JAVA

* Why Java?

1. Platform independent or architecture independent(WORA)

Platform -> Combination of Operating System + underlined hardware

* C & C++

C compiler produces an executable file which run on only specific platform.

Example: Compile C code on windows it can run on windows only.

* Extension of source file for java -> .java
* Extension of compiler output -> .class

(intermediate byte code)

WORA -> Write Once and Run Anywhere

Run on any database

Run on any web server

1. Simple & Robust (Strong, fail proof)
2. Secure
3. Automatic Garbage Collection

In C++ if we are creating an object but no longer need it, we need to clean it from the memory(By destructor, delete)

Java -> Garbage Collector

1. Inherent Multi-Threaded support
2. Object Oriented Support

* Why java is not completely object oriented?

Ans: for the purpose of efficiency, java supports primitive data types.

Primitive types are not object types

Also download IntelliJ IDE

* Structure of java program

1. Main Method:

Entry point of a java program.

public static void main(String[] args)

{//code to be executed }

public: Access modifier that makes the method accessible from anywhere.

static: Allows the method to be called without creating the object

void: Indicates that the method does not return any value

String[] args: Parameter to receive command-line arguments

Day2:

* Variables:

What?

* A variable is a container that holds data that can change during the execution.

Syntax:

dataType variableName = value;

* Datatypes:

Java has 8 primitive datatypes:

1. int: used for integers
2. float: used for decimal numbers
3. char: Used for single character
4. boolean: Used for true/false values.
5. byte: used for small integers
6. short: used for small integers
7. long: Used for large integers
8. double: Used for large decimal numbers

* Declaring and Initializing variables
* Declaration: int x;
* Initialization: x = 10;
* Combined: int x=10;

public class Main {  
 public static void main(String[] args) {  
 int age = 24; //Integer  
 float price = 99.9f; //Float(f suffix is mandatory)  
 char grade = 'A'; //Character  
 boolean isEligible=true; //Boolean  
  
 System.*out*.println("Age : "+ age);  
 System.*out*.println("Price: "+ price);  
 System.*out*.println("Grade: "+ grade);  
 System.*out*.println("Eligibility: "+ isEligible);  
   
 }  
 }

Comments

1. Single line Comment: Starts with //
2. Multi-Line Comment : Starts with /\* ends with \*/

* Operators:

What?

Operators are symbols that performs operations on variables or values.

Categories:

1.Arithmetic Operators(+,-,\*,/,%)

2.Relational Operators(==,!=,>,<,>=,<=)

3.Logical Operator(&&,||,!)

* Arithmetic Operators(+,-,\*,/,%)
* Used for basic mathematical operations
* public class ArithmeticExample {  
   public static void main(String[] args) {  
   int a=10,b=5;  
   System.*out*.println("Addition: "+(a+b));  
   System.*out*.println("Subtraction: "+(a-b));  
   System.*out*.println("Multiplication: "+(a\*b));  
   System.*out*.println("Division: "+(a/b));  
   System.*out*.println("Modulus: "+(a%b));  
   }  
  }
* Relational Operators

Used to compare values

public class RelationalExample {  
 public static void main(String[] args) {  
 int a =10,b=20;  
 System.*out*.println("a == b : "+(a==b));  
 System.*out*.println("a != b : "+(a!=b));  
 System.*out*.println("a > b : "+(a>b));  
 System.*out*.println("a < b : "+(a<b));  
 }  
}

* Logical Operators:

Used to combine multiple conditions

public class LogicalExample {  
 public static void main(String[] args) {  
 int age = 20;  
 boolean hasId = true;  
  
 //AND operator  
 System.*out*.println(age>=18 && hasId); //true  
  
 //OR operator  
 System.*out*.println(age < 18 || hasId); //true  
  
 //NOT operator  
 System.*out*.println(!(age >= 18)); // false  
 }  
}

* Control Statements

Control statements allow us to control the flow of execution in java program.

1. if statement

The if statement is used to execute a block of code if a condition evaluates to true.

Syntax:

if(condition){

//Code to execute if the condition is true

}

public class IfExample {  
 public static void main(String[] args) {  
 int number = 10;  
  
 if(number > 0){  
 System.*out*.println("The number is positive");  
 }  
 }  
}

2.if-else statement

The if-else statement provides an alternative path of execution when condition is false.

Syntax:

if(condition){

//Code to execute if the condition is true

} else {

//Code to execute If the condition is false

}

public class IfElseExample {  
 public static void main(String[] args) {  
 int number = -5;  
  
 if(number > 0){  
 System.*out*.println("The number is positive");  
 } else {  
 System.*out*.println("The number is negative");  
 }  
 }  
}

* if-else-if Ladder

to test multiple conditions

Syntax:

if(condition1){

// code to execute if condition1 is true

} else if(condition2){

// code to execute if condition2 is true

} else {

//Code to execute if none of the conditions are true

}

public class IfElseIfExample {  
 public static void main(String[] args) {  
 int marks = 75;  
  
 if(marks >= 90){  
 System.*out*.println("Grade: A+");  
 } else if (marks >= 80) {  
 System.*out*.println("Grade: A");  
 } else if (marks >= 70) {  
 System.*out*.println("Grade: B");  
 } else if (marks >= 60) {  
 System.*out*.println("Grade: C");  
 } else {  
 System.*out*.println("Please Attempt again");  
 }  
 }  
}