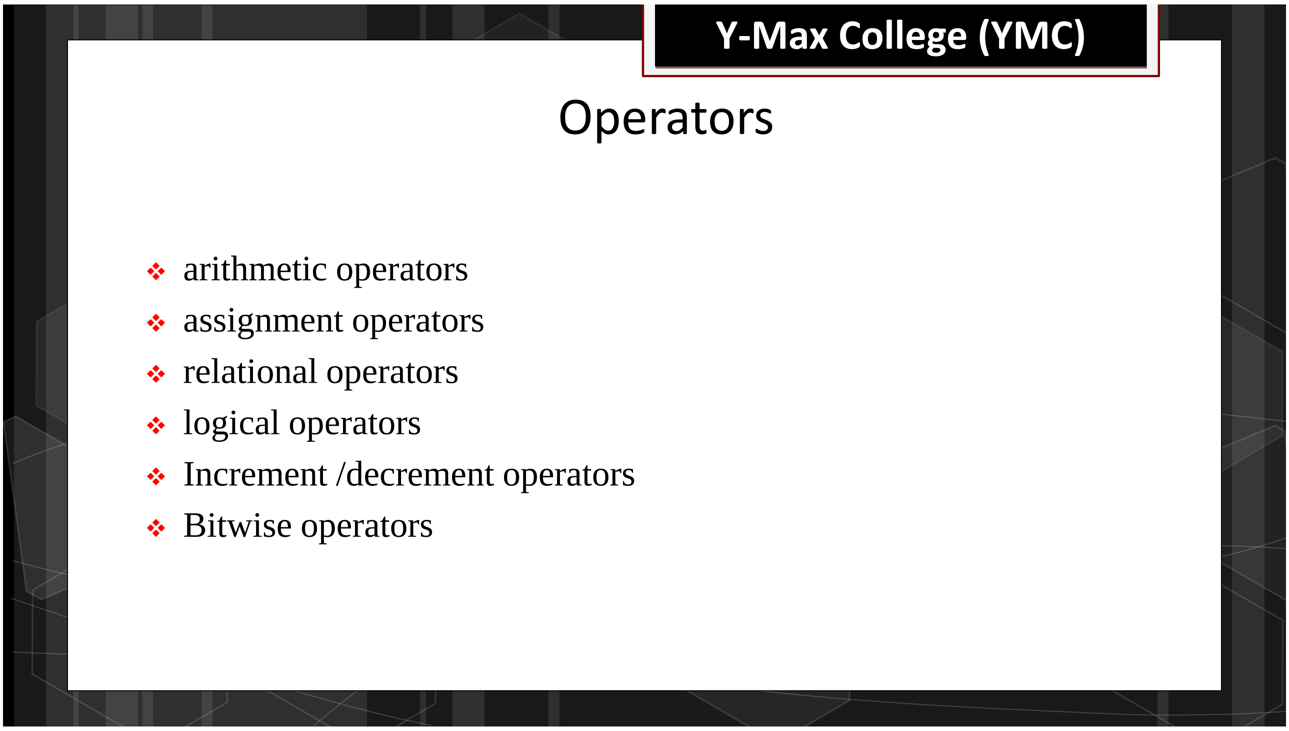
Type	Storage Requirement	Range
Boolean		
boolean	Either true or false	
Character		
char	16-bits Unicode	0 to 65535
Floating-Point Numbers		
float	32-bits floating-point numbers (4 bytes)	Approximately +/- 3.40282347E+38F (6-7 significant decimal digits)
double	64-bits floating-point numbers (8 bytes)	Approximately +/- 1.79769313486231570E+308F (15 significant decimal digits)



Y-Max College (YMC)

Data Type’s Default Values

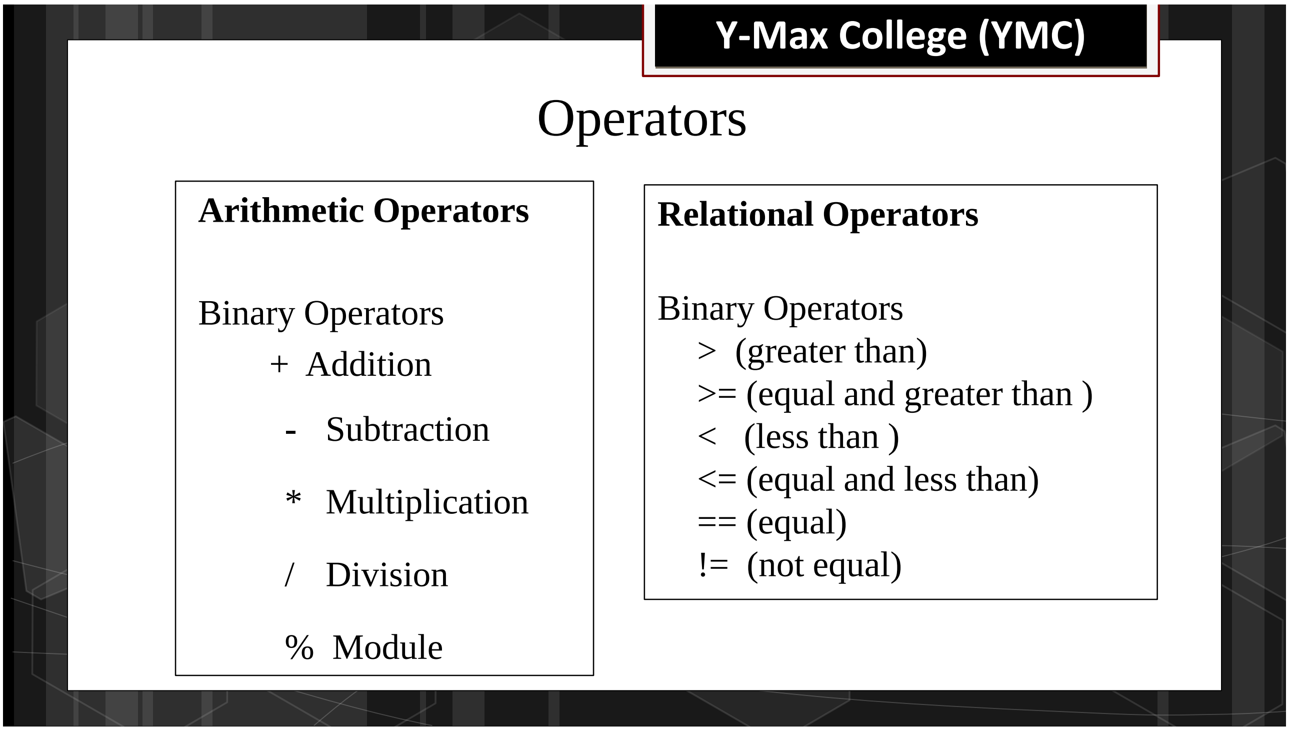
Data Types	Default Values (For Fields)
byte	0
short	0
int	0
long	0L
float	0.0f
double	0.0d
char	‘\u0000’
string (any object)	null
boolean	false



Y-Max College (YMC)

Operators

- arithmetic operators
- assignment operators
- relational operators
- logical operators
- Increment /decrement operators
- Bitwise operators



Y-Max College (YMC)

Operators

Arithmetic Operators

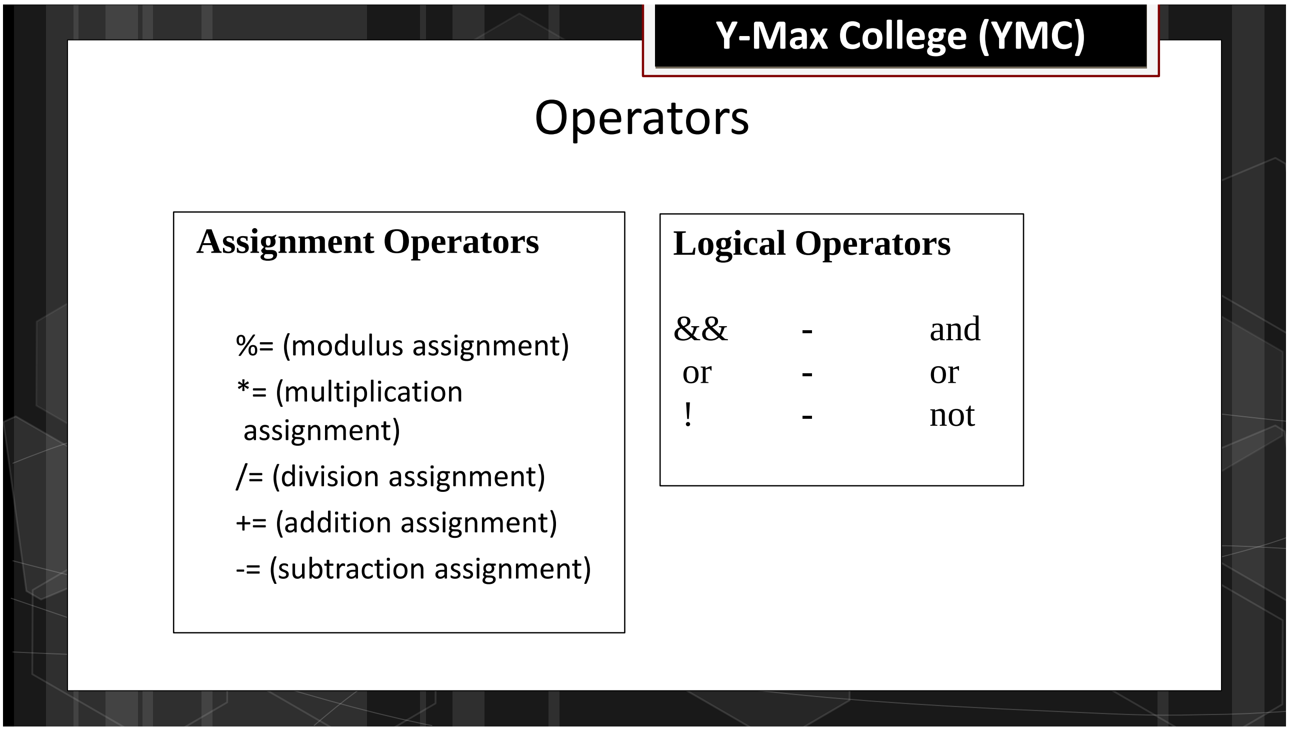
Binary Operators

+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Module

Relational Operators

Binary Operators

>	(greater than)
>=	(equal and greater than)
<	(less than)
<=	(equal and less than)
==	(equal)
!=	(not equal)



Y-Max College (YMC)

Operators

Assignment Operators

%= (modulus assignment)

*= (multiplication assignment)

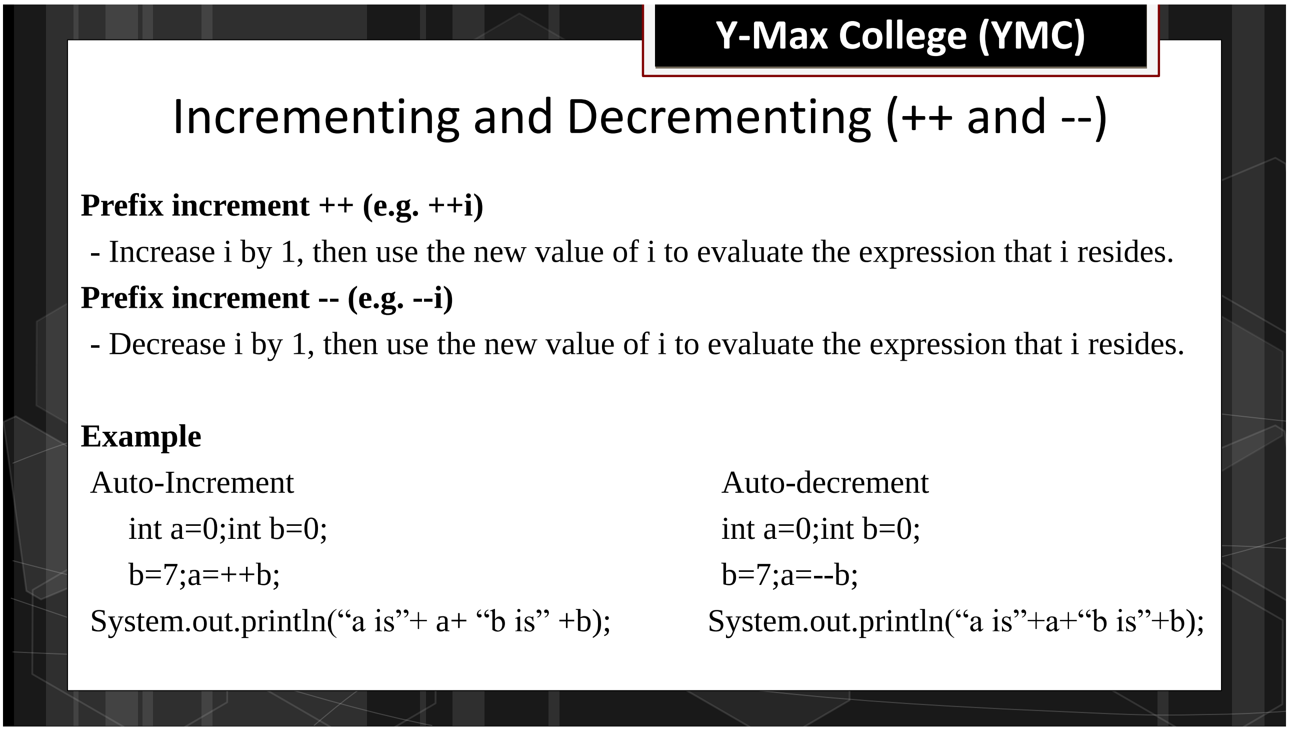
/= (division assignment)

+= (addition assignment)

-= (subtraction assignment)

Logical Operators

&&	-	and
or	-	or
!	-	not



Y-Max College (YMC)

Incrementing and Decrementing (++ and --)

Prefix increment ++ (e.g. ++i)

- Increase i by 1, then use the new value of i to evaluate the expression that i resides.

Prefix increment -- (e.g. --i)

- Decrease i by 1, then use the new value of i to evaluate the expression that i resides.

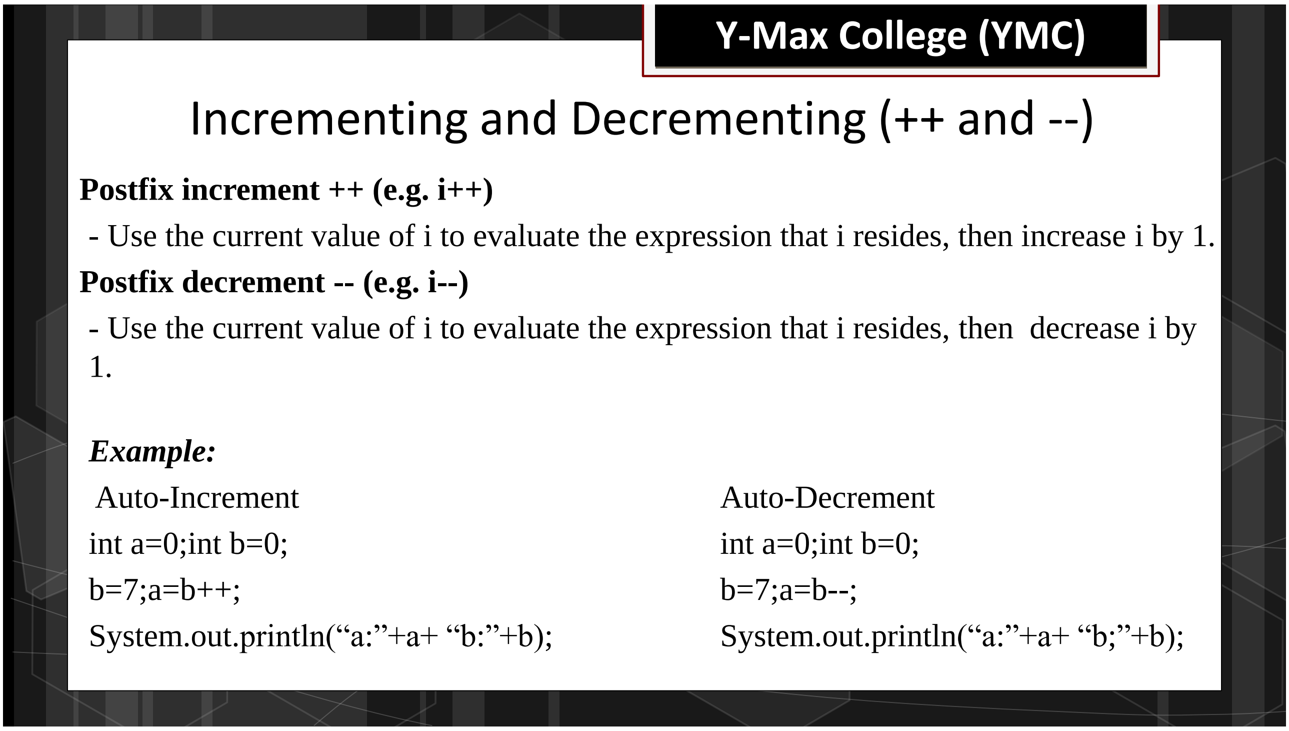
Example

Auto-Increment

```
    int a=0;int b=0;
    b=7;a=++b;
System.out.println("a is"+ a+ "b is" +b);
```

Auto-decrement

```
int a=0;int b=0;
b=7;a=--b;
System.out.println("a is"+a+"b is"+b);
```



Y-Max College (YMC)

Incrementing and Decrementing (++ and --)

Postfix increment ++ (e.g. i++)

- Use the current value of i to evaluate the expression that i resides, then increase i by 1.

Postfix decrement -- (e.g. i--)

- Use the current value of i to evaluate the expression that i resides, then decrease i by 1.

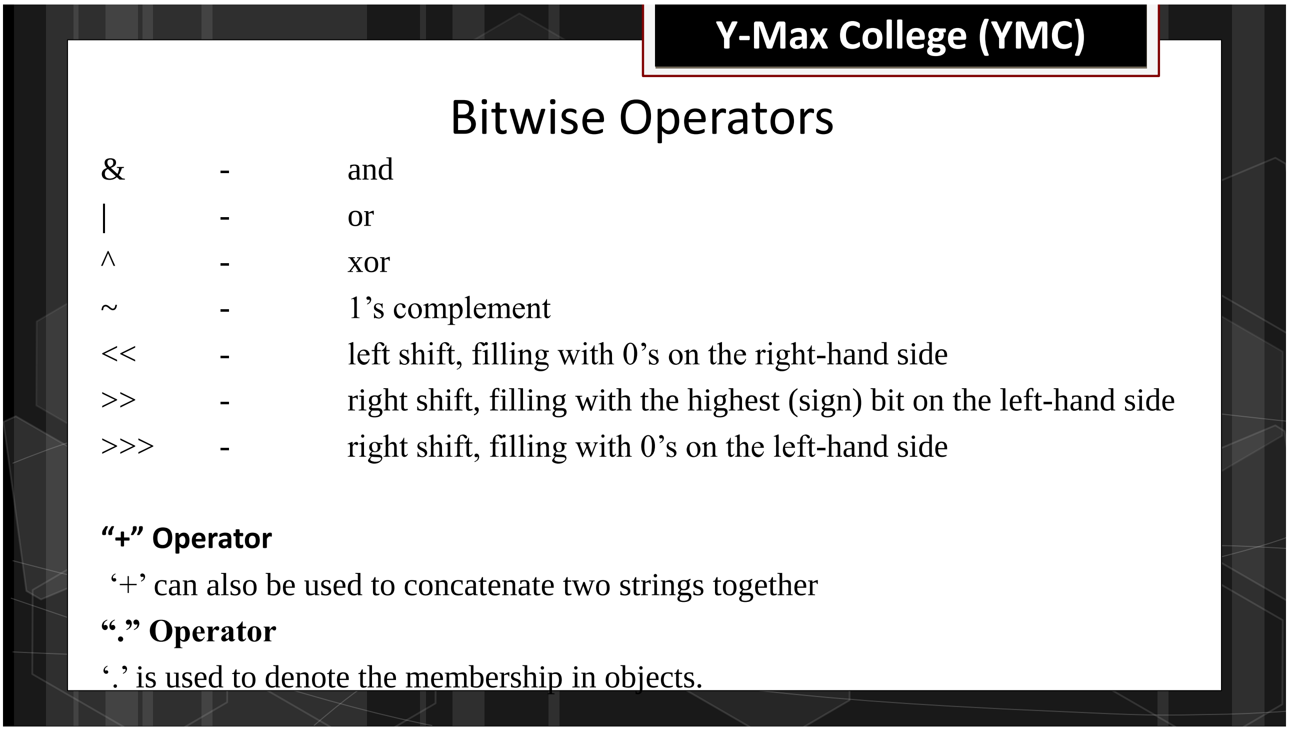
Example:

Auto-Increment

```
int a=0;int b=0;
b=7;a=b++;
System.out.println("a:"+a+ "b:"+b);
```

Auto-Decrement

```
int a=0;int b=0;
b=7;a=b--;
System.out.println("a:"+a+ "b;"+b);
```



Y-Max College (YMC)

Bitwise Operators

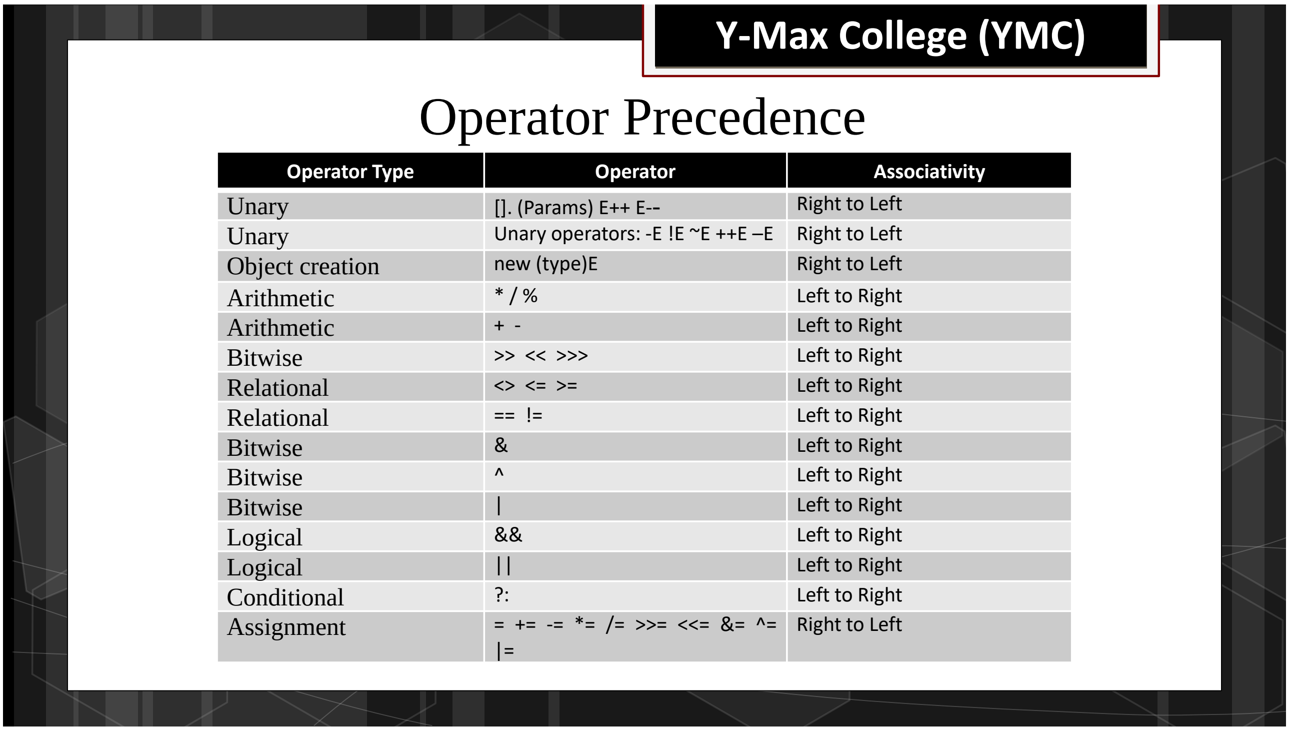
&	-	and
|	-	or
^	-	xor
~	-	1’s complement
<<	-	left shift, filling with 0’s on the right-hand side
>>	-	right shift, filling with the highest (sign) bit on the left-hand side
>>>	-	right shift, filling with 0’s on the left-hand side

“+” Operator

‘+’ can also be used to concatenate two strings together

“.” Operator

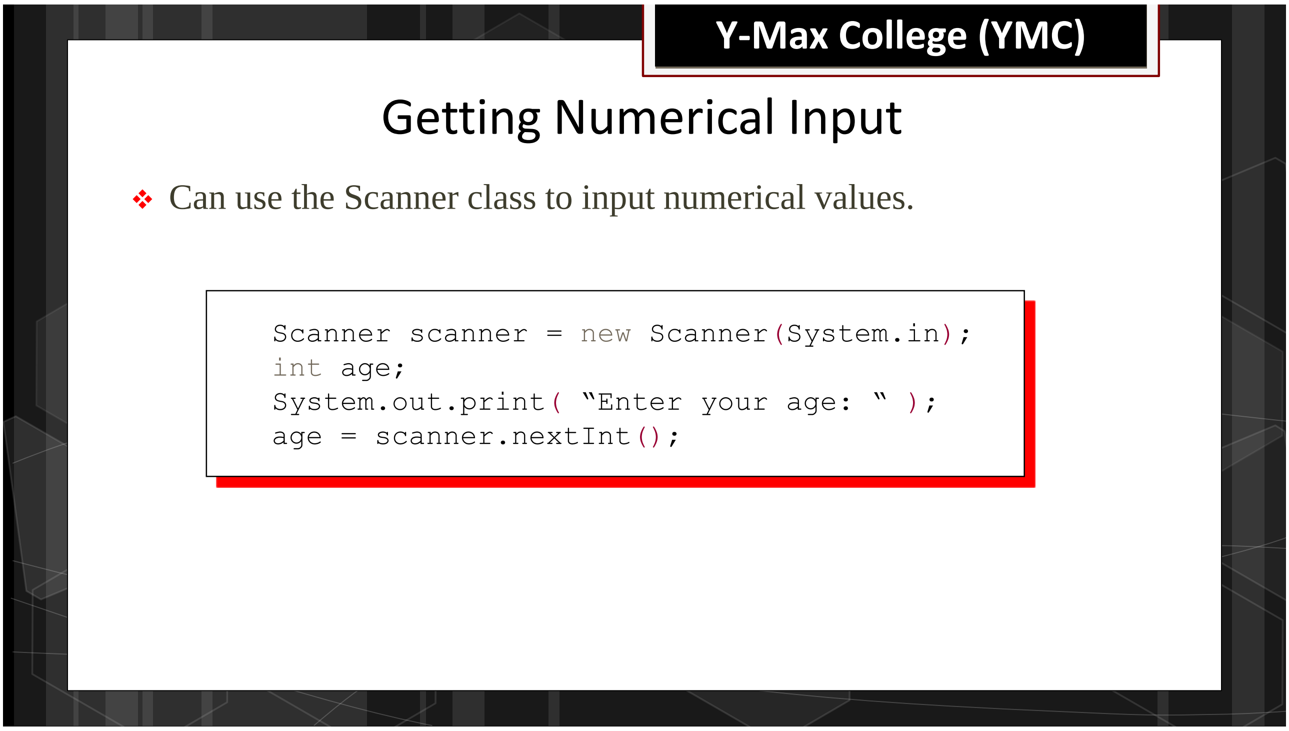
‘.’ is used to denote the membership in objects.



Y-Max College (YMC)

Operator Precedence

Operator Type	Operator	Associativity
Unary	[]. (Params) E++ E--	Right to Left
Unary	Unary operators: -E !E ~E ++E –E	Right to Left
Object creation	new (type)E	Right to Left
Arithmetic	* / %	Left to Right
Arithmetic	+ -	Left to Right
Bitwise	>> << >>>	Left to Right
Relational	<> <= >=	Left to Right
Relational	== !=	Left to Right
Bitwise	&	Left to Right
Bitwise	^	Left to Right
Bitwise	|	Left to Right
Logical	&&	Left to Right
Logical	||	Left to Right
Conditional	?:	Left to Right
Assignment	= += -= *= /= >>= <<= &= ^= |=	Right to Left

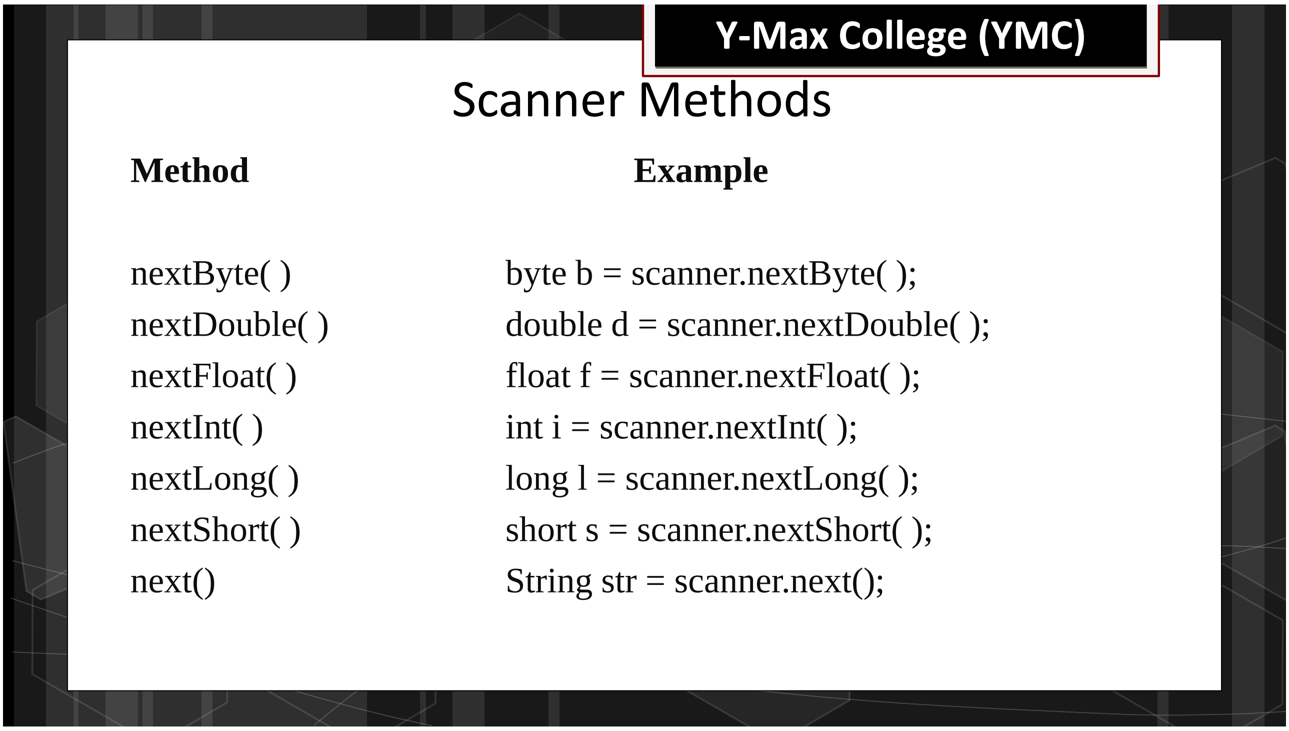


Y-Max College (YMC)

Getting Numerical Input

- Can use the Scanner class to input numerical values.

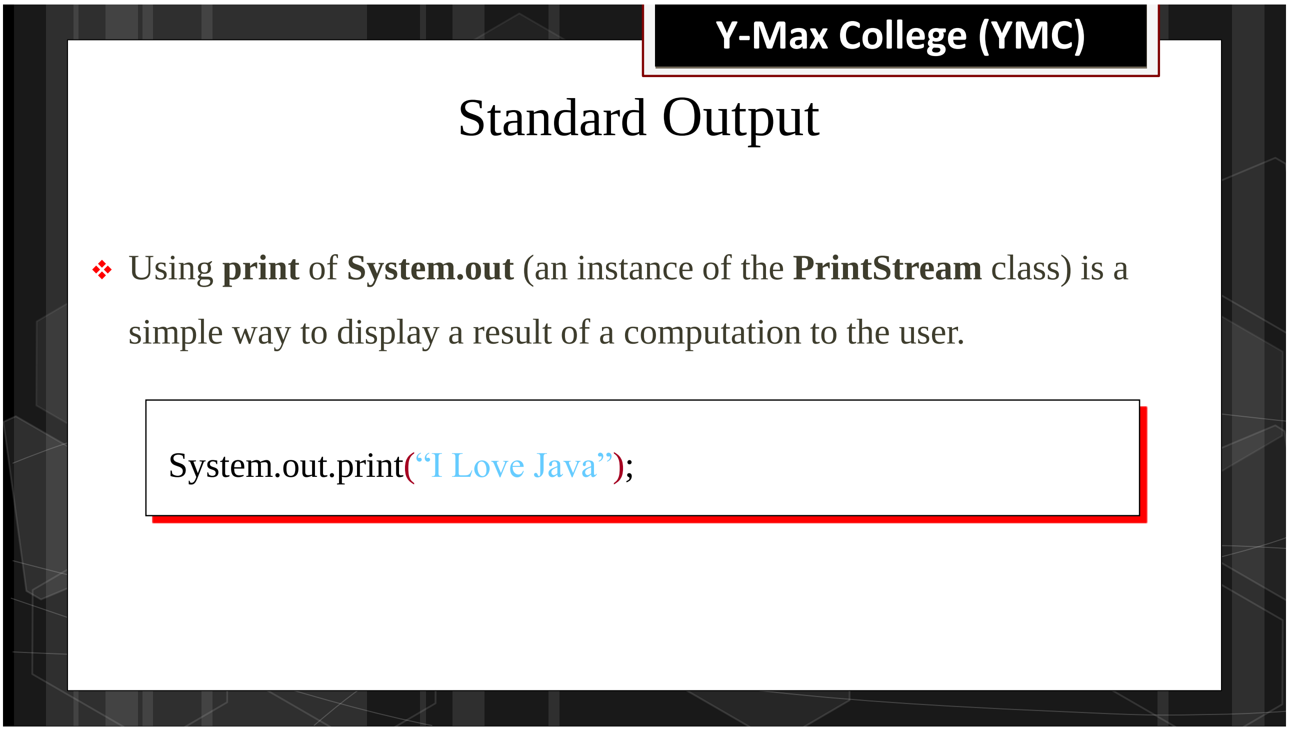
```
Scanner scanner = new Scanner(System.in);
int age;
System.out.print( “Enter your age: ‟ );
age = scanner.nextInt();
```



Y-Max College (YMC)

Scanner Methods

Method	**Example**
nextByte( )	byte b = scanner.nextByte( );
nextDouble( )	double d = scanner.nextDouble( );
nextFloat( )	float f = scanner.nextFloat( );
nextInt( )	int i = scanner.nextInt( );
nextLong( )	long l = scanner.nextLong( );
nextShort( )	short s = scanner.nextShort( );
next()	String str = scanner.next();

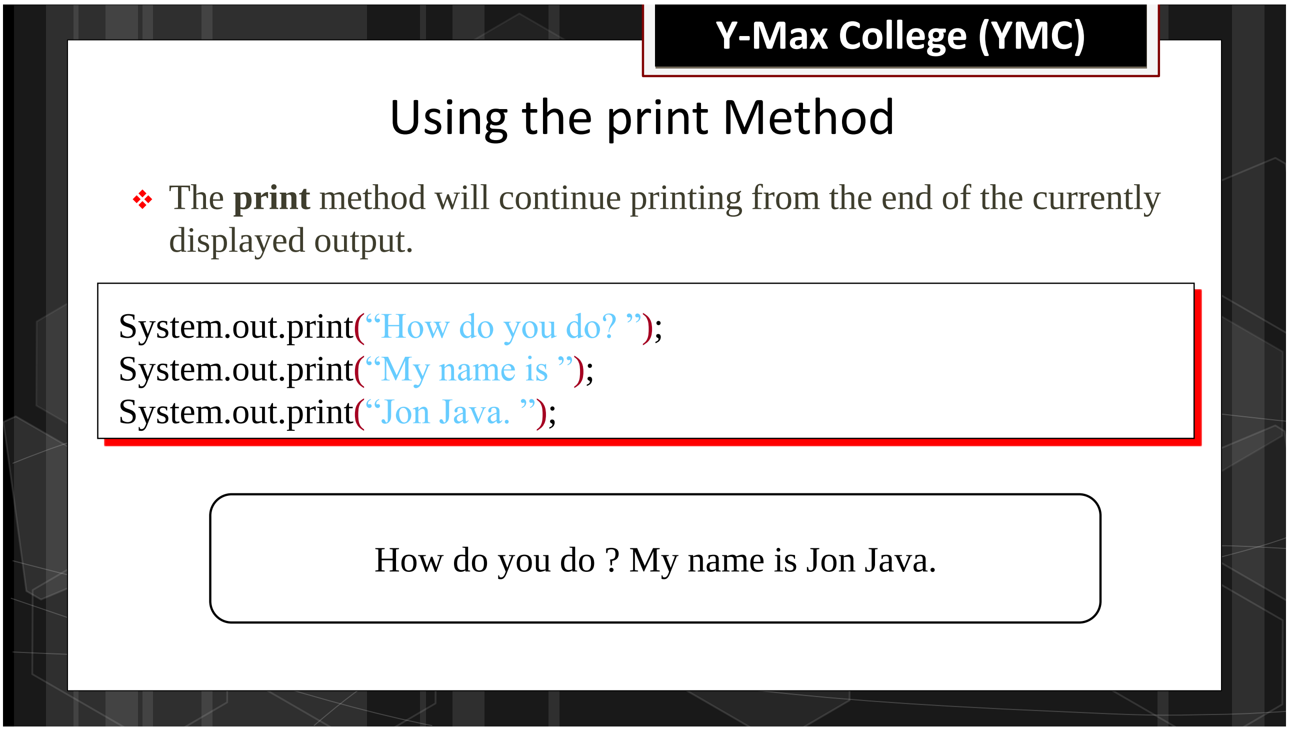


Y-Max College (YMC)

Standard Output

- ❖ Using **print** of **System.out** (an instance of the **PrintStream** class) is a simple way to display a result of a computation to the user.

```
System.out.print(“I Love Java”);
```



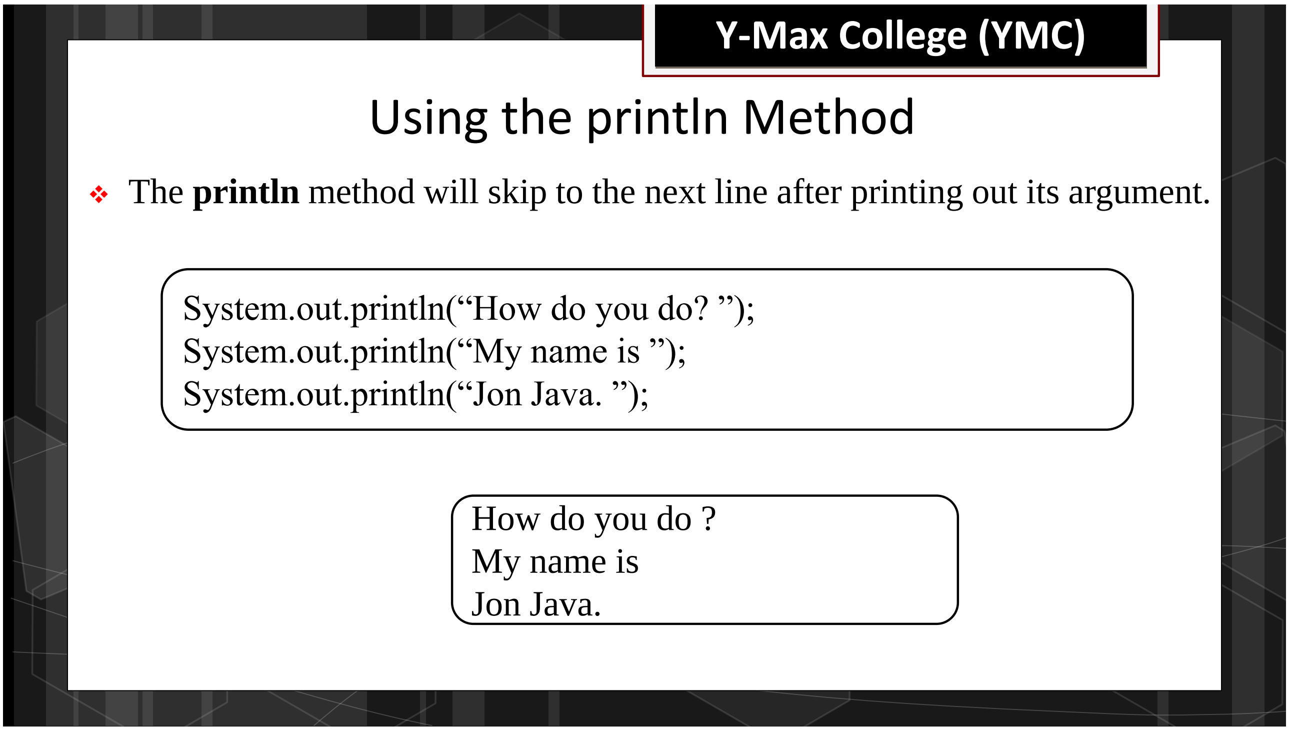
Y-Max College (YMC)

Using the print Method

- ❖ The **print** method will continue printing from the end of the currently displayed output.

```
System.out.print(“How do you do? ”);
System.out.print(“My name is ”);
System.out.print(“Jon Java. ”);
```

```
How do you do ? My name is Jon Java.
```



Y-Max College (YMC)

Using the println Method

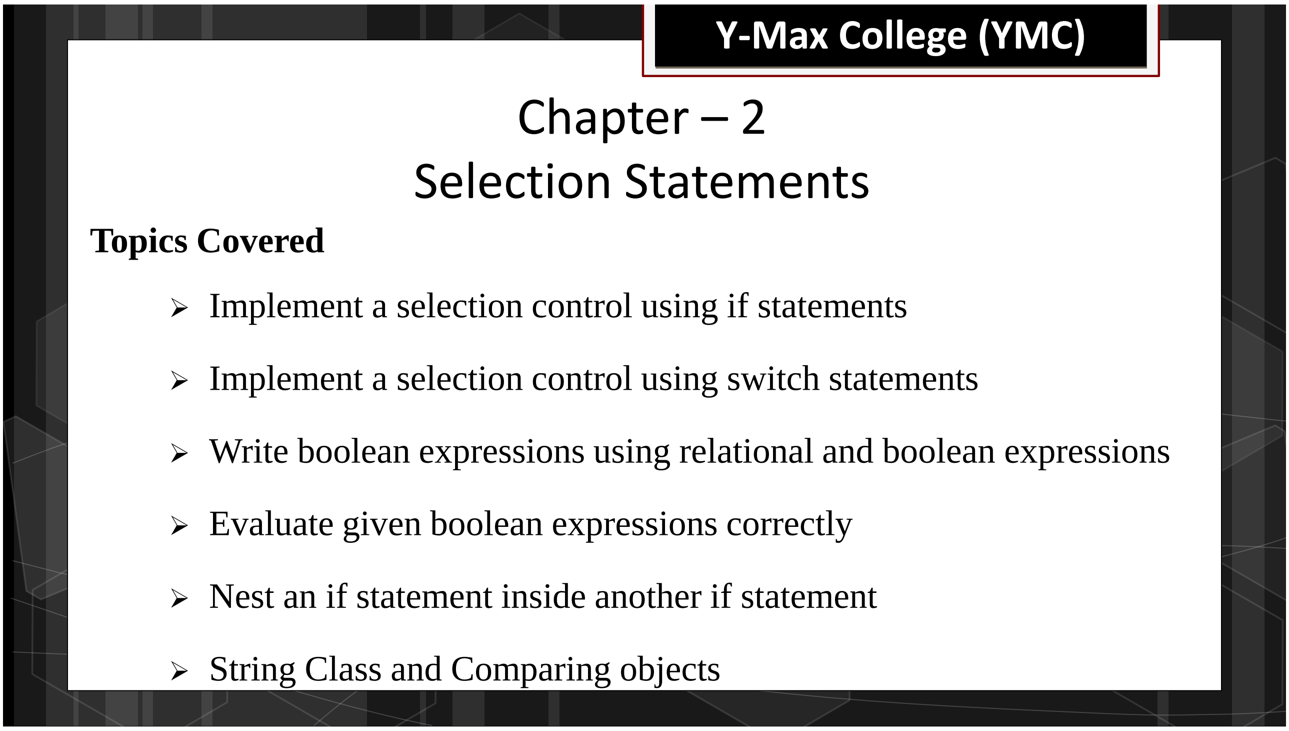
- The **println** method will skip to the next line after printing out its argument.

```
System.out.println(“How do you do? ”);
System.out.println(“My name is ”);
System.out.println(“Jon Java. ”);
```

```
How do you do ?
My name is
Jon Java.
```

Y-Max College (YMC)

Any Questions….?

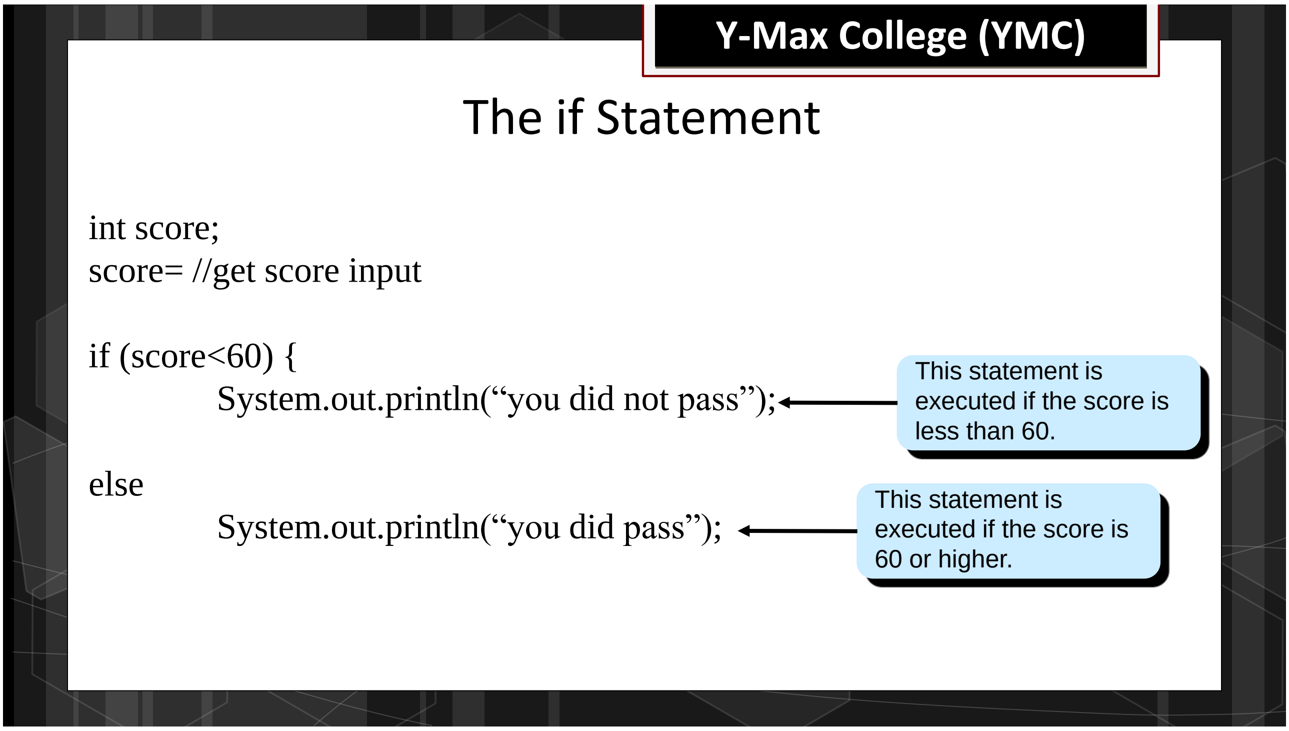


Y-Max College (YMC)

Chapter – 2
Selection Statements

Topics Covered

- Implement a selection control using if statements
- Implement a selection control using switch statements
- Write boolean expressions using relational and boolean expressions
- Evaluate given boolean expressions correctly
- Nest an if statement inside another if statement
- String Class and Comparing objects



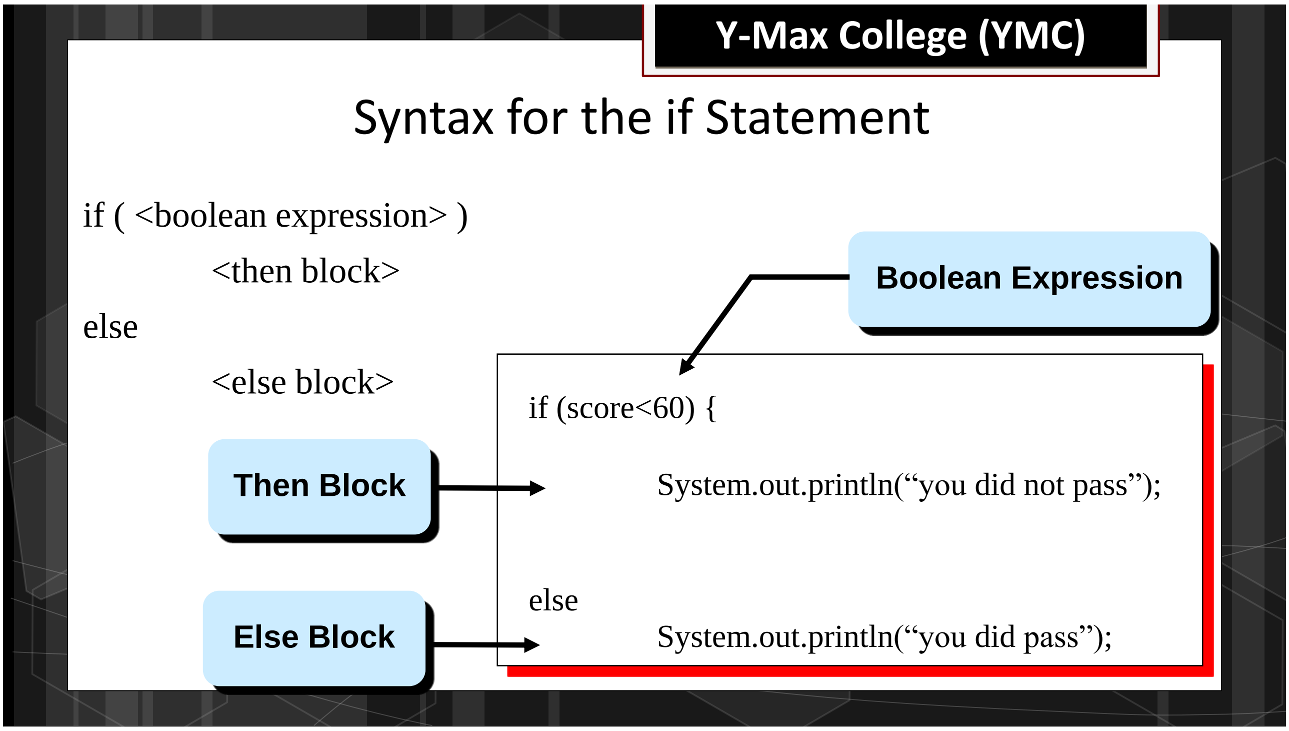
Y-Max College (YMC)

The if Statement

```
int score;
score= //get score input

if (score<60) {
        System.out.println(“you did not pass”);

else
        System.out.println(“you did pass”);
```

This statement is executed if the score is less than 60.

This statement is executed if the score is 60 or higher.



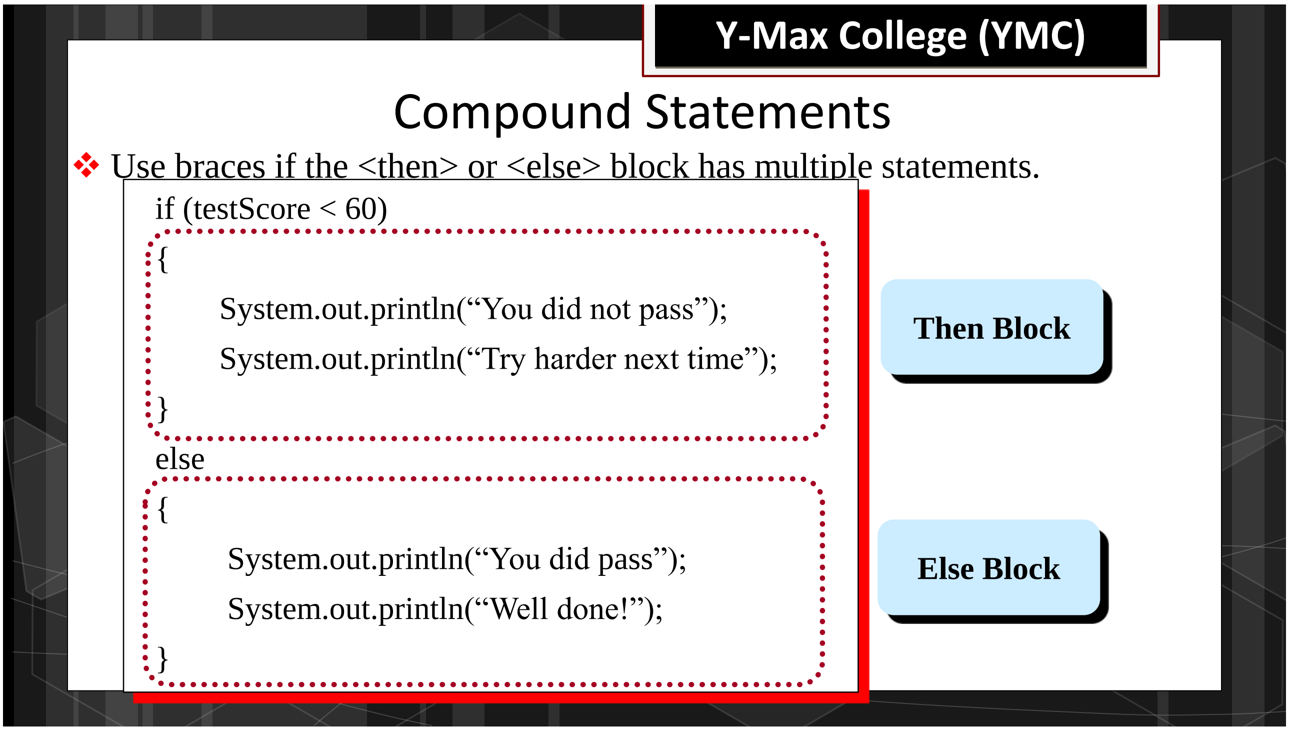
Y-Max College (YMC)

Syntax for the if Statement

```
if ( <boolean expression> )
        <then block>
else
        <else block>
```

```
if (score<60) {
        System.out.println("you did not pass");

else
        System.out.println("you did pass");
```

Boolean Expression
Then Block
Else Block



Y-Max College (YMC)

Compound Statements

- Use braces if the <then> or <else> block has multiple statements.

```
if (testScore < 60)
{
        System.out.println("You did not pass");
        System.out.println("Try harder next time");
}
else
{
        System.out.println("You did pass");
        System.out.println("Well done!");
}
```

Then Block
Else Block

Y-Max College (YMC)

Syntax for if Compound Statements

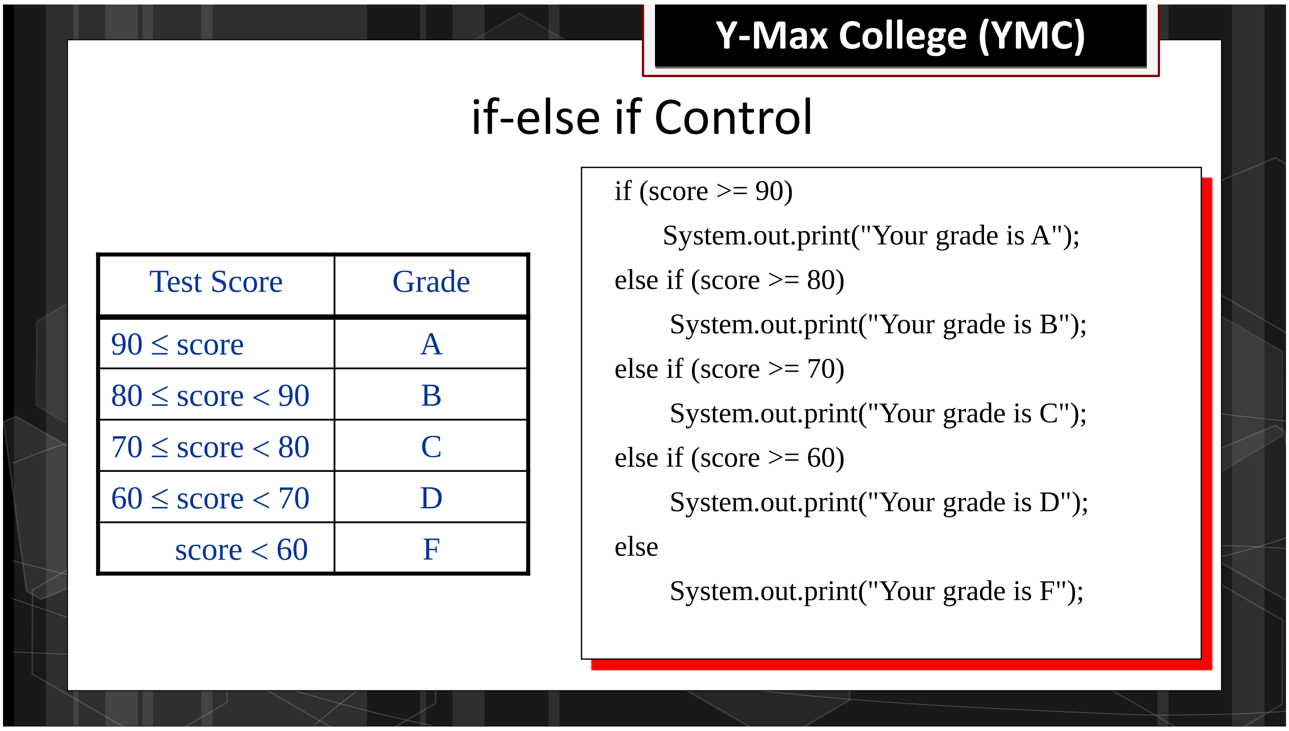
```
if ( <boolean expression> ) {
//statements
   …….
}
else {
//statements
   …….
}
```

Y-Max College (YMC)

The nested-if Statement

- The then and else block of an if statement can contain any valid statements, including other if statements.
- An if statement containing another if statement is called a nested-if statement.

```
if (testScore >= 60) {
      if (studentAge < 10) {
            System.out.println("You did a great job");
      } else {
            System.out.println("You did pass"); //test score >= 60
      }                                          //and age >= 10
} else { //test score < 60
      System.out.println("You did not pass");
}
```

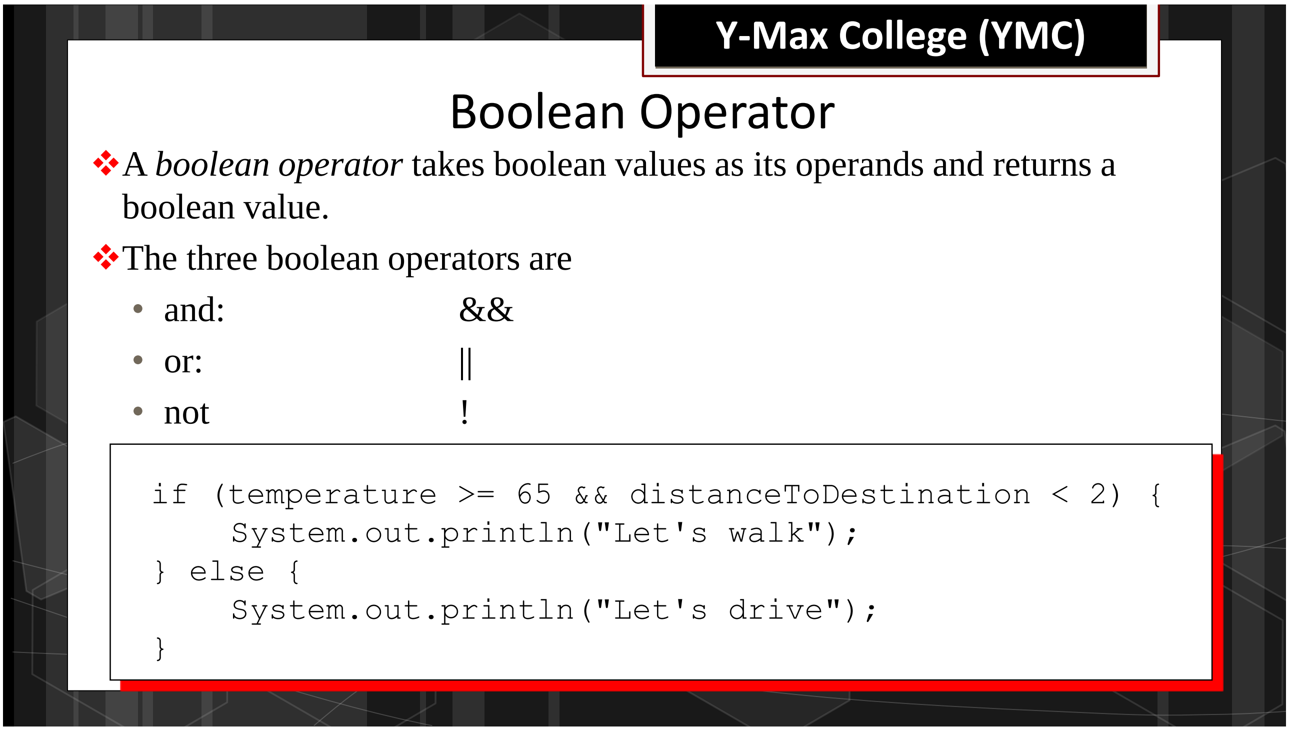


Y-Max College (YMC)

if-else if Control

Test Score	Grade
90 ≤ score	A
80 ≤ score < 90	B
70 ≤ score < 80	C
60 ≤ score < 70	D
score < 60	F

```
if (score >= 90)
    System.out.print("Your grade is A");
else if (score >= 80)
    System.out.print("Your grade is B");
else if (score >= 70)
    System.out.print("Your grade is C");
else if (score >= 60)
    System.out.print("Your grade is D");
else
    System.out.print("Your grade is F");
```



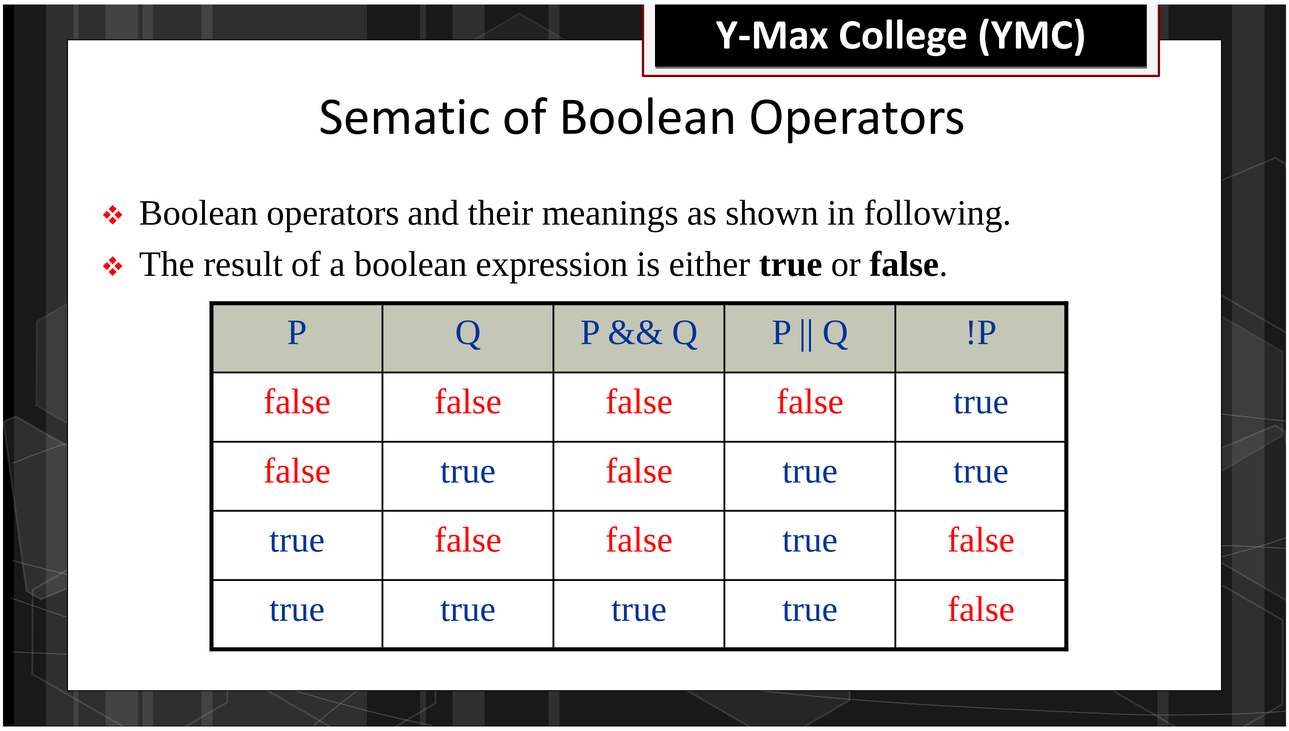
Y-Max College (YMC)

Boolean Operator

- A *boolean operator* takes boolean values as its operands and returns a boolean value.
- The three boolean operators are

and:	&&
or:	||
not	!

```
if (temperature >= 65 && distanceToDestination < 2) {
    System.out.println("Let's walk");
} else {
    System.out.println("Let's drive");
}
```

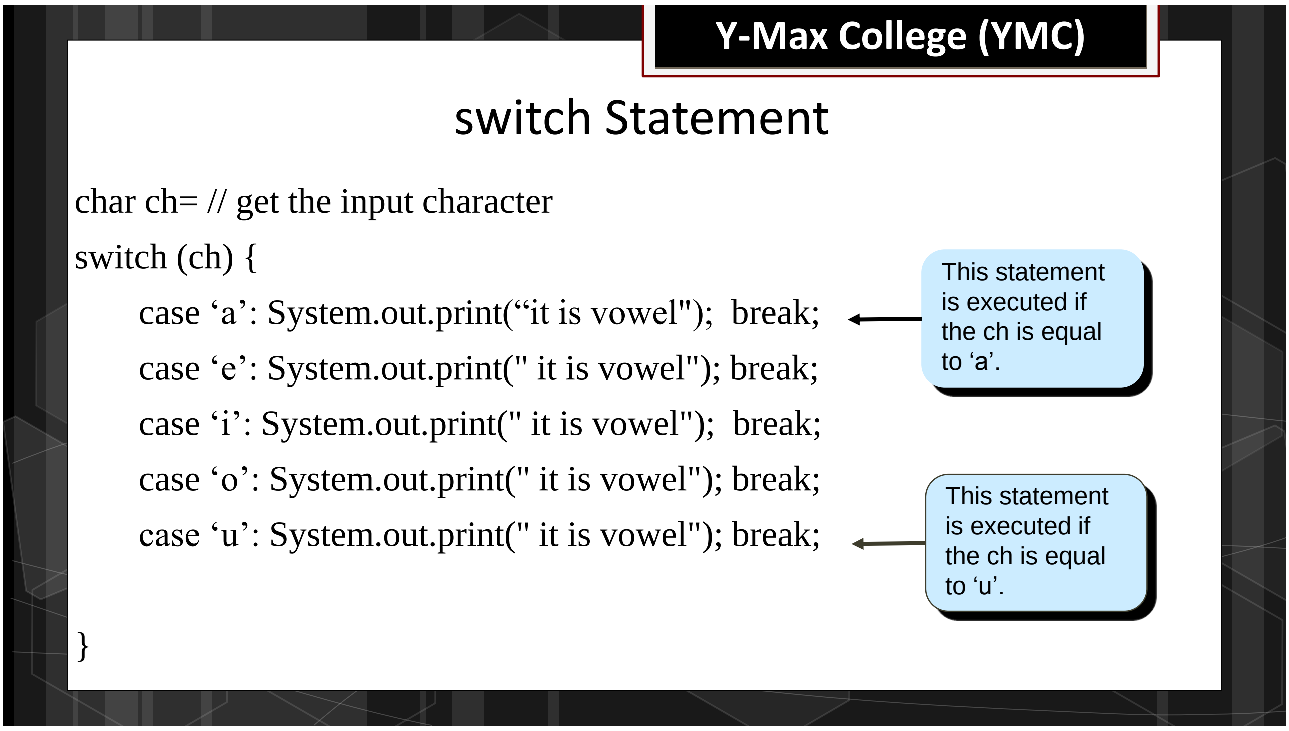


Y-Max College (YMC)

Sematic of Boolean Operators

- Boolean operators and their meanings as shown in following.
- The result of a boolean expression is either **true** or **false**.

P	Q	P && Q	P || Q	!P
false	false	false	false	true
false	true	false	true	true
true	false	false	true	false
true	true	true	true	false



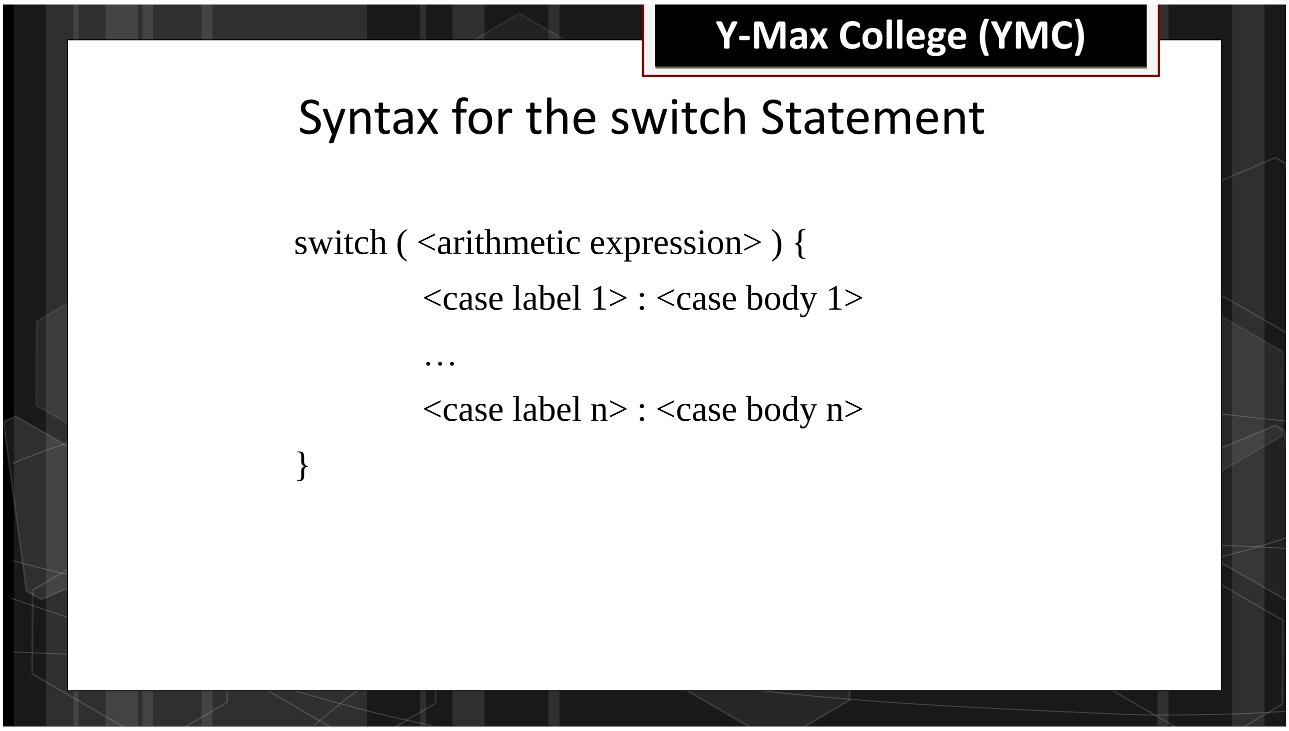
Y-Max College (YMC)

switch Statement

```
char ch= // get the input character
switch (ch) {
      case ‘a’: System.out.print(“it is vowel");  break;
      case ‘e’: System.out.print(" it is vowel"); break;
      case ‘i’: System.out.print(" it is vowel");  break;
      case ‘o’: System.out.print(" it is vowel"); break;
      case ‘u’: System.out.print(" it is vowel"); break;

}
```

This statement is executed if the ch is equal to ‘a’.

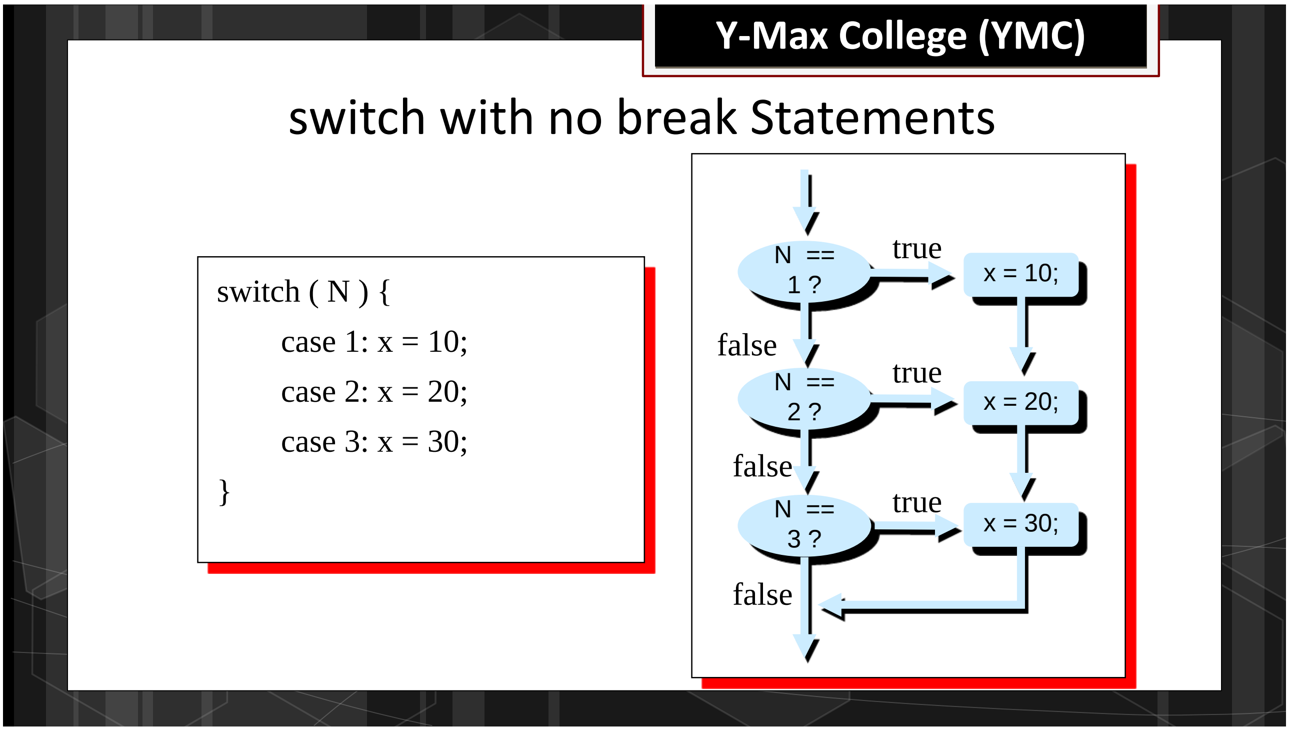
This statement is executed if the ch is equal to ‘u’.



Y-Max College (YMC)

Syntax for the switch Statement

```
switch ( <arithmetic expression> ) {
        <case label 1> : <case body 1>
        …
        <case label n> : <case body n>
}
```

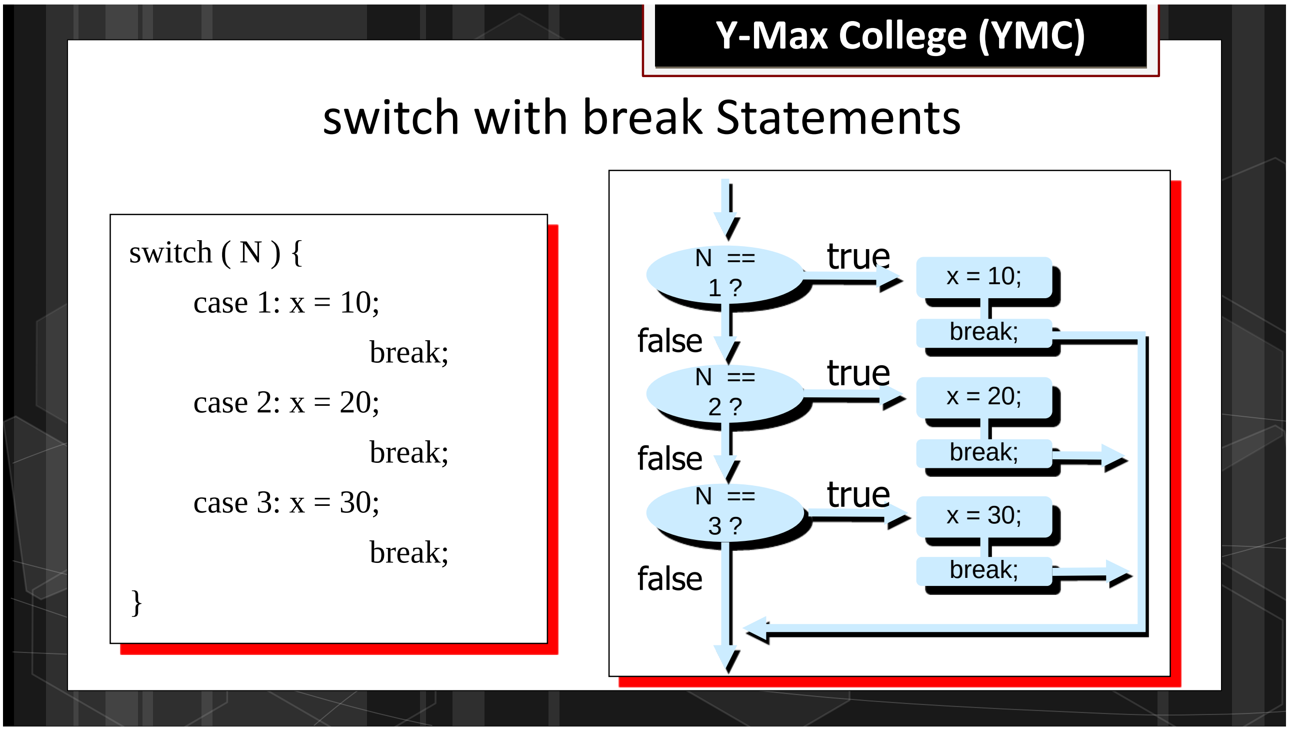


Y-Max College (YMC)

switch with no break Statements

```
switch ( N ) {
    case 1: x = 10;
    case 2: x = 20;
    case 3: x = 30;
}
```

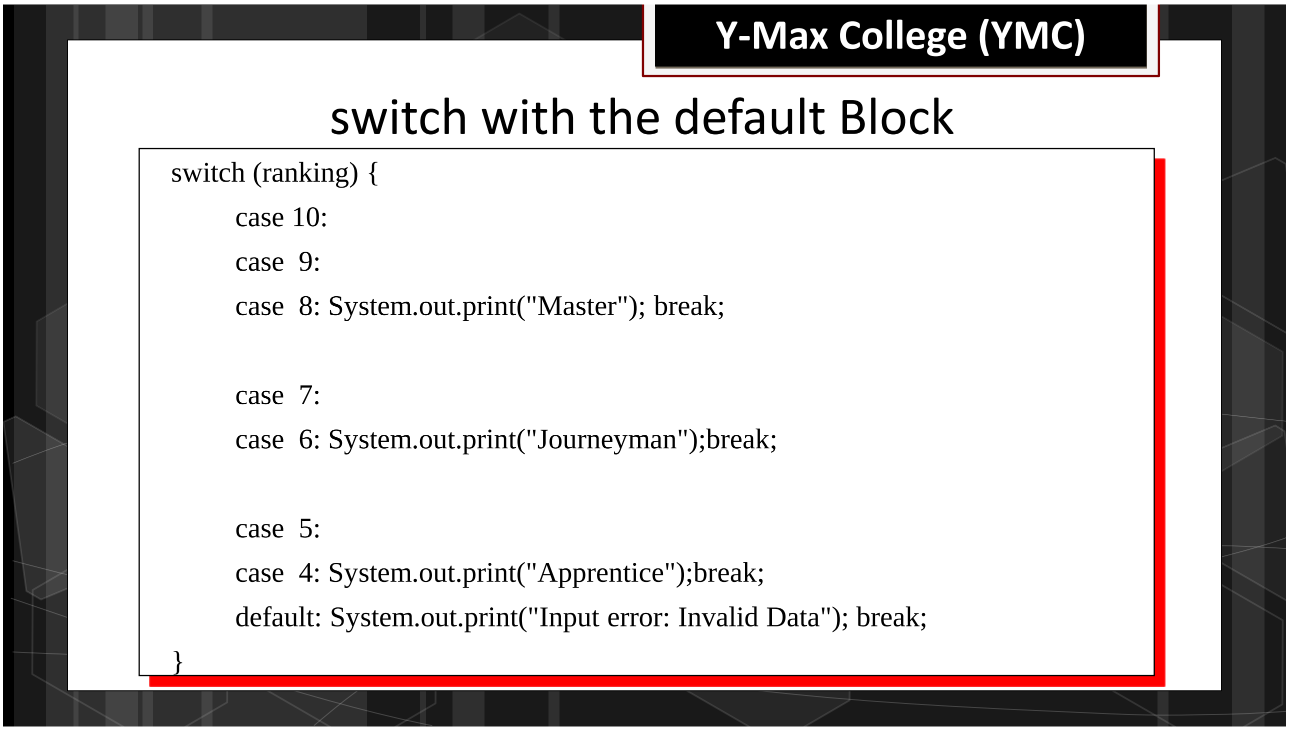
N == 1 ?
true
x = 10;
false
N == 2 ?
true
x = 20;
false
N == 3 ?
true
x = 30;
false



Y-Max College (YMC)

switch with break Statements

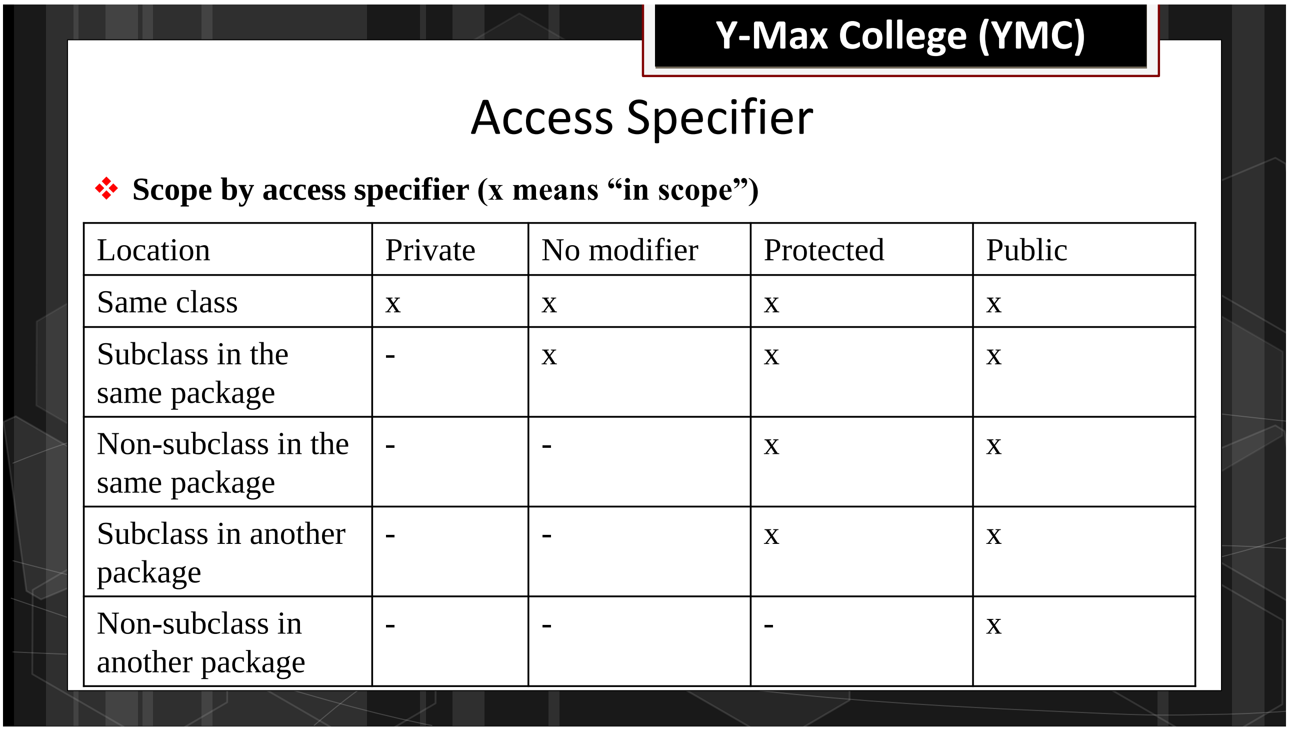
```
switch ( N ) {
        case 1: x = 10;
                        break;
        case 2: x = 20;
                        break;
        case 3: x = 30;
                        break;
}
```



Y-Max College (YMC)

switch with the default Block

```
switch (ranking) {
        case 10:
        case  9:
        case  8: System.out.print("Master"); break;

        case  7:
        case  6: System.out.print("Journeyman");break;

        case  5:
        case  4: System.out.print("Apprentice");break;
        default: System.out.print("Input error: Invalid Data"); break;
}
```

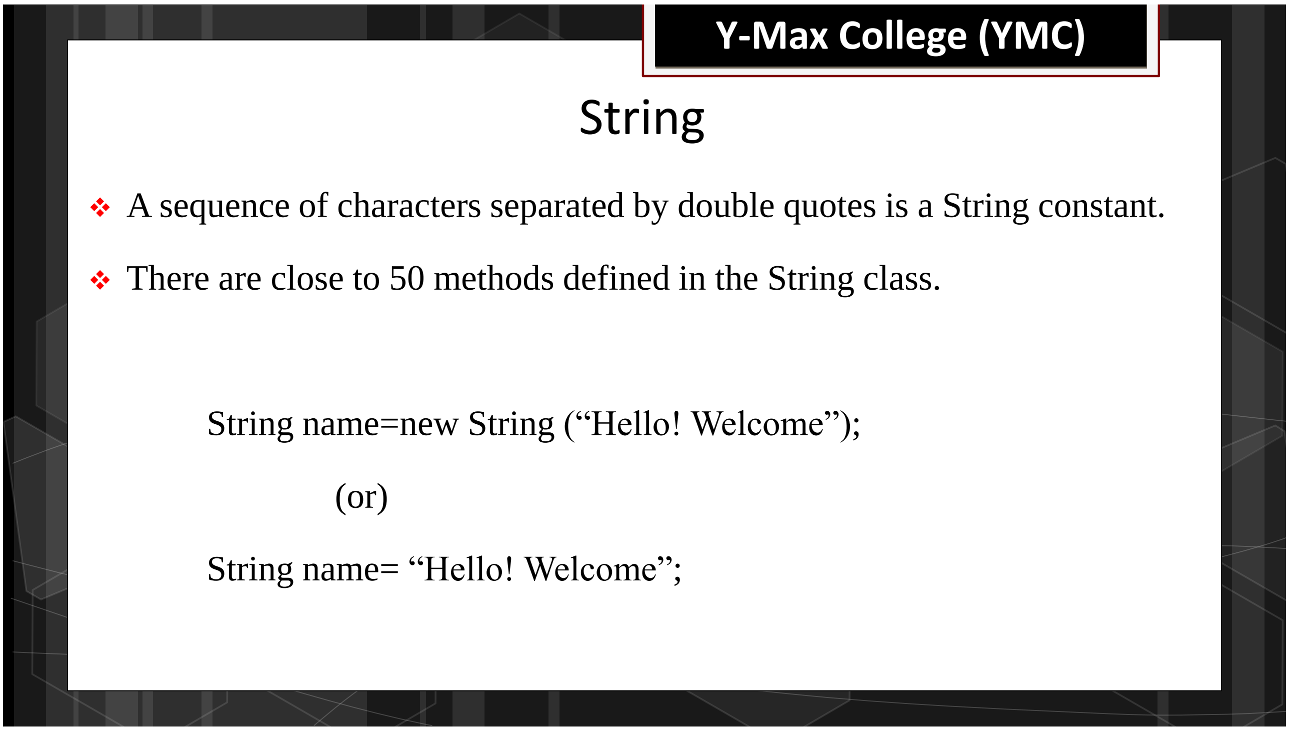


Y-Max College (YMC)

Access Specifier

❖ **Scope by access specifier (x means “in scope”)**

Location	Private	No modifier	Protected	Public
Same class	x	x	x	x
Subclass in the same package	-	x	x	x
Non-subclass in the same package	-	-	x	x
Subclass in another package	-	-	x	x
Non-subclass in another package	-	-	-	x

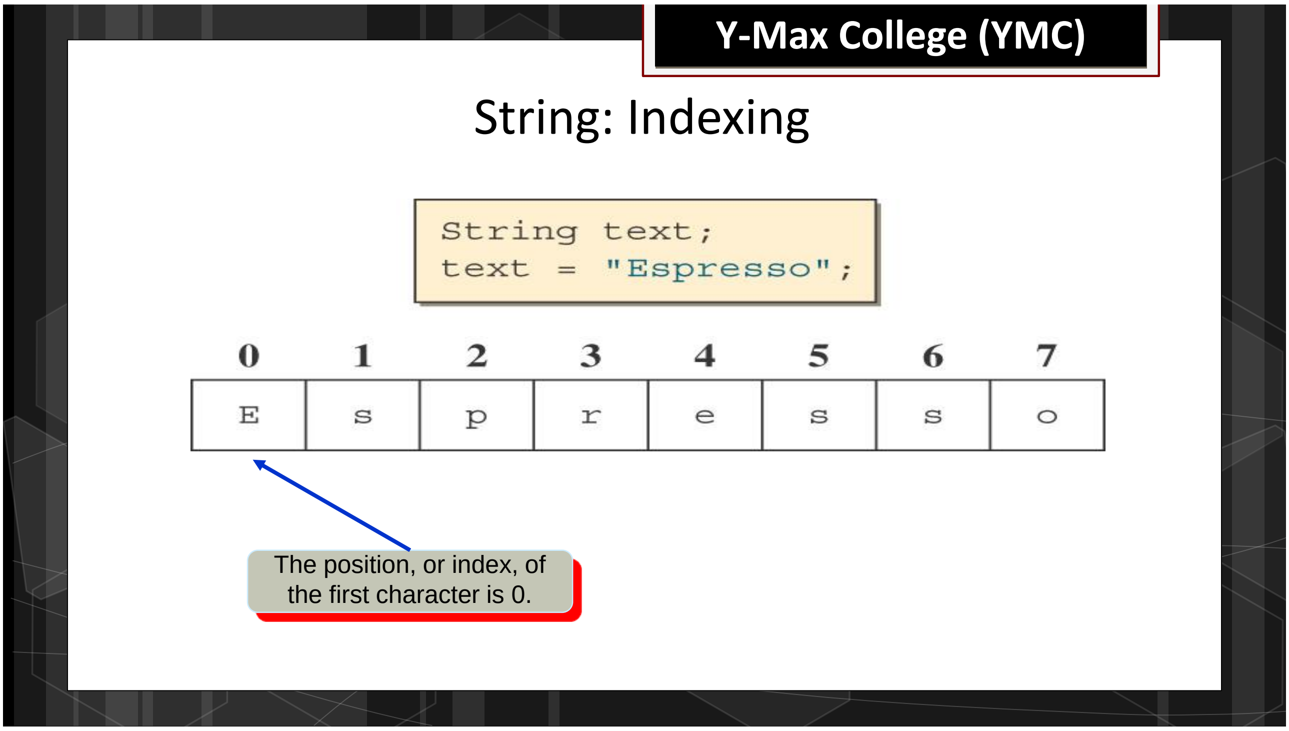


Y-Max College (YMC)

String

- A sequence of characters separated by double quotes is a String constant.
- There are close to 50 methods defined in the String class.

```
String name=new String (“Hello! Welcome”);
            (or)
String name= “Hello! Welcome”;
```



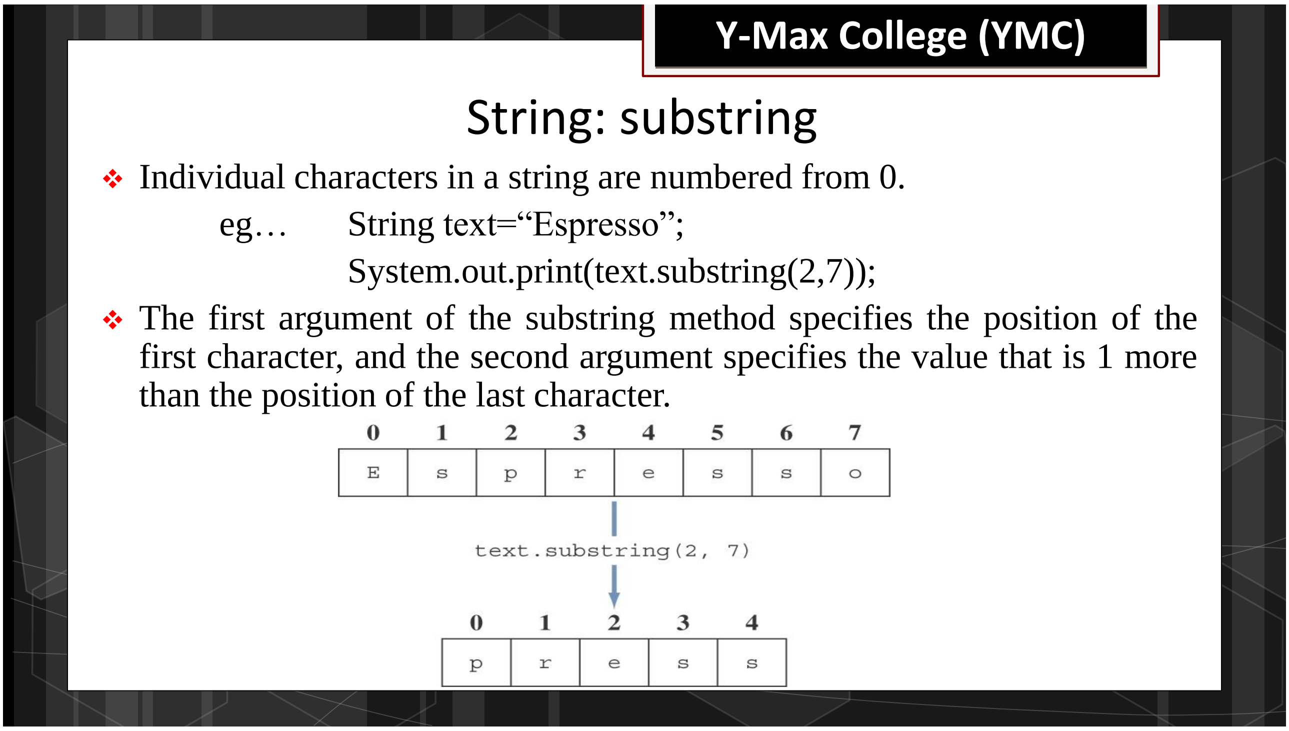
Y-Max College (YMC)

String: Indexing

```
String text;
text = "Espresso";
```

0	1	2	3	4	5	6	7
E	s	p	r	e	s	s	o

The position, or index, of the first character is 0.



Y-Max College (YMC)

String: substring

- Individual characters in a string are numbered from 0.

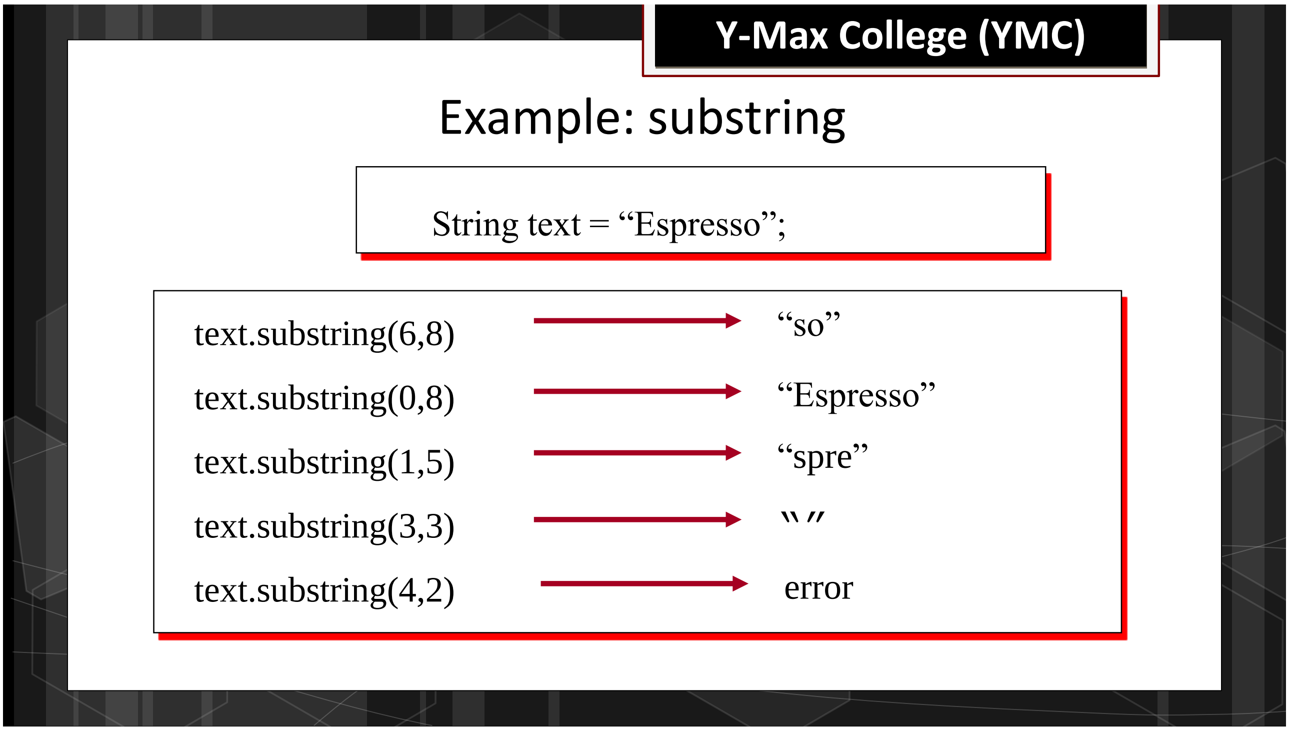
  eg… String text="Espresso";
  System.out.print(text.substring(2,7));

- The first argument of the substring method specifies the position of the first character, and the second argument specifies the value that is 1 more than the position of the last character.

0	1	2	3	4	5	6	7
E	s	p	r	e	s	s	o

```
text.substring(2, 7)
```

0	1	2	3	4
p	r	e	s	s



Y-Max College (YMC)

Example: substring

```
String text = “Espresso”;
```

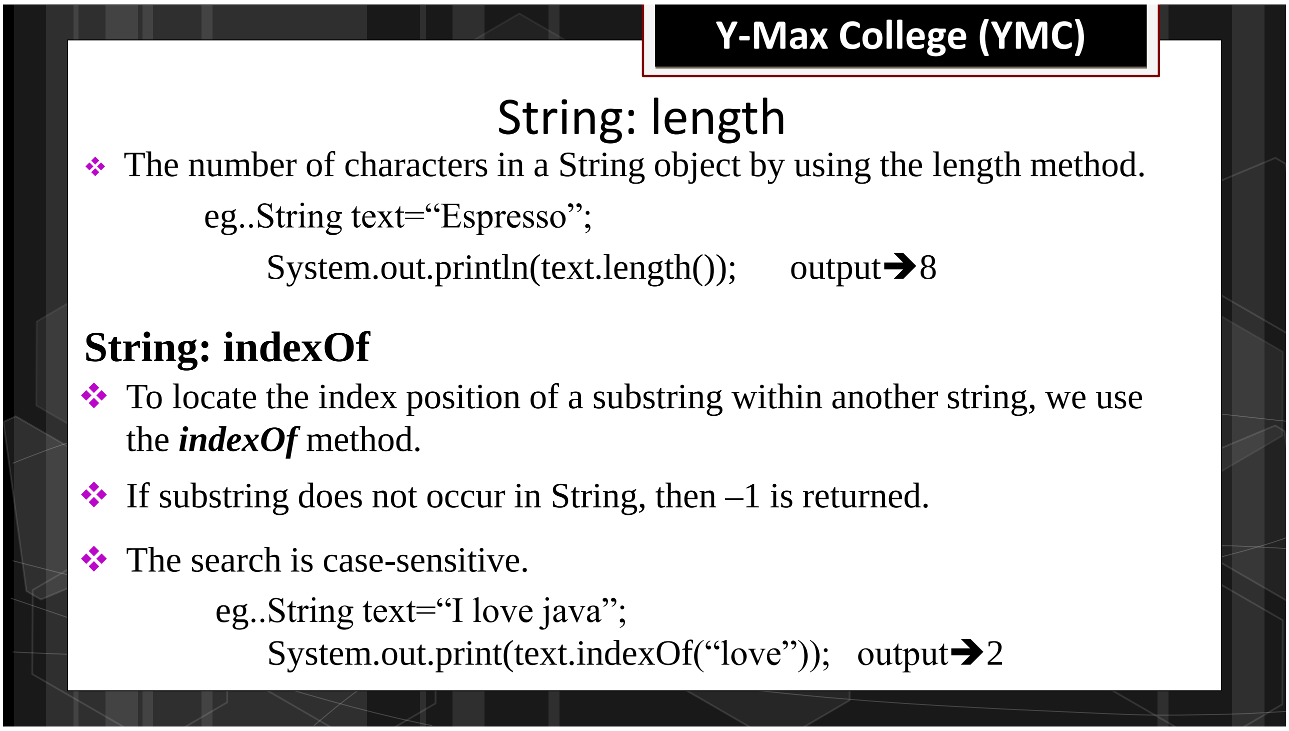
text.substring(6,8) → “so”

text.substring(0,8) → “Espresso”

text.substring(1,5) → “spre”

text.substring(3,3) → " "

text.substring(4,2) → error



Y-Max College (YMC)

String: length

- The number of characters in a String object by using the length method.

  eg..String text=“Espresso”;

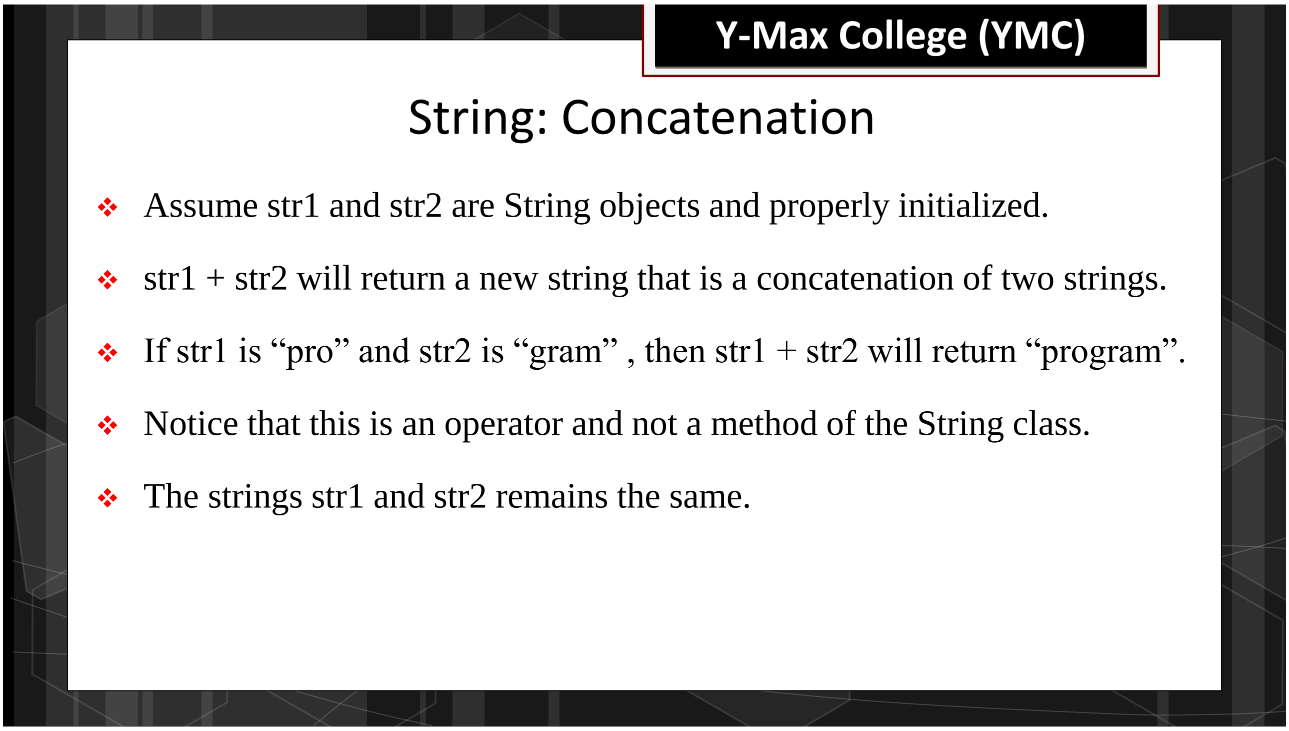
  System.out.println(text.length()); output➔8

String: indexOf

- To locate the index position of a substring within another string, we use the ***indexOf*** method.
- If substring does not occur in String, then –1 is returned.
- The search is case-sensitive.

  eg..String text=“I love java”;

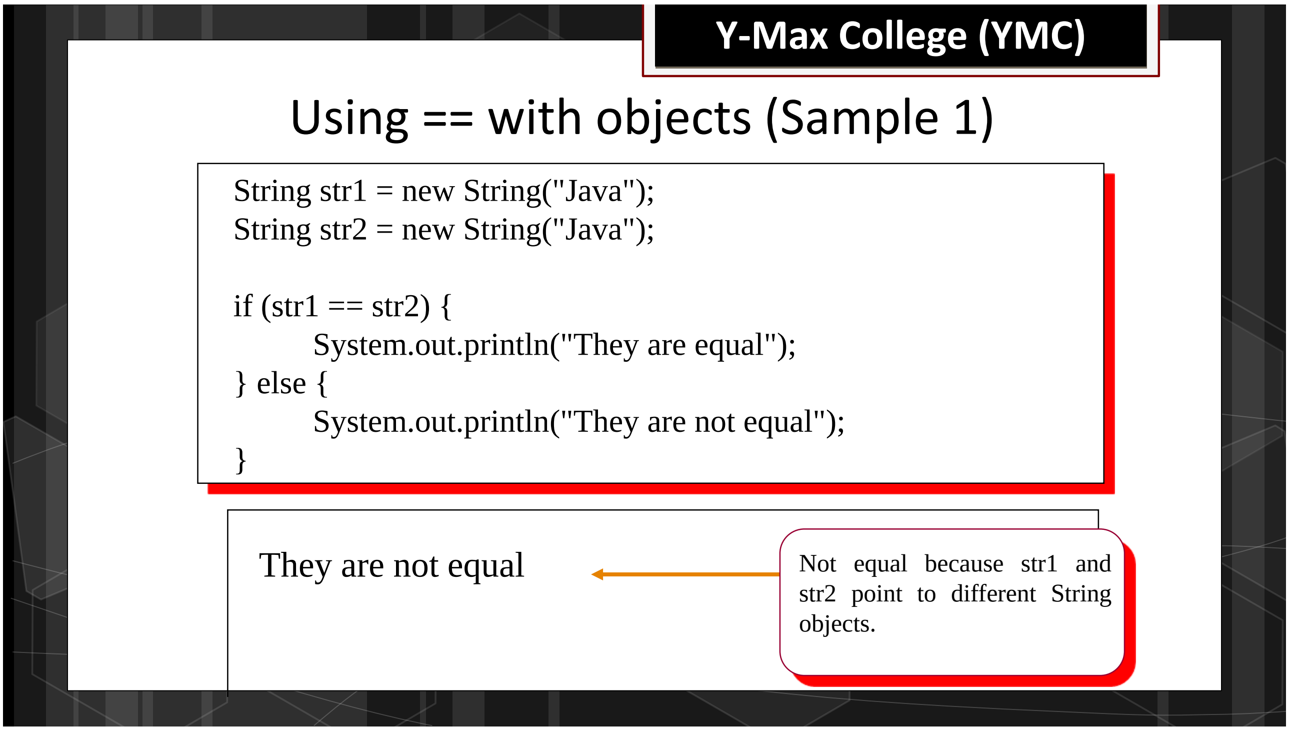
  System.out.print(text.indexOf(“love”)); output➔2



Y-Max College (YMC)

String: Concatenation

- Assume str1 and str2 are String objects and properly initialized.
- str1 + str2 will return a new string that is a concatenation of two strings.
- If str1 is “pro” and str2 is “gram” , then str1 + str2 will return “program”.
- Notice that this is an operator and not a method of the String class.
- The strings str1 and str2 remains the same.



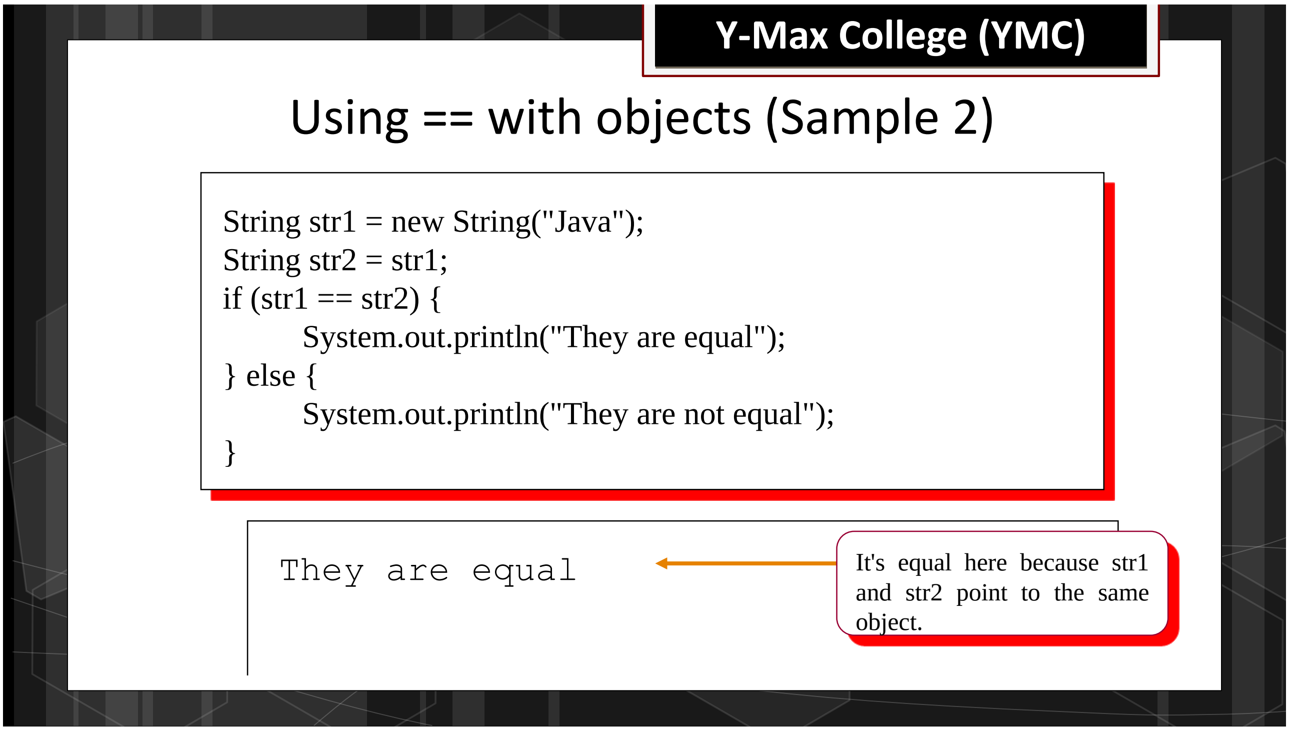
Y-Max College (YMC)

Using == with objects (Sample 1)

```
String str1 = new String("Java");
String str2 = new String("Java");

if (str1 == str2) {
        System.out.println("They are equal");
} else {
        System.out.println("They are not equal");
}
```

```
They are not equal
```

Not equal because str1 and str2 point to different String objects.



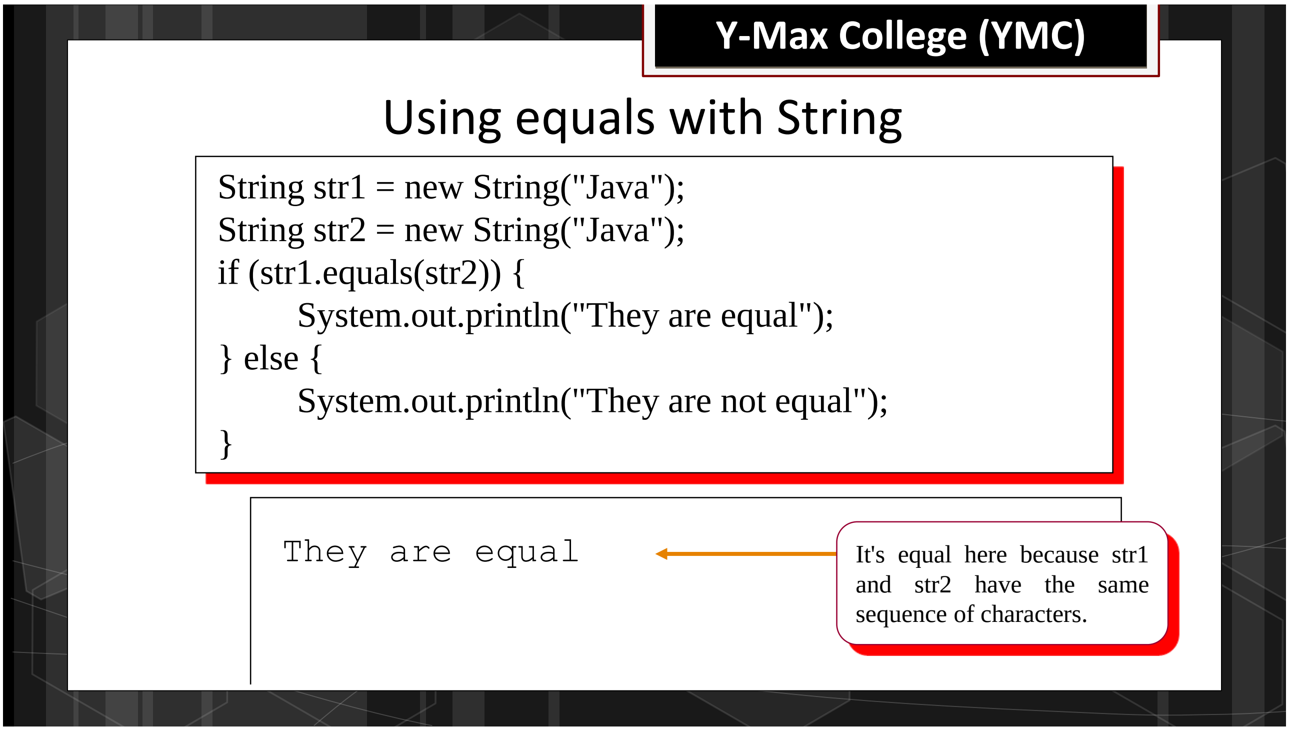
Y-Max College (YMC)

Using == with objects (Sample 2)

```
String str1 = new String("Java");
String str2 = str1;
if (str1 == str2) {
        System.out.println("They are equal");
} else {
        System.out.println("They are not equal");
}
```

```
They are equal
```

It's equal here because str1 and str2 point to the same object.



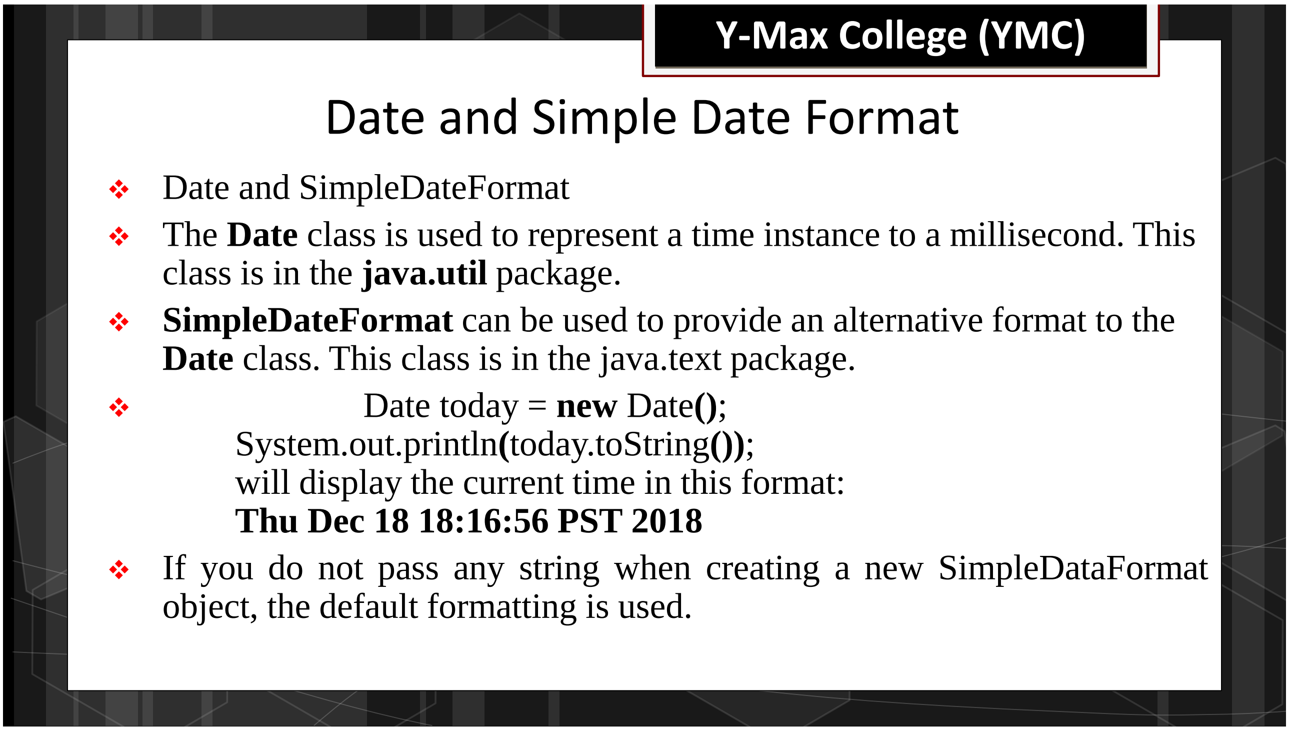
Y-Max College (YMC)

Using equals with String

```
String str1 = new String("Java");
String str2 = new String("Java");
if (str1.equals(str2)) {
        System.out.println("They are equal");
} else {
        System.out.println("They are not equal");
}
```

```
They are equal
```

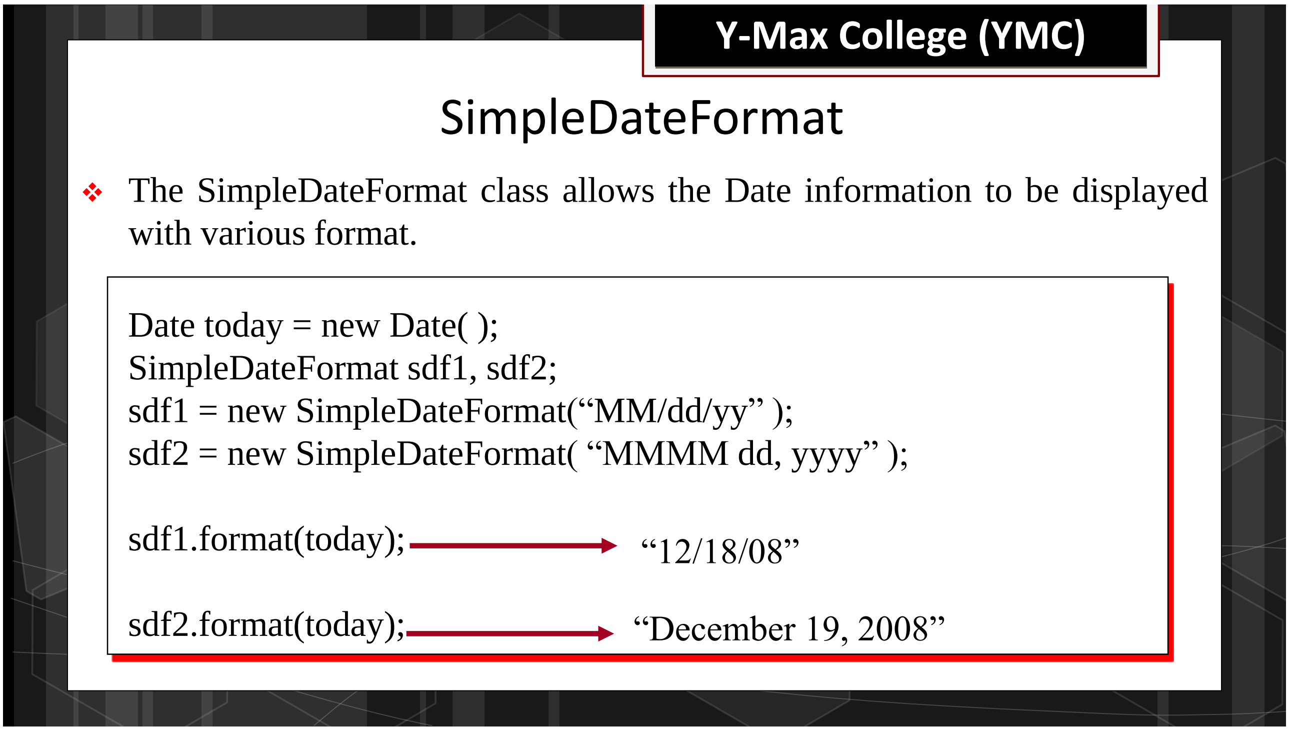
It's equal here because str1 and str2 have the same sequence of characters.



Y-Max College (YMC)

Date and Simple Date Format

- Date and SimpleDateFormat
- The **Date** class is used to represent a time instance to a millisecond. This class is in the **java.util** package.
- **SimpleDateFormat** can be used to provide an alternative format to the **Date** class. This class is in the java.text package.
- Date today = **new** Date**()**;
  System.out.println**(**today.toString**())**;
  will display the current time in this format:
  Thu Dec 18 18:16:56 PST 2018
- If you do not pass any string when creating a new SimpleDataFormat object, the default formatting is used.



Y-Max College (YMC)

SimpleDateFormat

- The SimpleDateFormat class allows the Date information to be displayed with various format.

```
Date today = new Date( );
SimpleDateFormat sdf1, sdf2;
sdf1 = new SimpleDateFormat(“MM/dd/yy” );
sdf2 = new SimpleDateFormat( “MMMM dd, yyyy” );

sdf1.format(today); → “12/18/08”

sdf2.format(today); → “December 19, 2008”
```

Y-Max College (YMC)

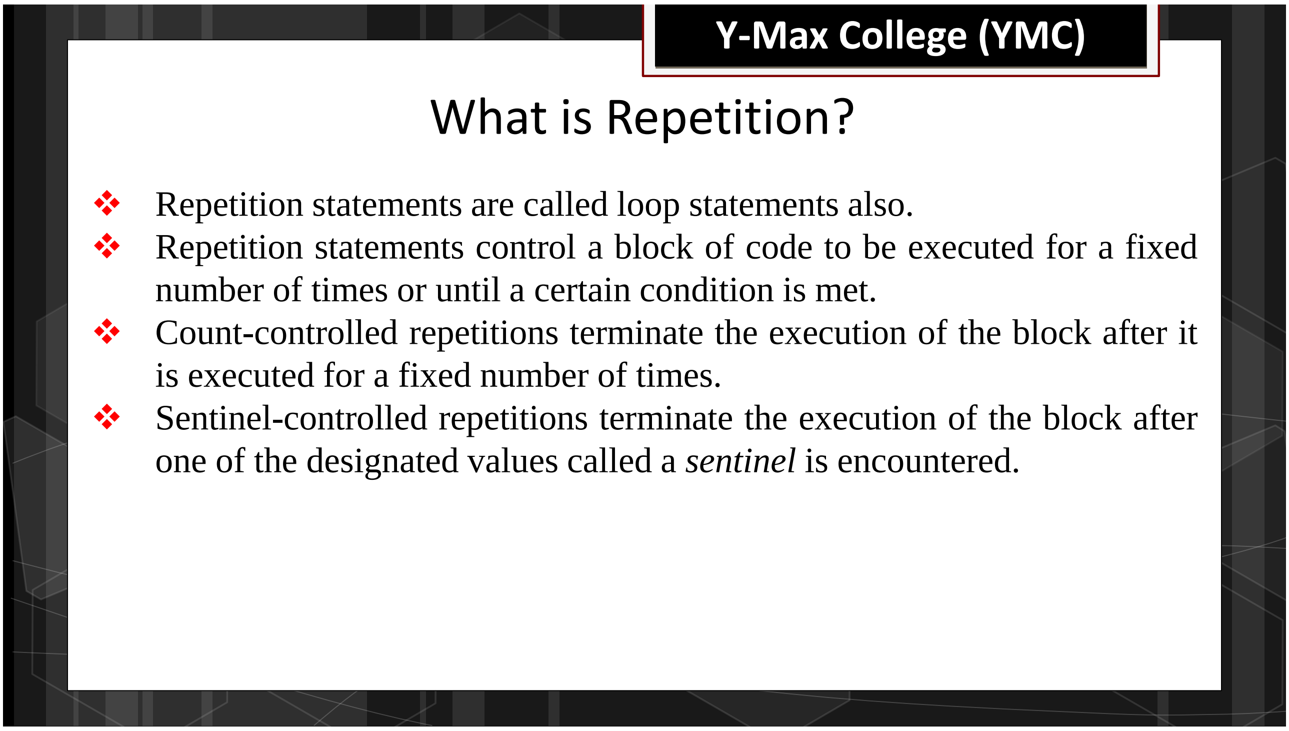
Any Questions….?

Y-Max College (YMC)

Chapter – 3
Repetition

Topics covered

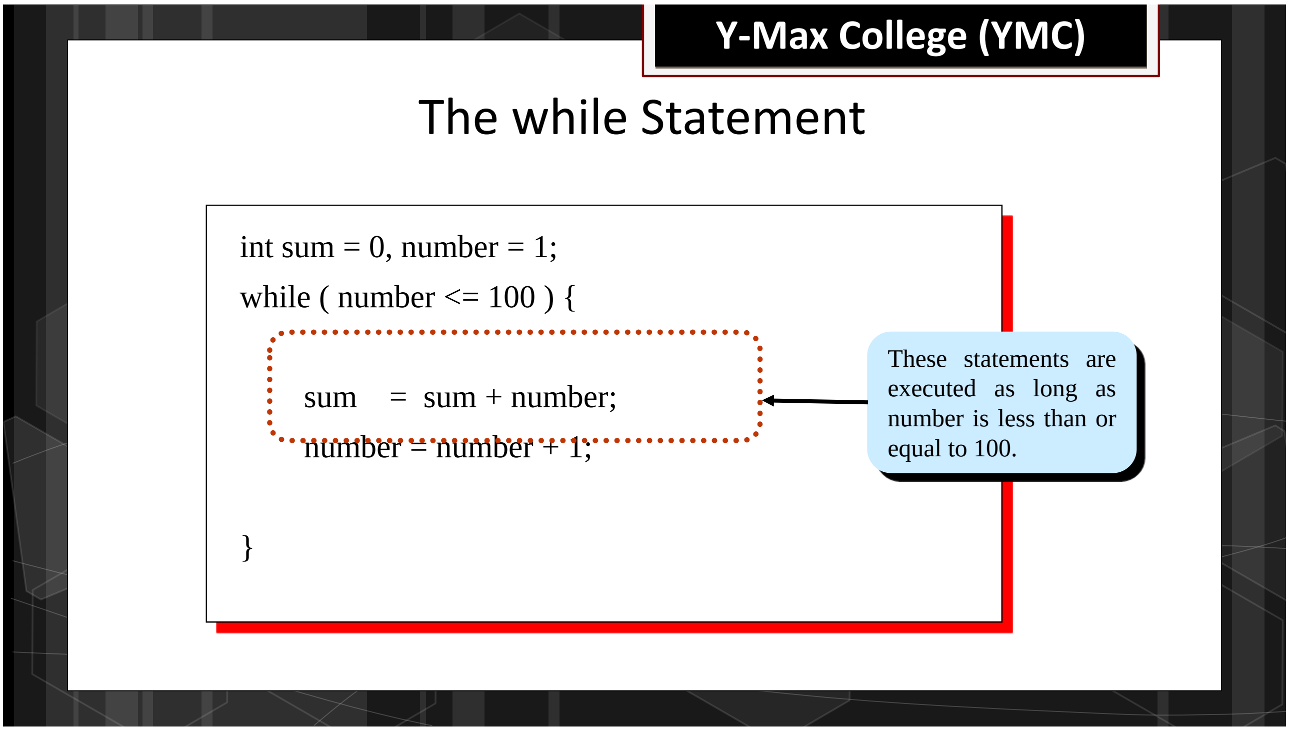
- Implement repetition control in a program using **while** statements.
- Implement repetition control in a program using **do-while** statements.
- Implement repetition control in a program using **for** statements.
- Nest a loop repetition statement inside another repetition statement.
- Java Arrays
- Using function



Y-Max College (YMC)

What is Repetition?

- Repetition statements are called loop statements also.
- Repetition statements control a block of code to be executed for a fixed number of times or until a certain condition is met.
- Count-controlled repetitions terminate the execution of the block after it is executed for a fixed number of times.
- Sentinel-controlled repetitions terminate the execution of the block after one of the designated values called a *sentinel* is encountered.

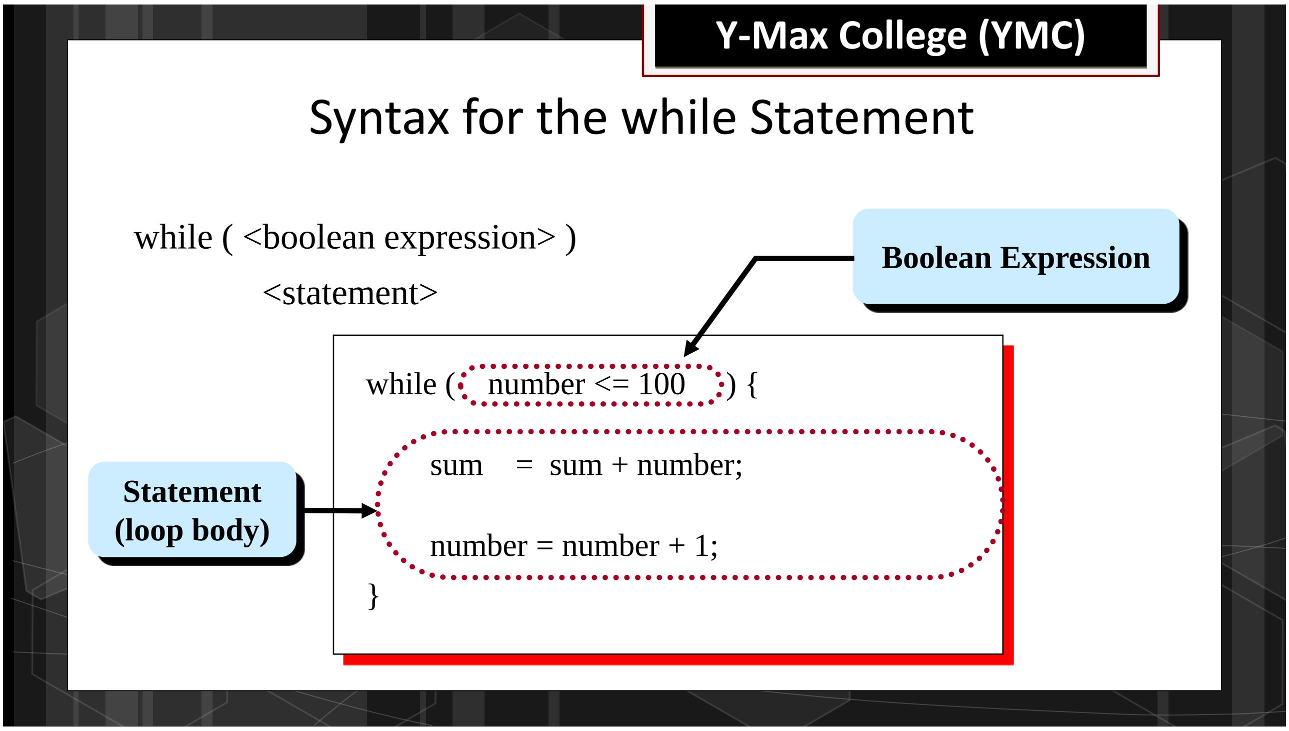


Y-Max College (YMC)

The while Statement

```
int sum = 0, number = 1;
while ( number <= 100 ) {

    sum    =  sum + number;
    number = number + 1;

}
```

These statements are executed as long as number is less than or equal to 100.



Y-Max College (YMC)

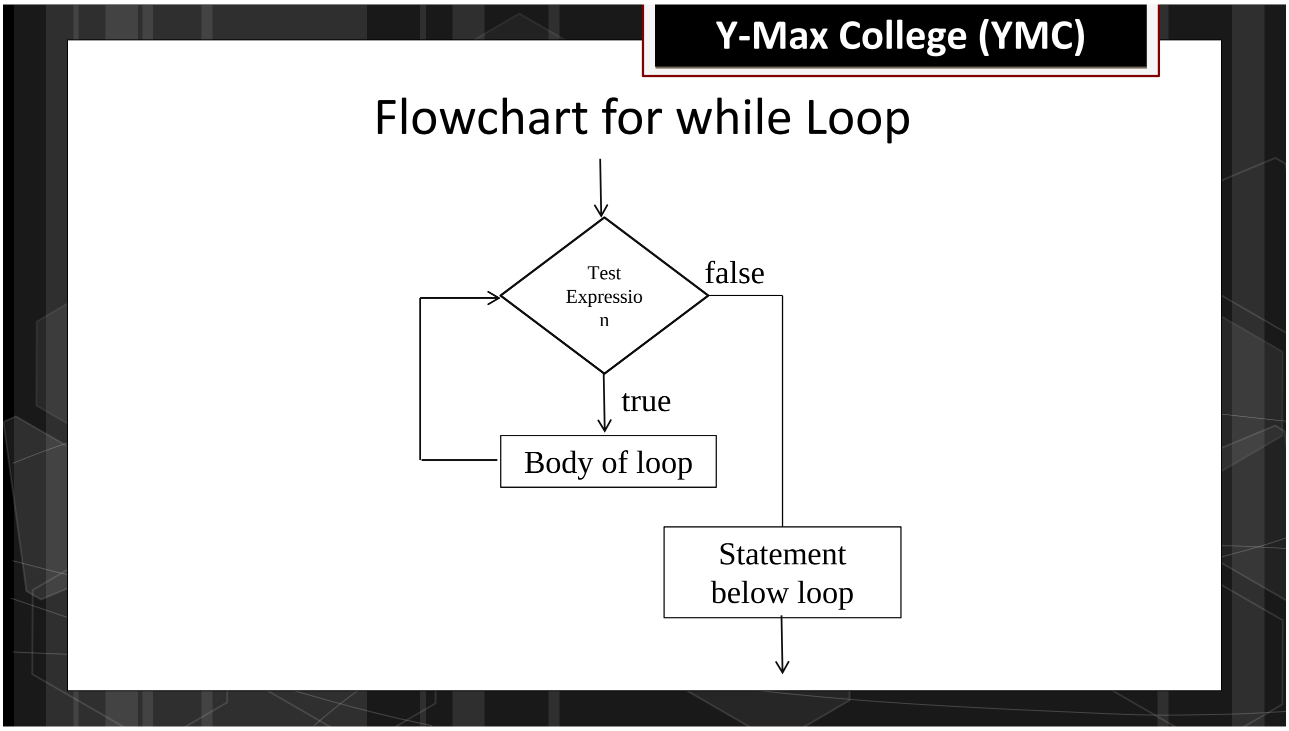
Syntax for the while Statement

```
while ( <boolean expression> )
        <statement>
```

Boolean Expression

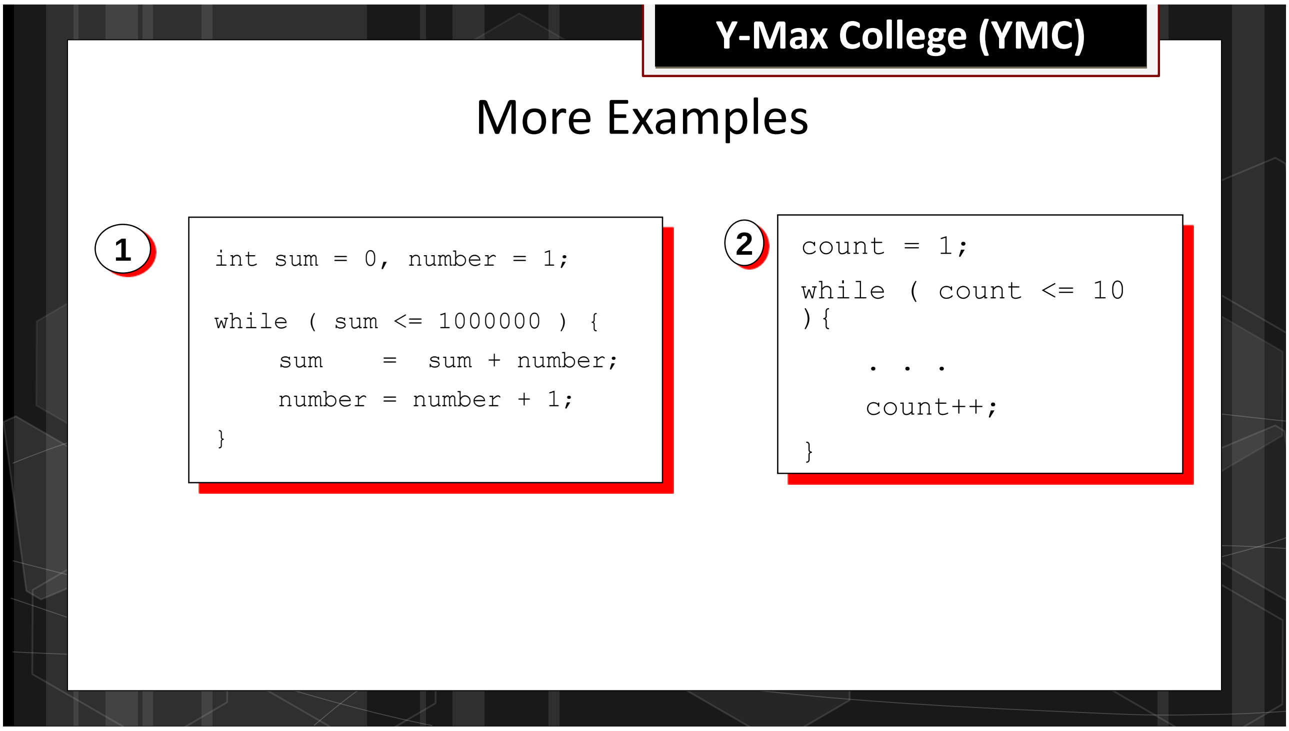
Statement (loop body)

```
while ( number <= 100 ) {
      sum    =  sum + number;

      number = number + 1;
}
```



Y-Max College (YMC)

Flowchart for while Loop



Y-Max College (YMC)

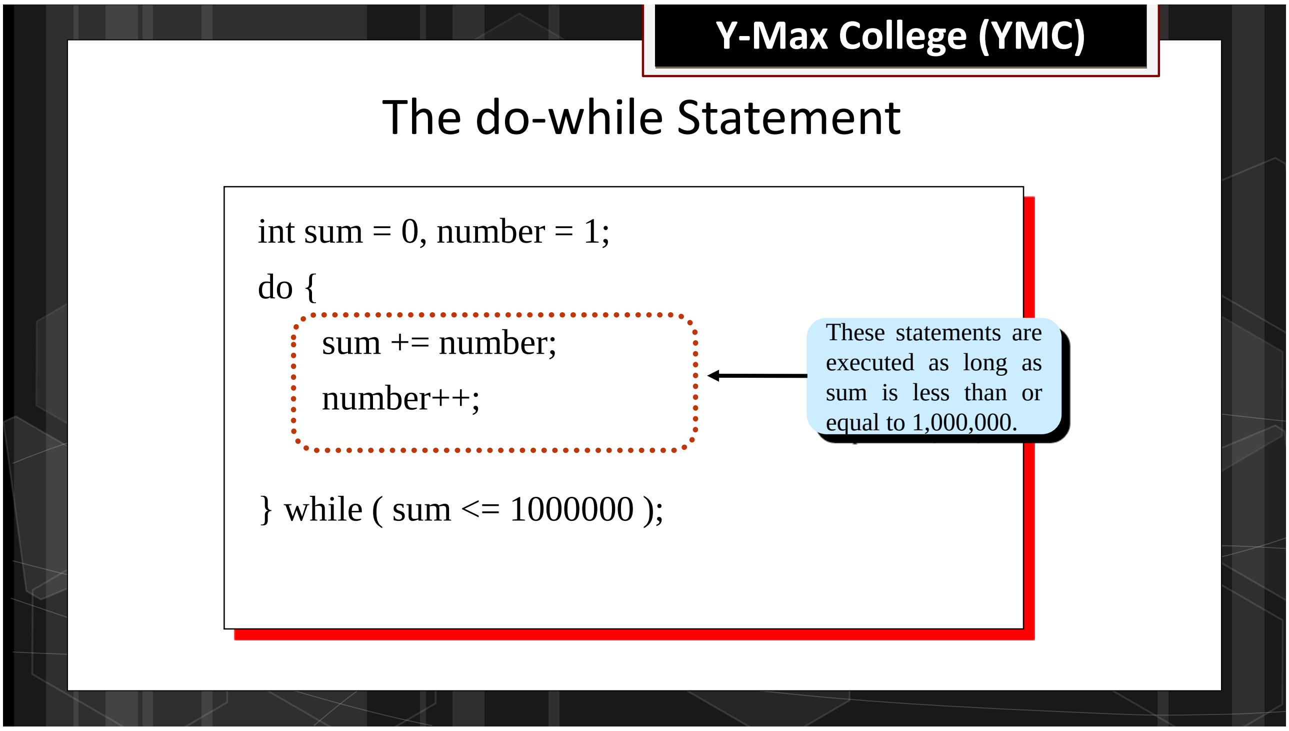
More Examples

1

```
int sum = 0, number = 1;

while ( sum <= 1000000 ) {
      sum    =  sum + number;
      number = number + 1;
}
```

2

```
count = 1;
while ( count <= 10
){
     . . .
     count++;
}
```

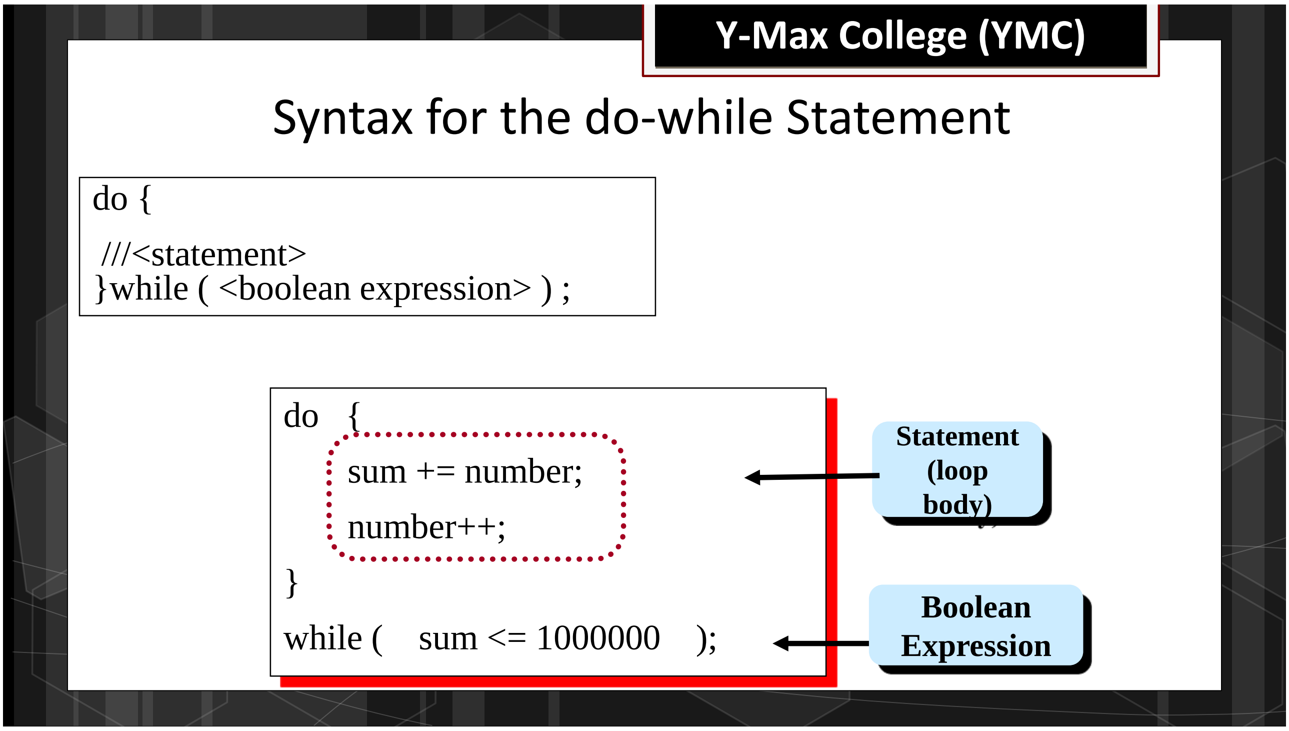


Y-Max College (YMC)

The do-while Statement

```
int sum = 0, number = 1;
do {
      sum += number;
      number++;

} while ( sum <= 1000000 );
```

These statements are executed as long as sum is less than or equal to 1,000,000.



Y-Max College (YMC)

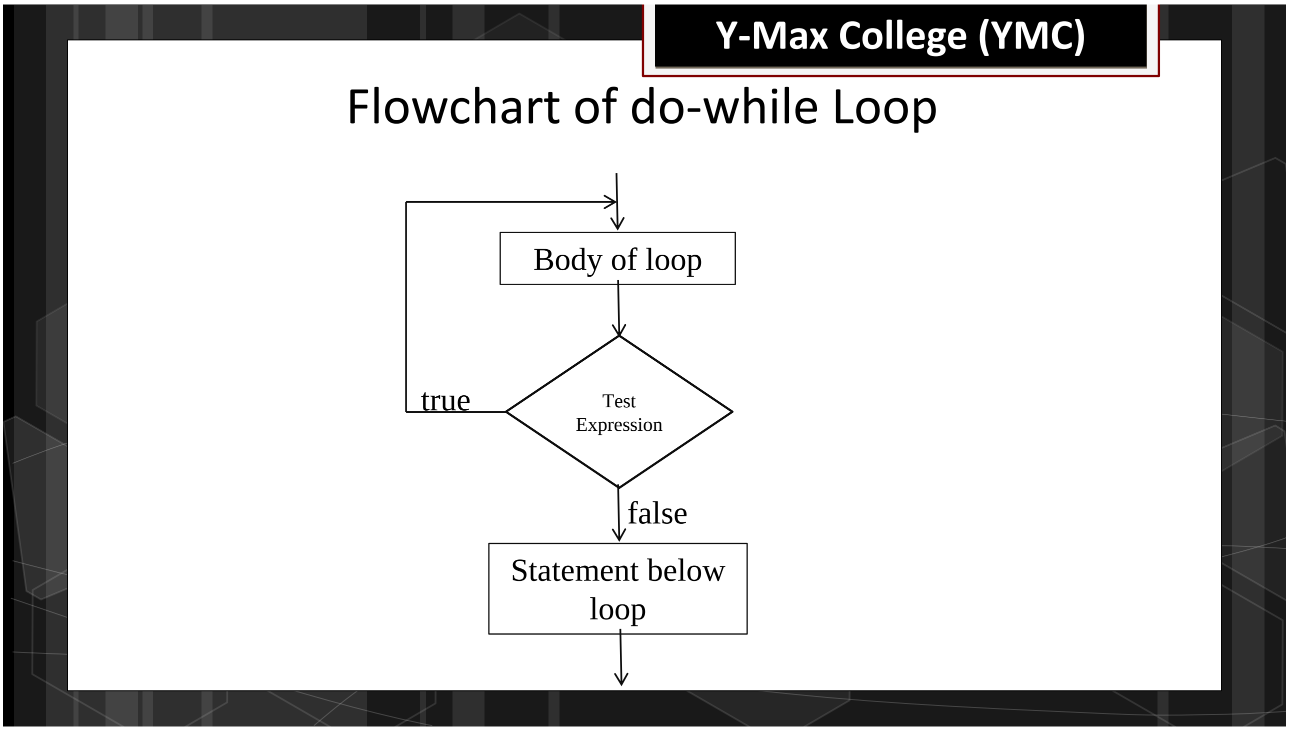
Syntax for the do-while Statement

```
do {
 ///<statement>
}while ( <boolean expression> ) ;
```

```
do   {
      sum += number;
      number++;
}
while (    sum <= 1000000    );
```

Statement (loop body)

Boolean Expression



Y-Max College (YMC)

Flowchart of do-while Loop

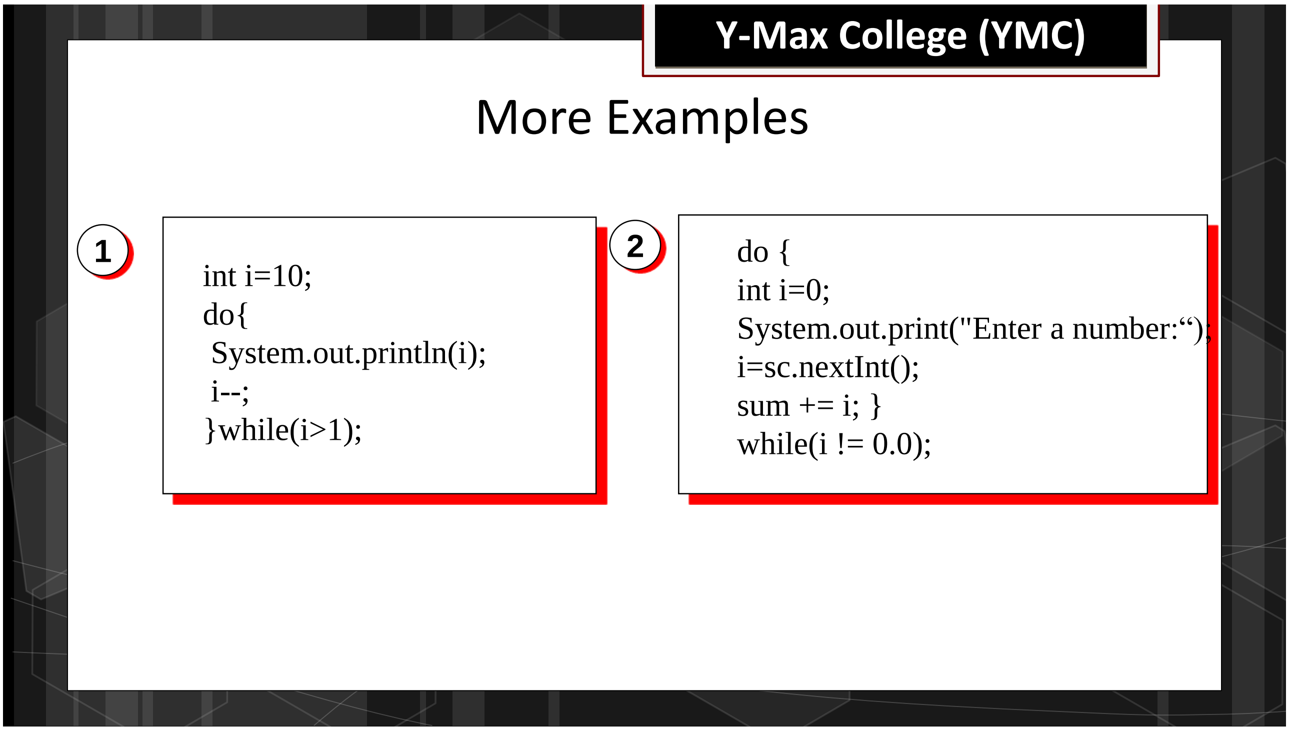
Body of loop

Test Expression

true

false

Statement below loop



Y-Max College (YMC)

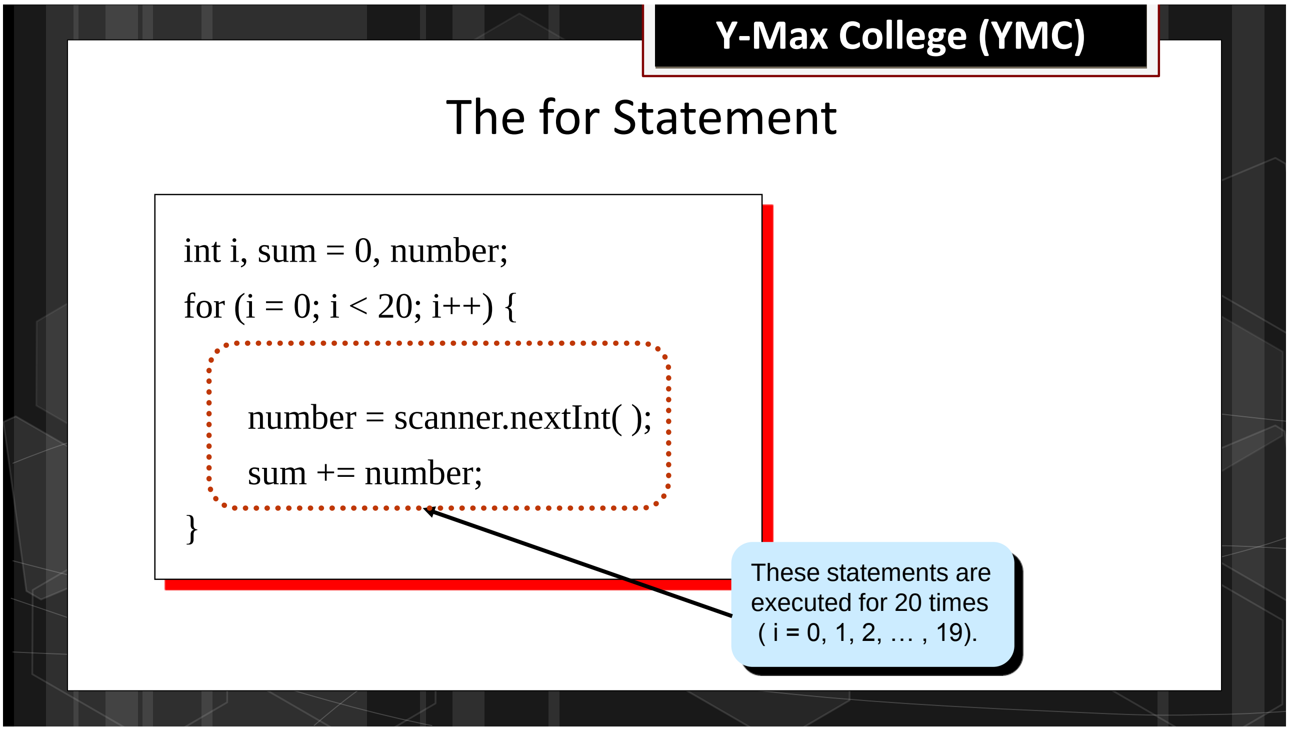
More Examples

1

```
int i=10;
do{
 System.out.println(i);
 i--;
}while(i>1);
```

2

```
do {
int i=0;
System.out.print("Enter a number:“);
i=sc.nextInt();
sum += i; }
while(i != 0.0);
```

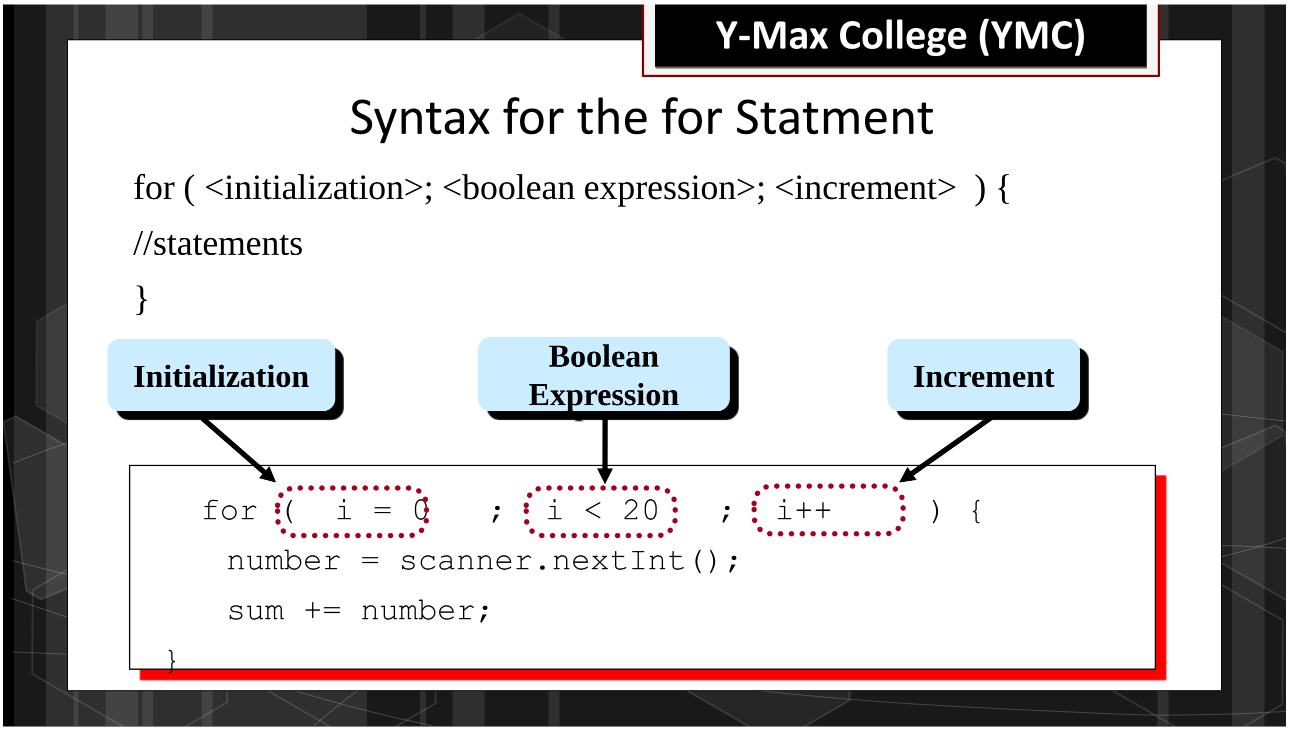


Y-Max College (YMC)

The for Statement

```
int i, sum = 0, number;
for (i = 0; i < 20; i++) {

    number = scanner.nextInt( );
    sum += number;
}
```

These statements are executed for 20 times (i = 0, 1, 2, … , 19).



Y-Max College (YMC)

Syntax for the for Statment

for (<initialization>; <boolean expression>; <increment>) {

//statements

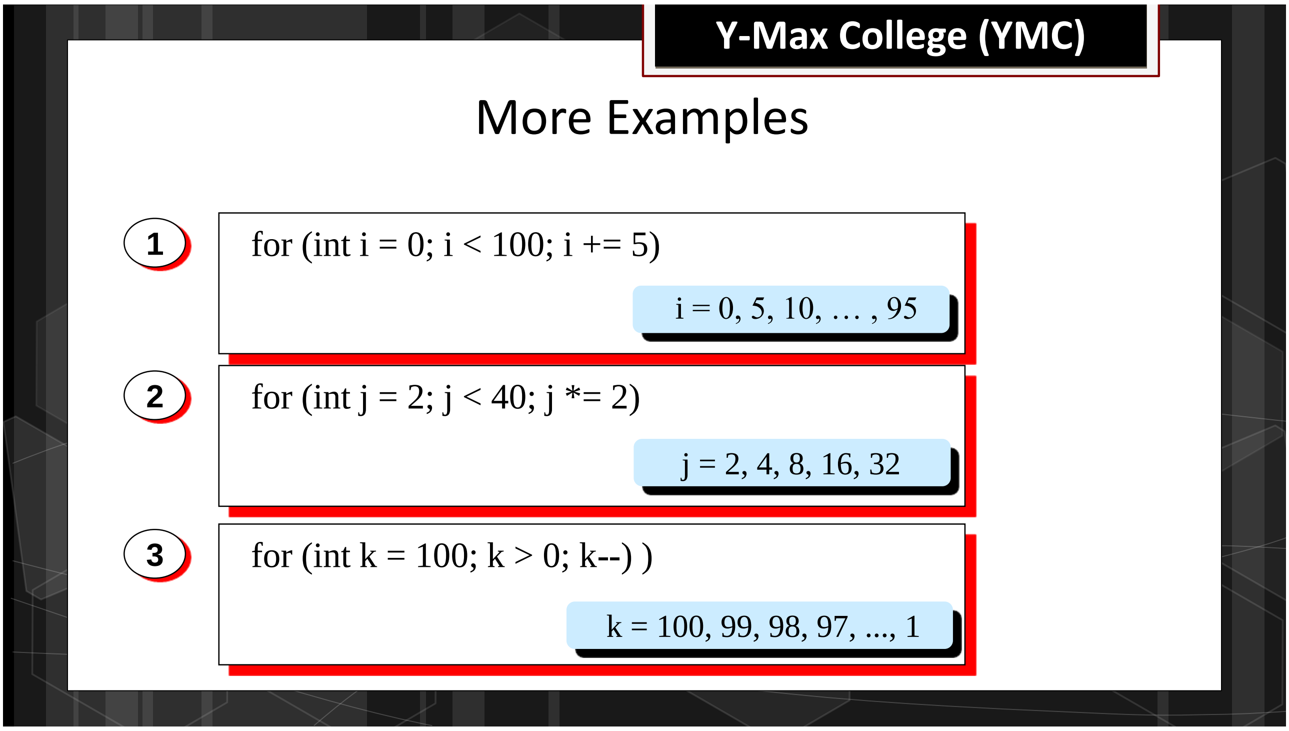
}

Initialization

Boolean Expression

Increment

```
    for ( i = 0   ; i < 20 ; i++   ) {
      number = scanner.nextInt();
      sum += number;
   }
```



Y-Max College (YMC)

More Examples

1

for (int i = 0; i < 100; i += 5)

i = 0, 5, 10, … , 95

2

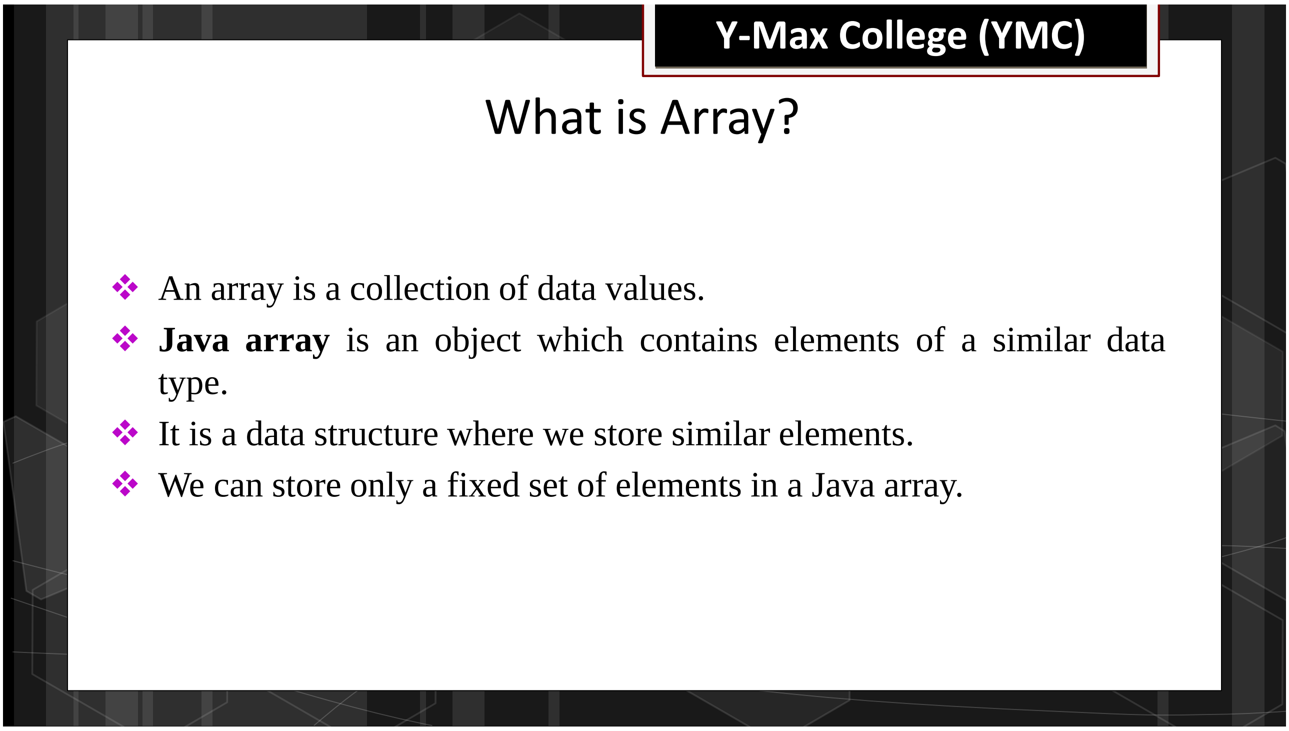
for (int j = 2; j < 40; j *= 2)

j = 2, 4, 8, 16, 32

3

for (int k = 100; k > 0; k--))

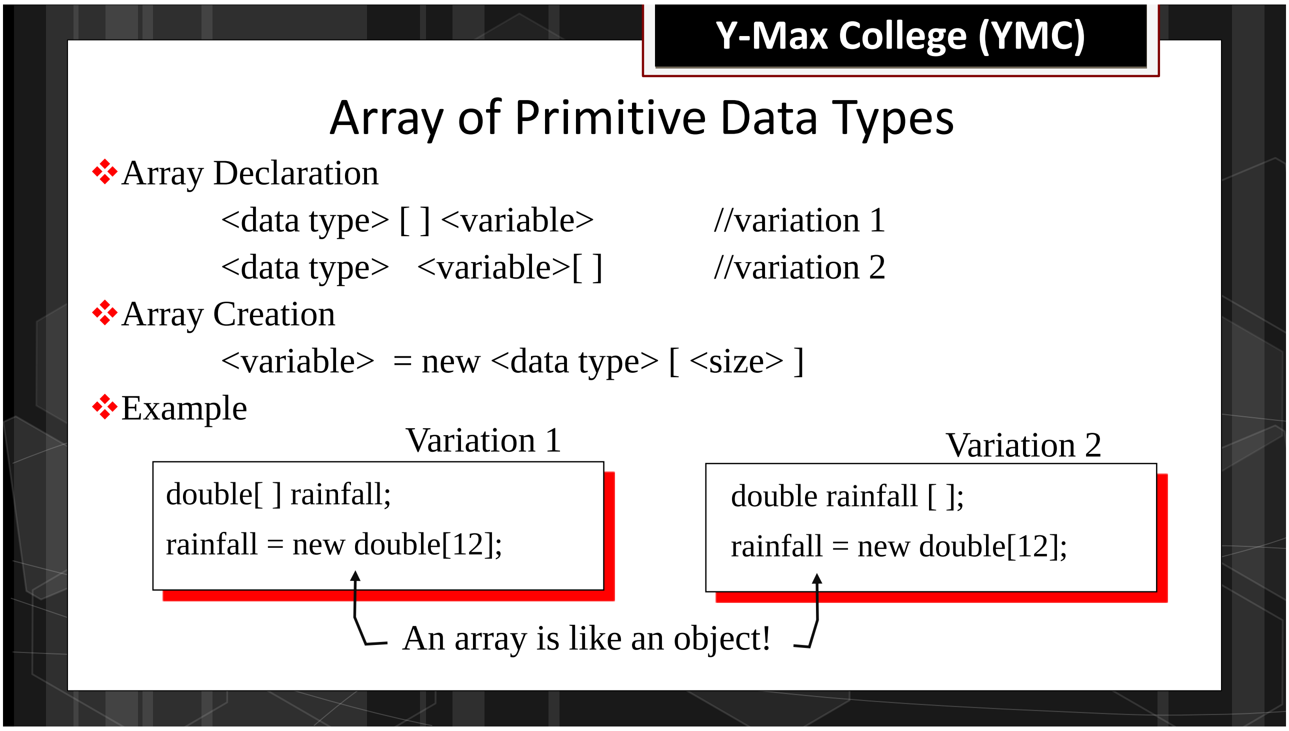
k = 100, 99, 98, 97, ..., 1



Y-Max College (YMC)

What is Array?

- An array is a collection of data values.
- **Java array** is an object which contains elements of a similar data type.
- It is a data structure where we store similar elements.
- We can store only a fixed set of elements in a Java array.



Y-Max College (YMC)

Array of Primitive Data Types

- Array Declaration

```
<data type> [ ] <variable>          //variation 1
<data type>   <variable>[ ]         //variation 2
```

- Array Creation

```
<variable>  = new <data type> [ <size> ]
```

- Example

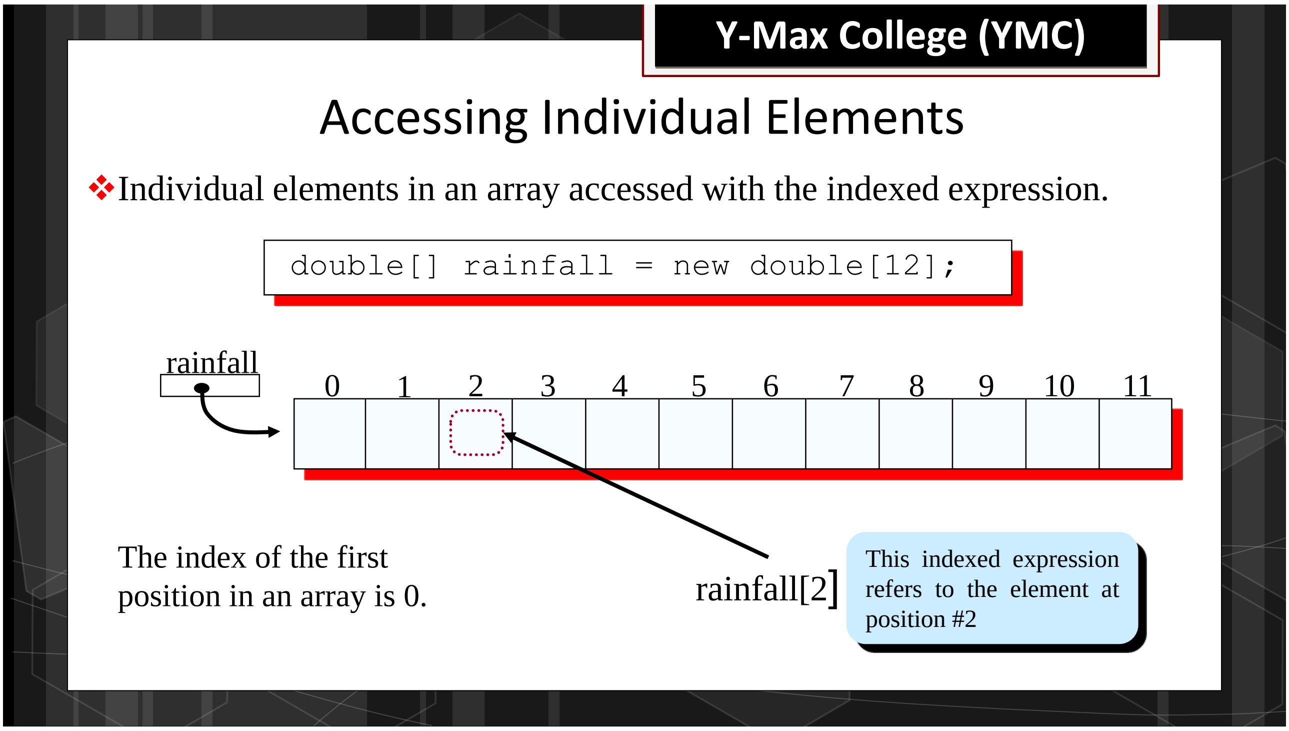
Variation 1

```
double[ ] rainfall;
rainfall = new double[12];
```

Variation 2

```
double rainfall [ ];
rainfall = new double[12];
```

An array is like an object!



Y-Max College (YMC)

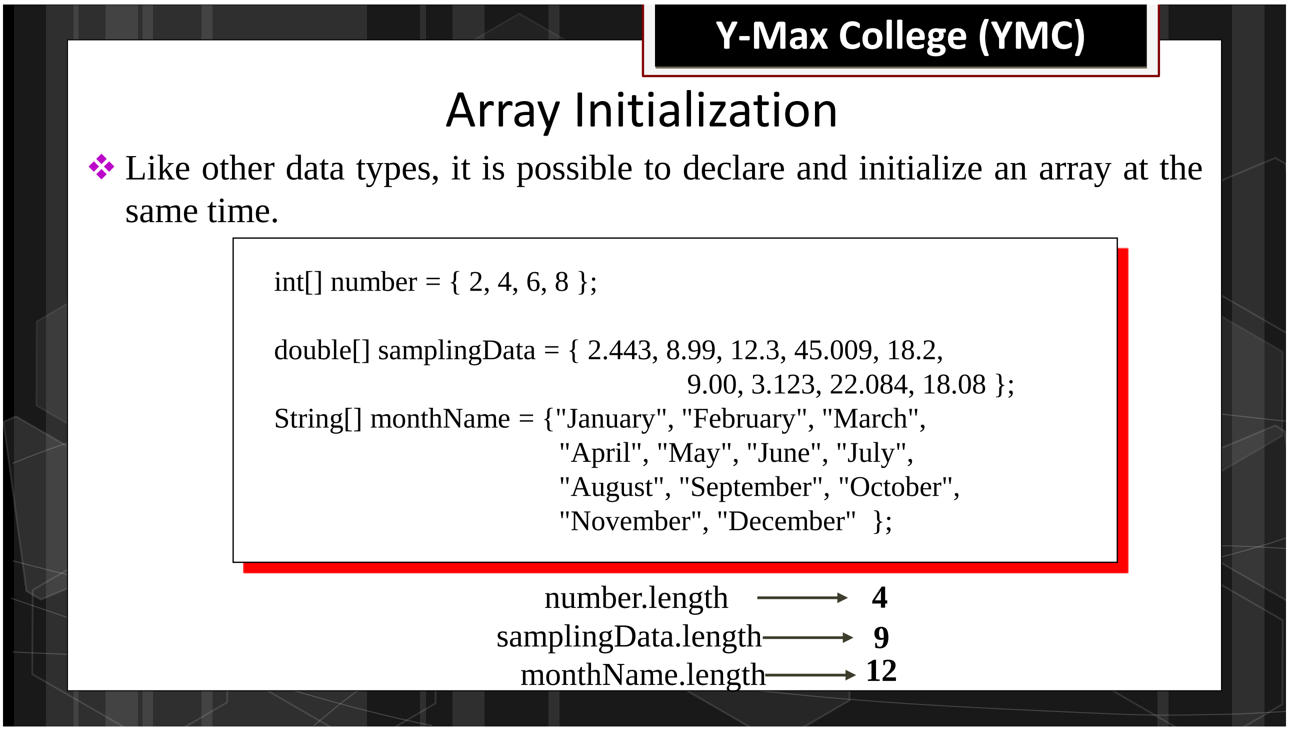
Accessing Individual Elements

❖Individual elements in an array accessed with the indexed expression.

```
double[] rainfall = new double[12];
```

The index of the first position in an array is 0.

This indexed expression refers to the element at position #2



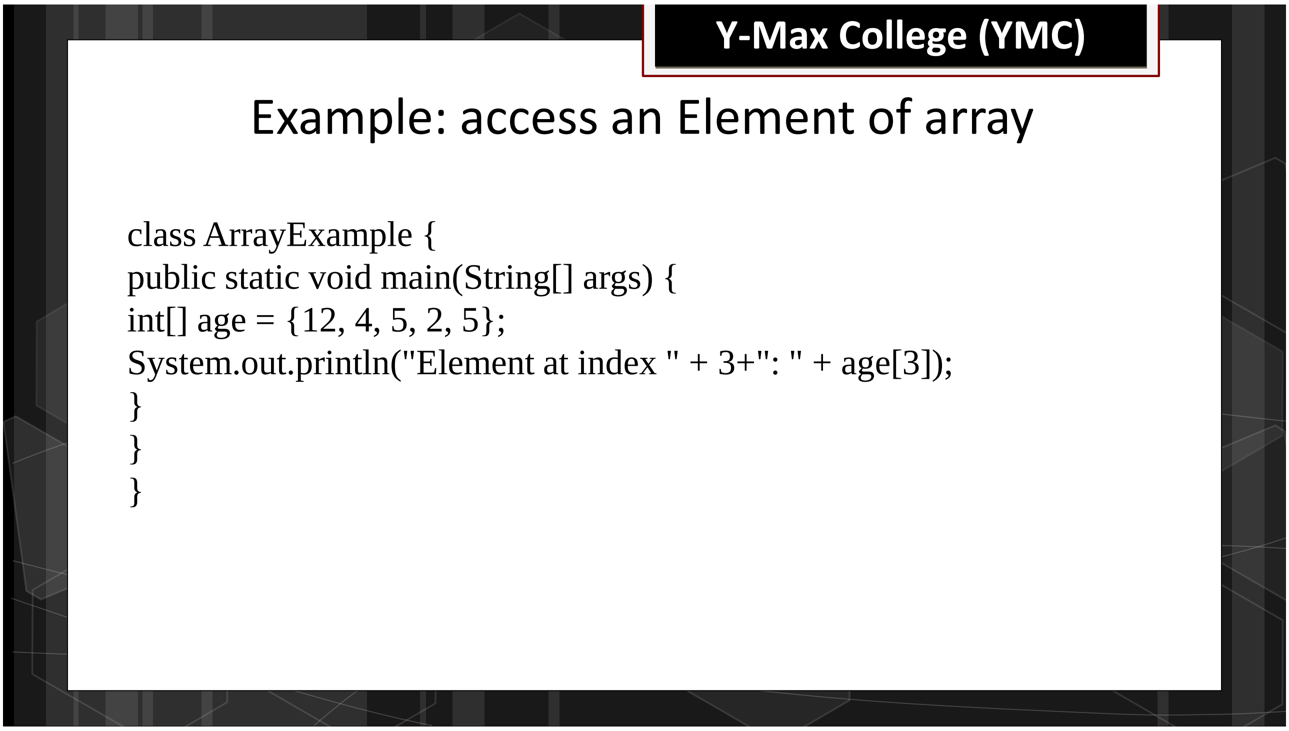
Y-Max College (YMC)

Array Initialization

❖ Like other data types, it is possible to declare and initialize an array at the same time.

```
int[] number = { 2, 4, 6, 8 };

double[] samplingData = { 2.443, 8.99, 12.3, 45.009, 18.2,
                                                  9.00, 3.123, 22.084, 18.08 };
String[] monthName = {"January", "February", "March",
                                   "April", "May", "June", "July",
                                   "August", "September", "October",
                                   "November", "December"  };
```

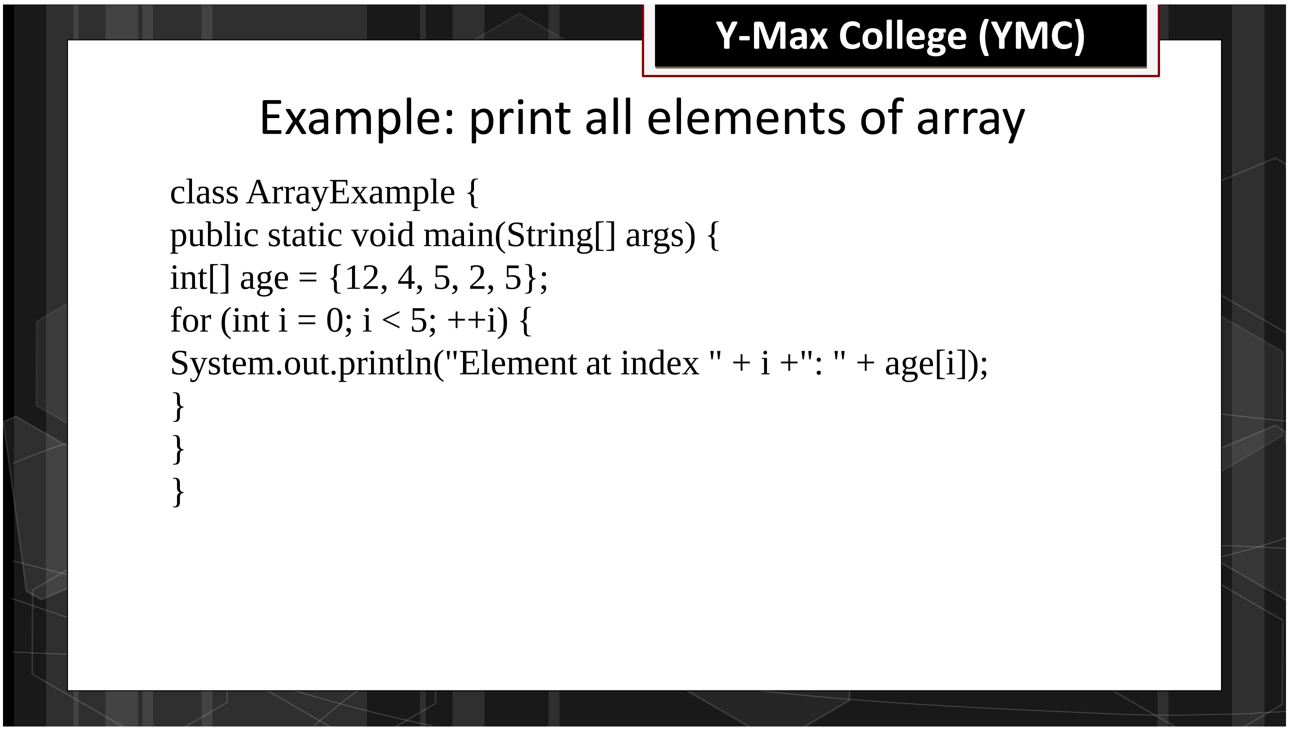
number.length → **4**
samplingData.length → **9**
monthName.length → **12**



Y-Max College (YMC)

Example: access an Element of array

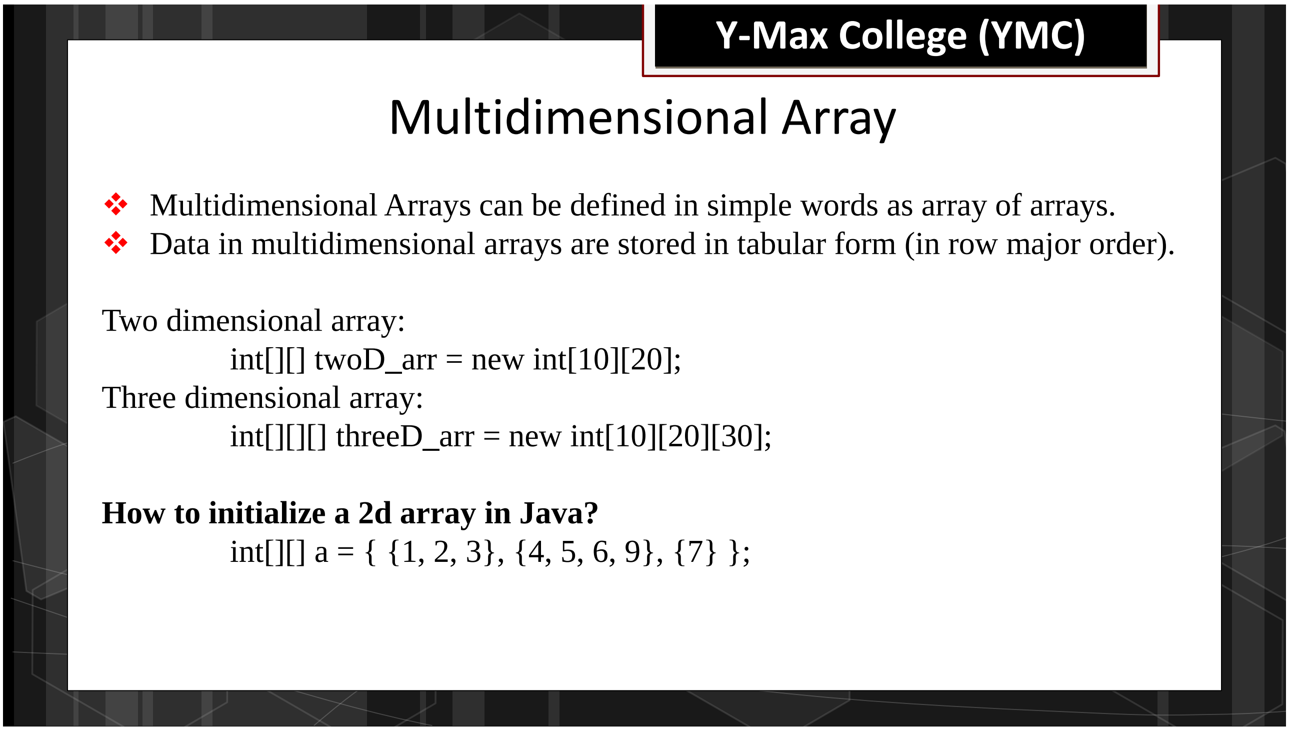
```
class ArrayExample {
public static void main(String[] args) {
int[] age = {12, 4, 5, 2, 5};
System.out.println("Element at index " + 3+": " + age[3]);
}
}
}
```



Y-Max College (YMC)

Example: print all elements of array

```
class ArrayExample {
public static void main(String[] args) {
int[] age = {12, 4, 5, 2, 5};
for (int i = 0; i < 5; ++i) {
System.out.println("Element at index " + i +": " + age[i]);
}
}
}
```



Y-Max College (YMC)

Multidimensional Array

- Multidimensional Arrays can be defined in simple words as array of arrays.
- Data in multidimensional arrays are stored in tabular form (in row major order).

Two dimensional array:

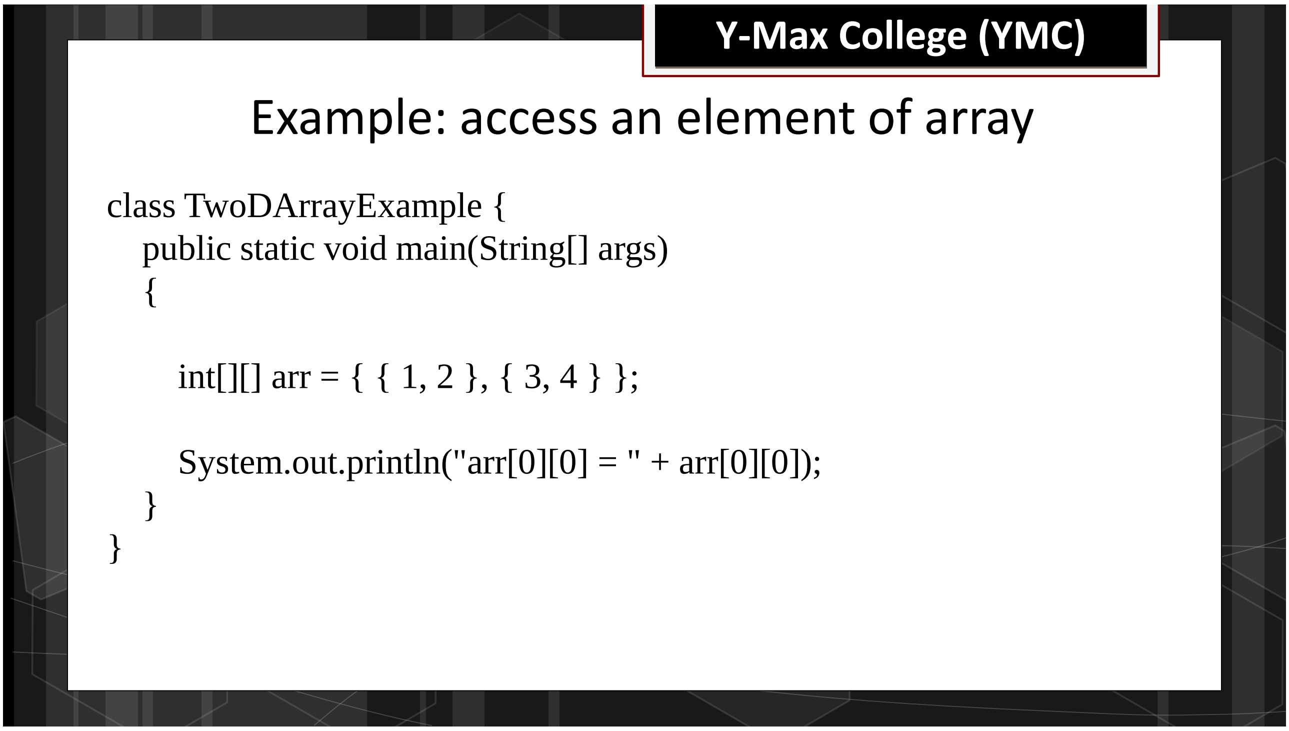
```
        int[][] twoD_arr = new int[10][20];
```

Three dimensional array:

```
        int[][][] threeD_arr = new int[10][20][30];
```

How to initialize a 2d array in Java?

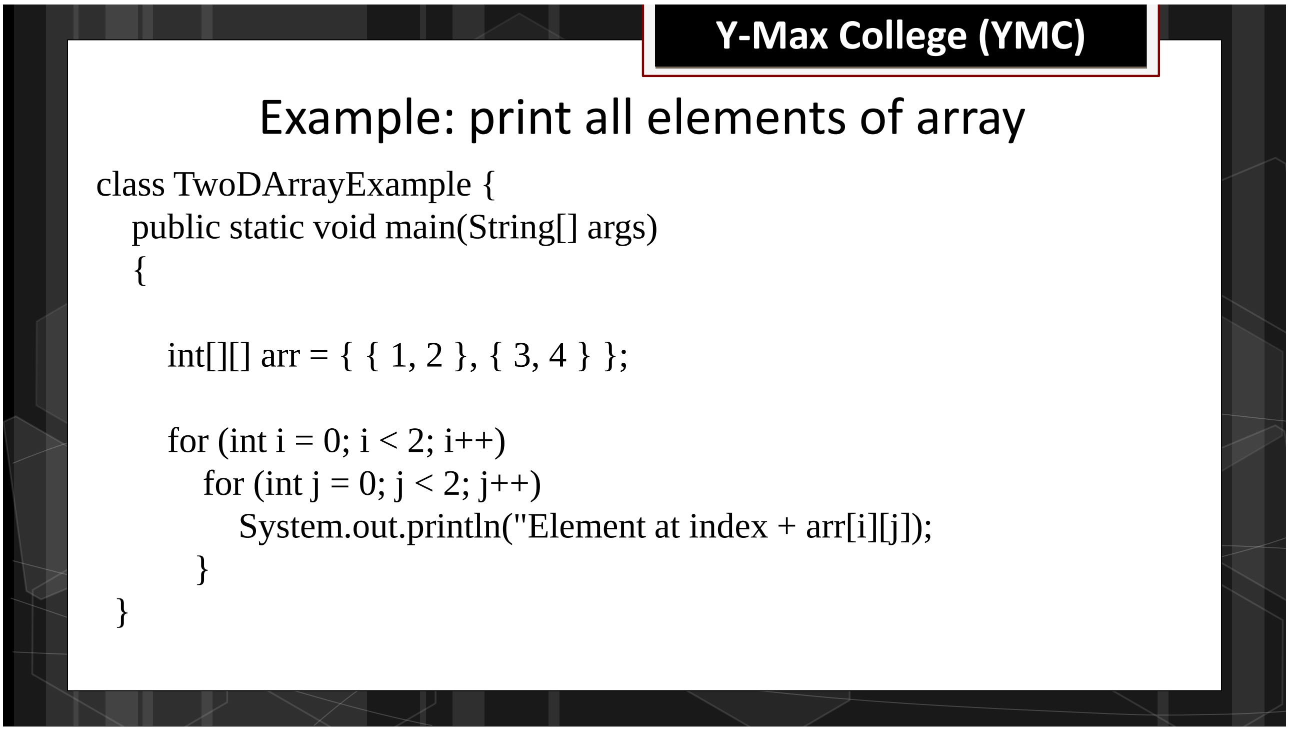
```
        int[][] a = { {1, 2, 3}, {4, 5, 6, 9}, {7} };
```



Y-Max College (YMC)

Example: access an element of array

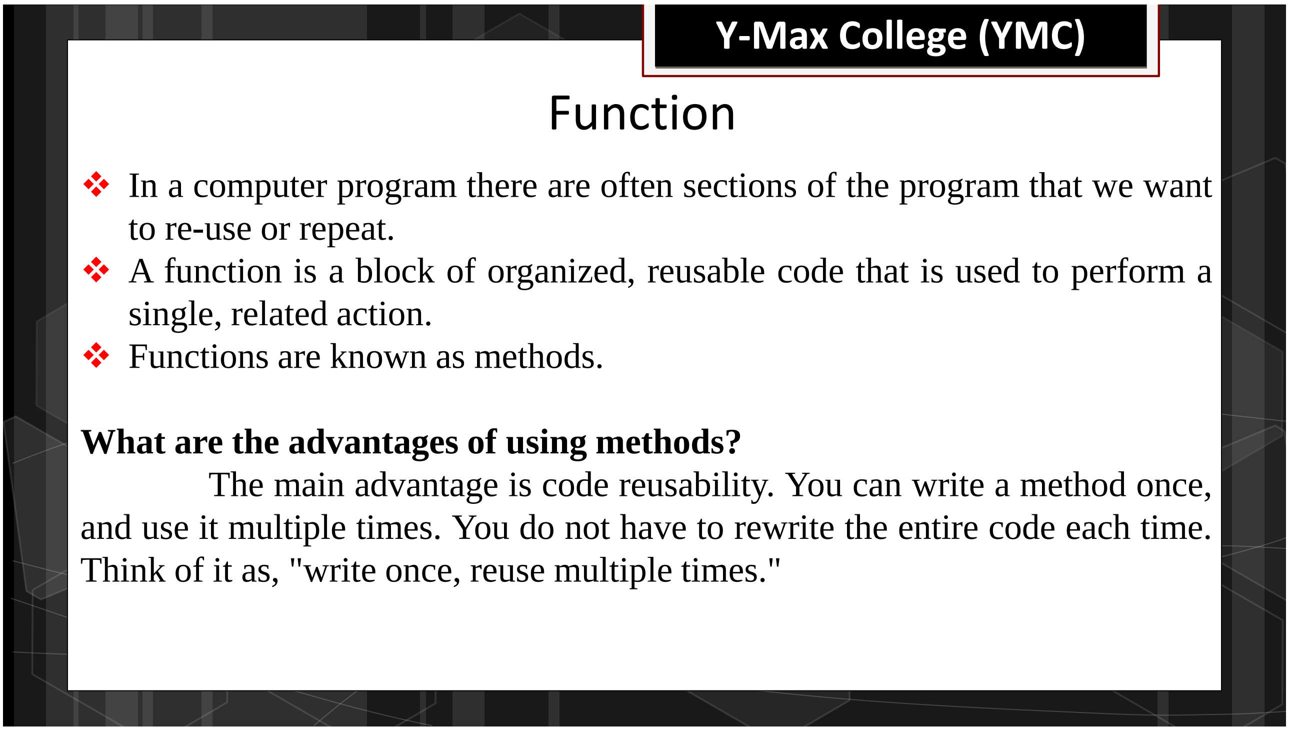
```
class TwoDArrayExample {
   public static void main(String[] args)
   {

      int[][] arr = { { 1, 2 }, { 3, 4 } };

      System.out.println("arr[0][0] = " + arr[0][0]);
   }
}
```



Y-Max College (YMC)

Example: print all elements of array

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

      int[][] arr = { { 1, 2 }, { 3, 4 } };

      for (int i = 0; i < 2; i++)
         for (int j = 0; j < 2; j++)
            System.out.println("Element at index + arr[i][j]);
         }
 }
```



Y-Max College (YMC)

Function

- In a computer program there are often sections of the program that we want to re-use or repeat.
- A function is a block of organized, reusable code that is used to perform a single, related action.
- Functions are known as methods.

What are the advantages of using methods?

The main advantage is code reusability. You can write a method once, and use it multiple times. You do not have to rewrite the entire code each time. Think of it as, "write once, reuse multiple times."

Y-Max College (YMC)

Types of Java Methods

Depending on whether a method is defined by the user, or available in standard library, there are two types of methods:

- Standard Library Methods
- User-defined Methods

Y-Max College (YMC)

Standard Library Methods

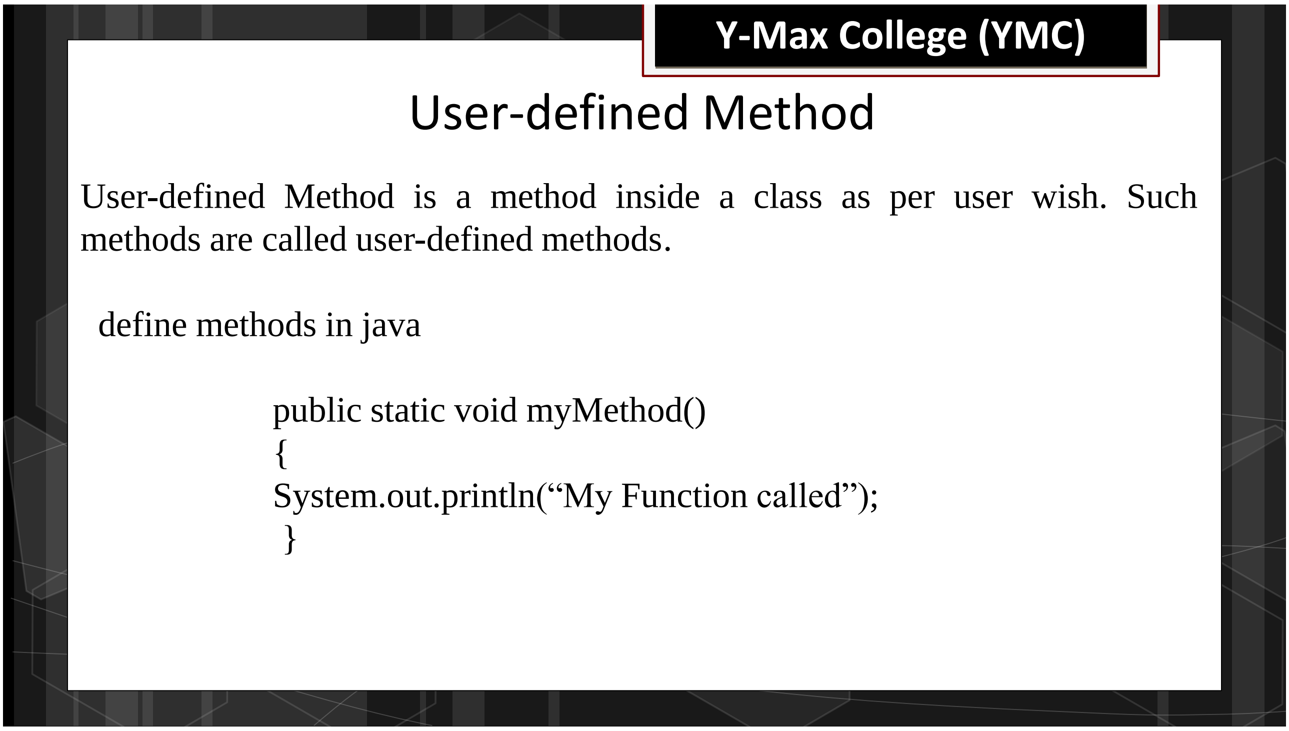
The standard library methods are built-in methods in Java that are readily available for use. These standard libraries come along with the Java Class Library (JCL) in a Java archive (*.jar) file with JVM and JRE.

eg..

- print() is a method of java.io.PrintSteam. The print("...") prints the string inside quotation marks.
- sqrt() is a method of Math class. It returns square root of a number.

eg..

```
        public class Numbers {
        public static void main(String[] args) {
        System.out.print("Square root of 4 is: " + Math.sqrt(4));
}        }
```



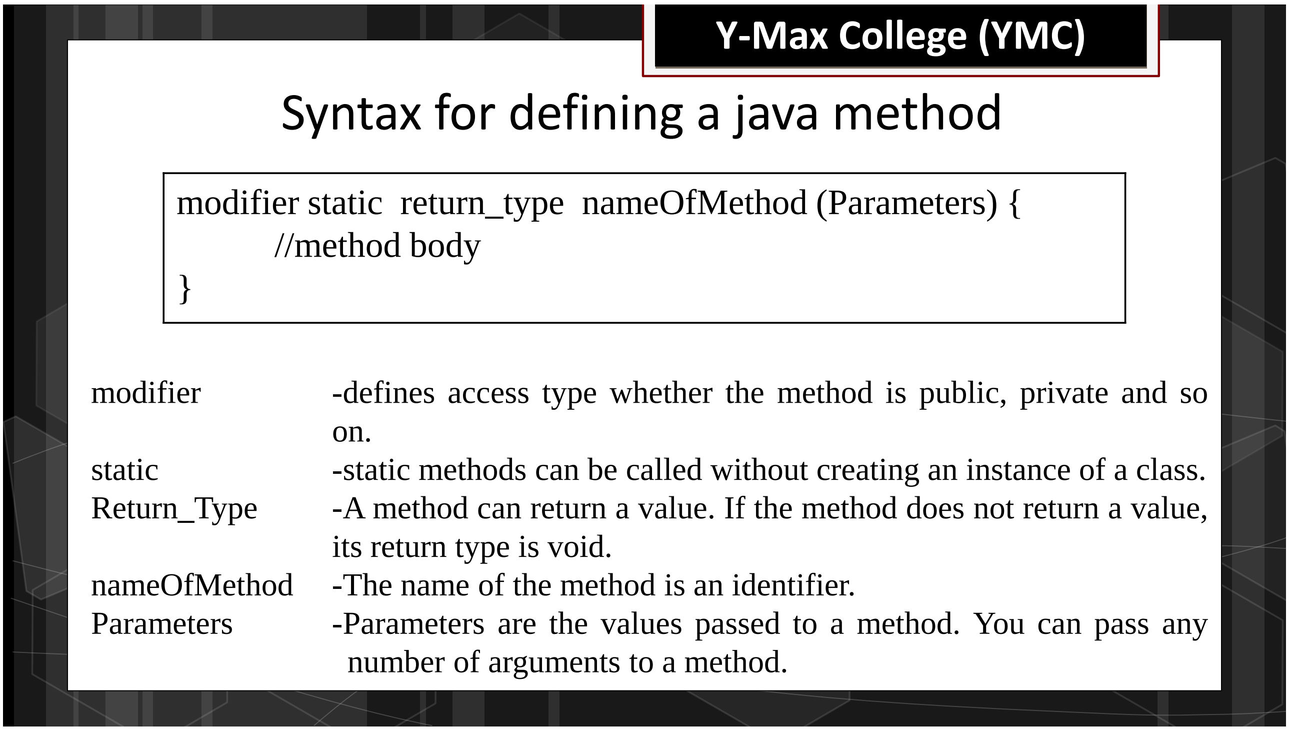
Y-Max College (YMC)

User-defined Method

User-defined Method is a method inside a class as per user wish. Such methods are called user-defined methods.

define methods in java

```
public static void myMethod()
{
System.out.println("My Function called");
 }
```

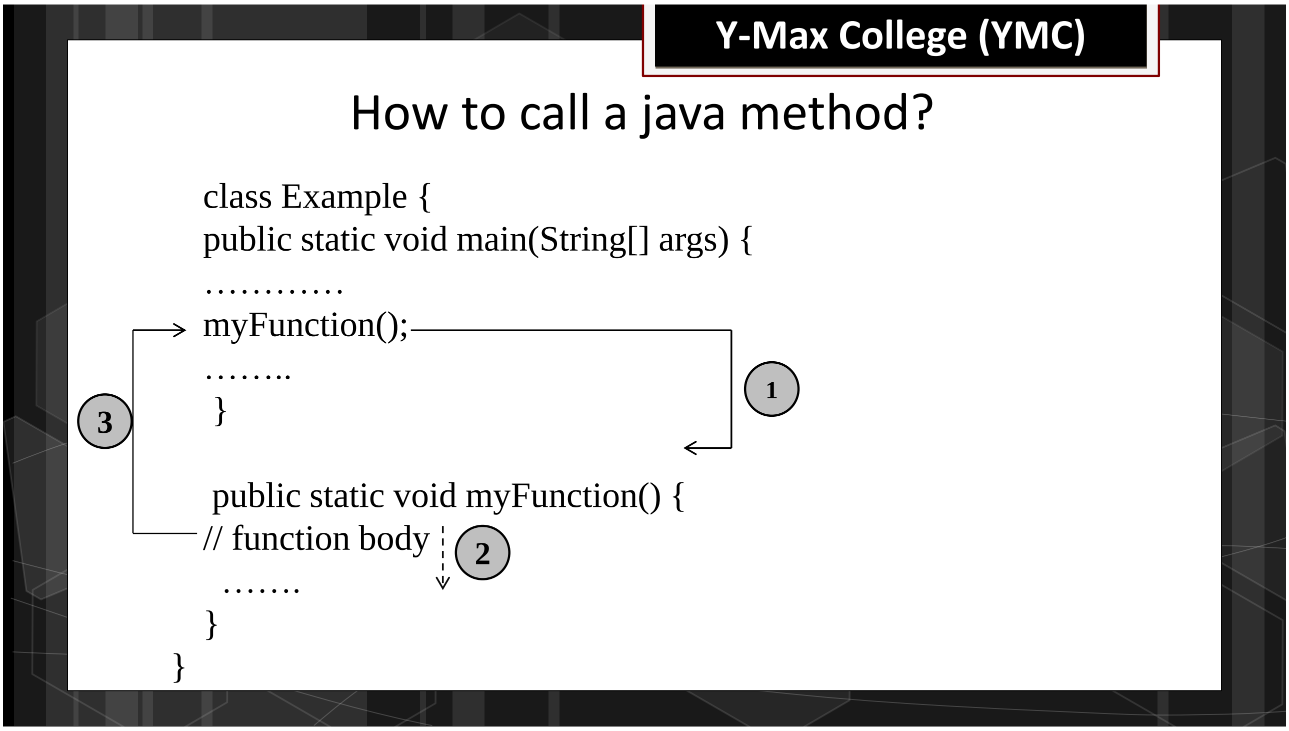


Y-Max College (YMC)

Syntax for defining a java method

```
modifier static  return_type  nameOfMethod (Parameters) {
          //method body
}
```

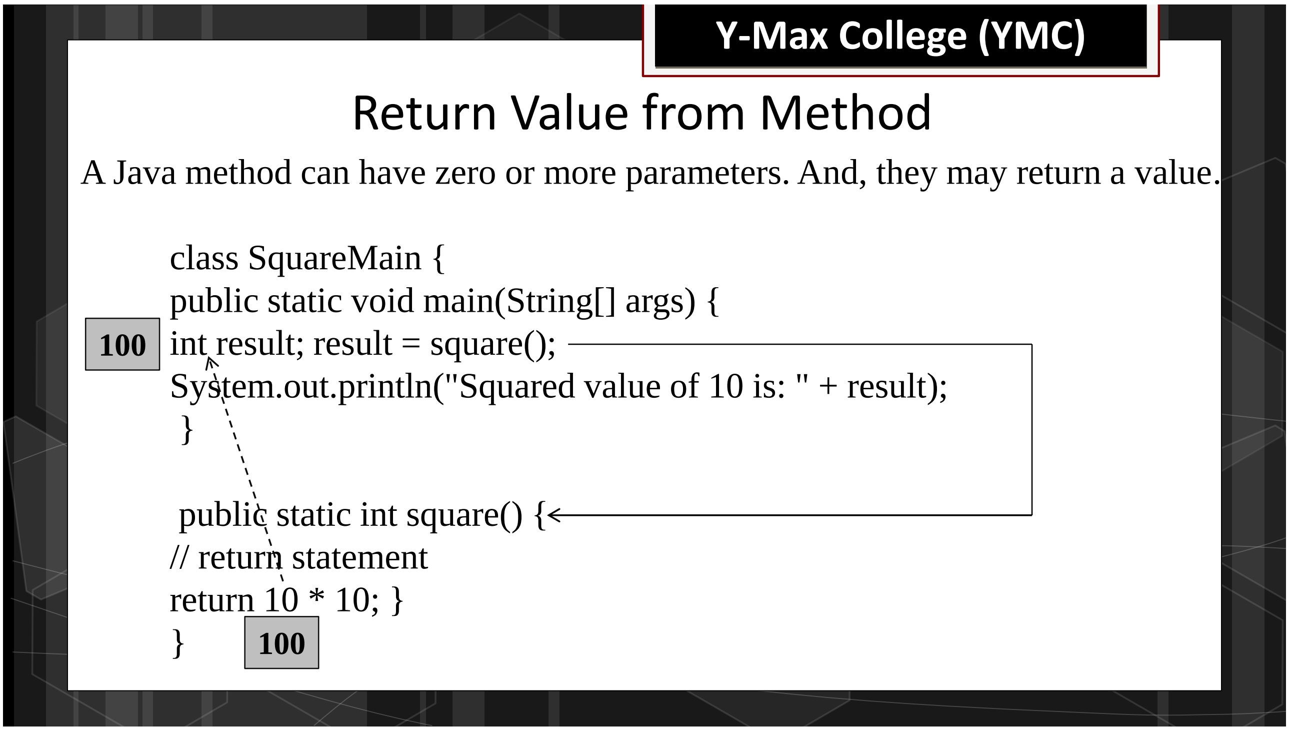
modifier	-defines access type whether the method is public, private and so on.
static	-static methods can be called without creating an instance of a class.
Return_Type	-A method can return a value. If the method does not return a value, its return type is void.
nameOfMethod	-The name of the method is an identifier.
Parameters	-Parameters are the values passed to a method. You can pass any number of arguments to a method.



Y-Max College (YMC)

How to call a java method?

```
class Example {
public static void main(String[] args) {
…………
myFunction();
……..
 }

 public static void myFunction() {
// function body
  …….
}
}
```



Y-Max College (YMC)

Return Value from Method

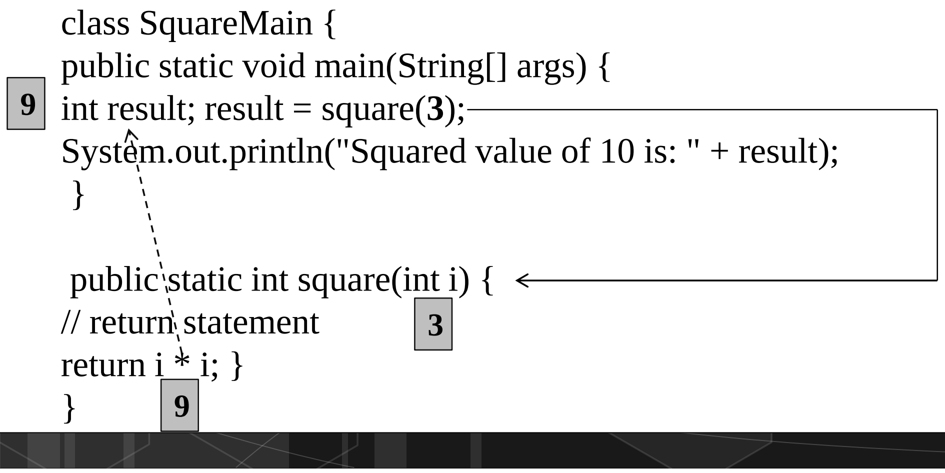
A Java method can have zero or more parameters. And, they may return a value.

```
class SquareMain {
public static void main(String[] args) {
int result; result = square();
System.out.println("Squared value of 10 is: " + result);
 }

 public static int square() {
// return statement
return 10 * 10; }
}
```

Y-Max College (YMC)

Method Accept Arguments and Returning Value

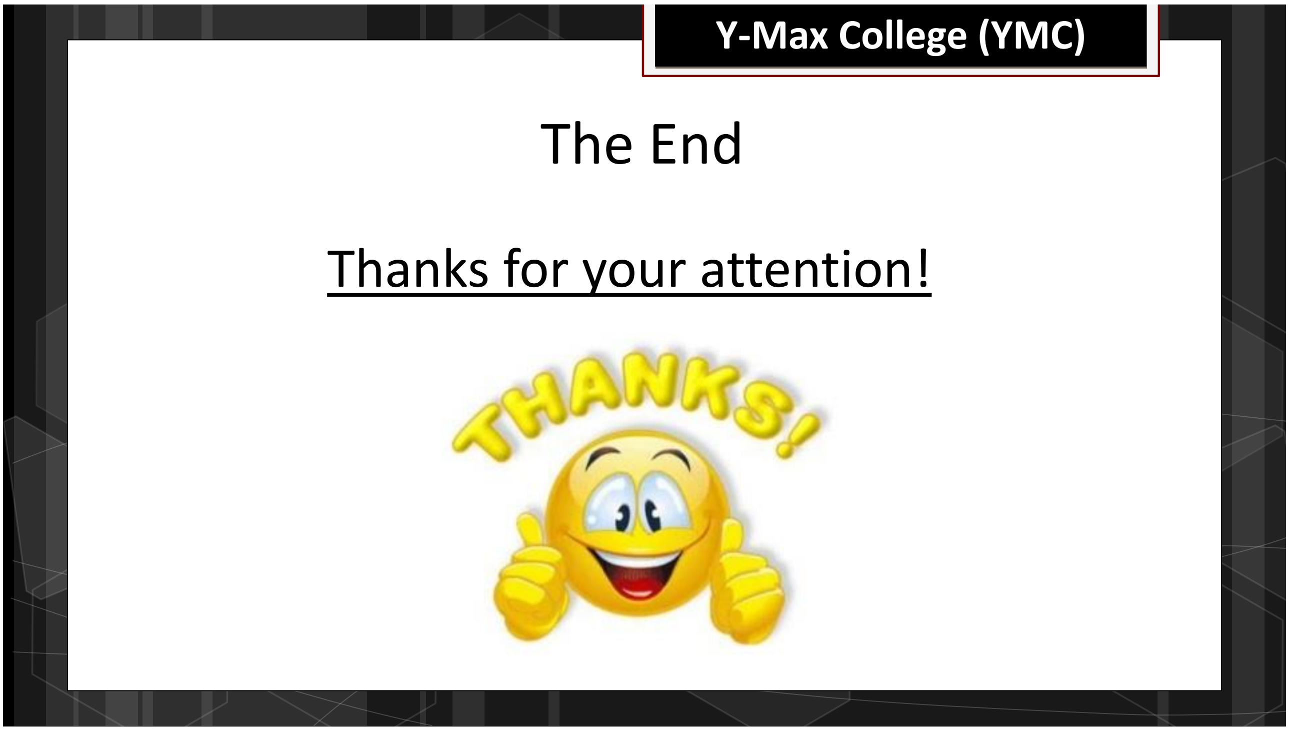
A Java method accepts parameters. And, they may return a value.



```
class SquareMain {
public static void main(String[] args) {
int result; result = square(3);
System.out.println("Squared value of 10 is: " + result);
 }

 public static int square(int i) {
// return statement
return i * i; }
}
```

Y-Max College (YMC)

Any Questions….?



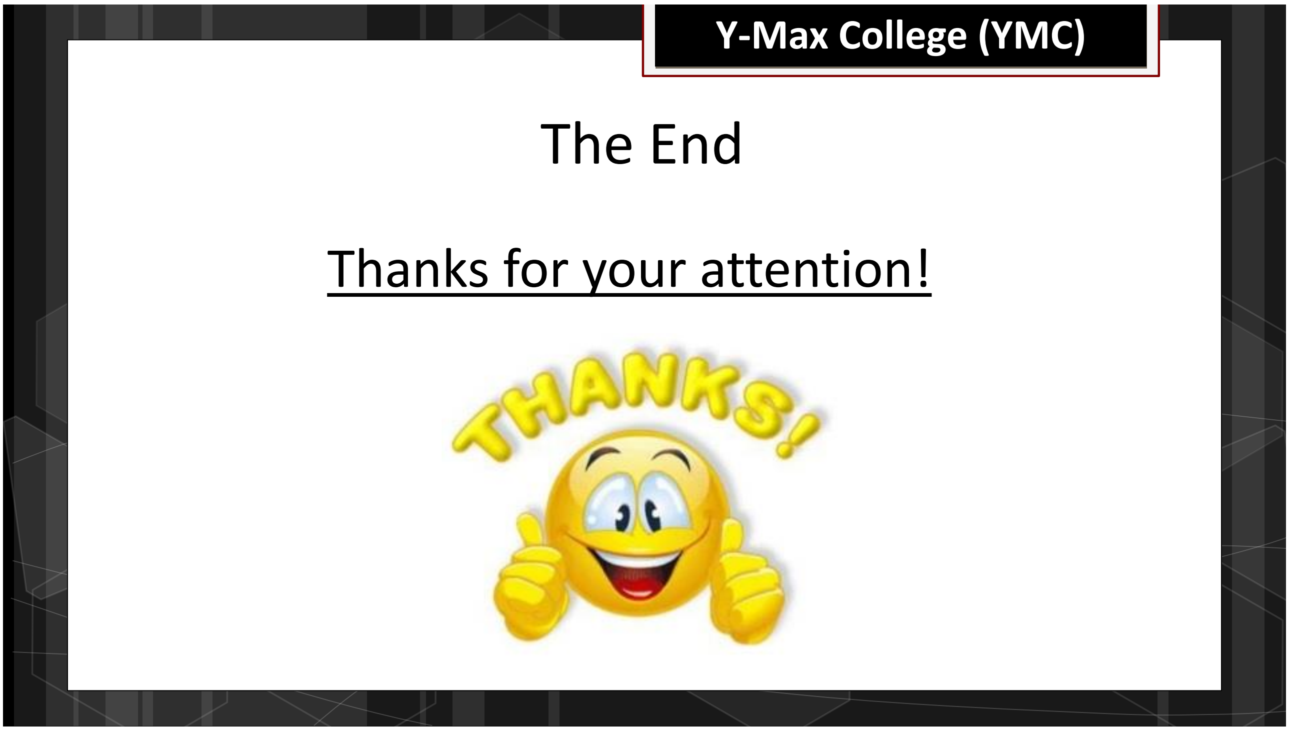
Y-Max College (YMC)

The End

Thanks for your attention!

Y-Max College (YMC)

Any Questions….?



Y-Max College (YMC)

The End

Thanks for your attention!