* What is Java Project ?

Source and set of libraries , set depending is all call together project .

* **What is** sources ?

Multiple together package is call the source

* What is package ?

Multiple class and interface is call the package

* What is class ?

Multiple method and concept is call the class

* What is method and concept?

Set of variables , set Date type , Set statement is call the Method,

Method having the return Type

For concept The not have the return type .

**Int 🡪 date type**

**X and y 🡪 variables**

**Z = x +y 🡪 Statement**

**10,20 🡪 constant**

Int x = 10;

Int y = 20;

Int z = x + y;

What is variable ?

One which hold value is not but is variable , We have there type if variable local variable, instances-variable , static- variable

**local variable:-**

**definition**:- variables declared inside a method constructor or block .

**Scope**:- only accessible within the method contains the or block where they are declared.

**instances-variable:-**

**definition**:- variables declared inside a class but outside any method constructor or block

**Scope:**- Accessible by all methods constructors and block in the class . Each instance of the class has its own copy .

**static- variable or class variables:-**

**definition:-** variables declared with the static keyworld inside but outside any method constructor or block.

**Scope:- s**hared among all instances of the class only one copy exits regardless of how many instances are created .

|  |  |
| --- | --- |
| abstract | :- Specifies that a class or method will be implemented later, in a subclass. |
| assert | :- Used for debugging purposes to make an assertion. |
| boolean | :- A data type that can hold true or false values. |
| break | :- Exits from a loop or a switch statement. |
| byte | :- A data type that can hold 8-bit data values. |
| case | :- Defines a block of code in a switch statement. |
| catch | :- Catches exceptions generated by try statements. |
| catch | :- Catches exceptions generated by try statements. |
| char | :- A data type that can hold a single 16-bit Unicode character. |
| class | :- Declares a new class. |
| continue | :- Skips the current iteration of a loop and proceeds to the next iteration. |
| default | :- Specifies the default block of code in a switch statement. |
| do | :- Starts a do-while loop. |
| double | :- A data type that can hold 64-bit floating-point numbers. |
| else | :- Indicates alternative branches in an if statement. |
| enum | :- Declares an enumerated type. |
| extends | :- Indicates that a class is derived from another class or interface. |
| final | :- Indicates that a variable holds a constant value or that a method cannot be overridden. |

|  |  |
| --- | --- |
| finally | A block of code that will always be executed after a try-catch block. |
| float | A data type that holds a 32-bit floating-point number. |
| for | Used to start a for loop. |
| if | Tests a true/false expression and branches accordingly. |
| implements | Specifies that a class implements an interface. |
| import | References other classes. |
| instanceof | Tests whether an object is an instance of a specific class or interface. |
| int | A data type that can hold a 32-bit signed integer. |
| interface | Declares an interface. |
| long | A data type that holds a 64-bit integer. |
| native | Specifies that a method is implemented with native (platform-specific) code. |
| new | Creates new objects. |
| null | Indicates that a reference does not refer to anything. |
| package | Declares a Java package. |
| private | An access specifier indicating that a method or variable may be accessed only in the class it’s declared in. |
| protected | An access specifier indicating that a method or variable may only be accessed in the class it’s declared in, subclasses, or other classes in the same package. |
| public | An access specifier indicating that a method or variable can be accessed by any other class. |
| return | Exits from a method and optionally returns a value. |
| short | A data type that can hold a 16-bit integer. |
| static | Indicates that a method or variable belongs to the class, rather than instances of the class. |
| super | Refers to the superclass of the current object. |
| switch | A control statement that allows a variable to be tested for equality against a list of values. |
| synchronized | Ensures that a method or block of code is only accessed by one thread at a time. |
| this | Refers to the current object in a method or constructor. |
| throw | Used to explicitly throw an exception. |
| throws | Indicates what exceptions may be thrown by a method. |
| transient | Prevents serialization of fields. |
| try | Starts a block of code that will be tested for exceptions. |
| void | Specifies that a method does not return a value. |
| volatile | Indicates that a variable may be changed unexpectedly. |
| while | Starts a while loop. |

**Non-Access Modifiers–Final & Static**

Keywords which have the function don’t have access modified

Final :-

final behavewell be different is different in class and method variables

* One you declare variables is a final cannot modified the value
* One you declared as a method as final cannot be Overriding method
* One you declare class as a final we cannot extend the class

Class :-

final class FinalClass {

// Class content

}

// Attempting to extend a final class will result in a compile-time error

// class SubClass extends FinalClass { } // Error

**final Method**

class ParentClass {

final void display() {

System.out.println("This is a final method.");

}

}

class ChildClass extends ParentClass {

// Attempting to override a final method will result in a compile-time error

// void display() { // Error

// System.out.println("Trying to override.");

// }

}

### final Variable

class FinalVariableExample {

final int MAX\_VALUE = 100;

void display() {

// MAX\_VALUE = 200; // Error: cannot assign a value to final variable

System.out.println("MAX\_VALUE: " + MAX\_VALUE);

}

public static void main(String[] args) {

FinalVariableExample obj = new FinalVariableExample();

obj.display();

}

}

**Static Modifier**

* Only static behavior to different way one static variables and methods
* Static variables can access any where in the project without create object.
* Static method can access any where in project without create object

**Class – 1**

Access Modifiers

* What is Package in Java Project?

It refers which class is belongs to that package

Ex:- package com.method.ref

* What is class in java?
* Access modifiers ?

| **Access Modifier** | **Within Class** | **Within Package** | **Outside Package by Subclass** | **Outside Package** |
| --- | --- | --- | --- | --- |
| Private | Yes | No | No | No |
| Default | Yes | Yes | No | No |
| Protected | Yes | Yes | Yes | No |
| Public | Yes | Yes | Yes | Yes |

**Pubilc:-**

* Which helps to access, Anywhere in the project ,
* Can be **Variable, Method , class** .
* Accessible from any other class.

**Private:-**

* Only in with in class
* **Variable , method .**
* Accessible only within the same class.
* Not accessible from any other class.

**Protected: -**

* You can access with in packages.
* Variables , method .
* Accessible within the same package and by subclasses in different packages.

**Default :-**

* You can access with in packages.
* Variables , method.
* Accessible only within the same package.
* If no access modifier is specified, it is considered default.

**Class – 2**

**JVM,JDK,JRE,IDE**

**JVM** :- When you write java class it compiler the java class and it Translation

(or)

The runtime execution environment for Java bytecode.

Compilation Process :- JVM as set up standards to intend fine grammar **mistake and syntax Error**

* **JRE: -** Java runtime environment .
* The JRE is the **implementation** of the JVM. It’s what physically exists.
* **Purpose**: It provides the necessary runtime environment for Java applications to run.\
* A set of libraries and other files that the JVM uses at runtime.
* Ensures that dependencies (like standard Java libraries) are available to your programs.

JDK: -

* **Purpose**: It’s a **software development environment** used for creating Java applications and applets.
* Its environment products for the JVM and JRE

IDE:-

* Which is user inter face product to build web and stand application
* An IDE combines several essential tools into one cohesive environment:
* **Code Editor**
* **Debugger:**
* **Build Tools.**
* **Version Control Integration:**
* **Project Management**
* **Refactoring Tools**
* **Testing Frameworks**
* **Documentation Generation:**

**Popular Java IDEs**:

**Eclipse**: A versatile IDE with extensive plugin support. It’s cross-platform and widely used in the Java community.

**Create a java Project :**

* **If create java projects must be install JDK**
* **IDE must be install**
* Clike on file 🡪 New 🡪 other 🡪 Tab 🡪 Mavene project

**Class – 3**

**Static Variable**

* Static variables are also known as Class Variables.
* If you declare any variable as static, it is known as a static variable.
* A static variable is common to all the instances (or objects) of the class because it is a class level variable.
* In other words you can say that only a single copy of static variable is created and shared among all the instances of the class.

**Advantages of static variable:-**

* \*\* It makes your program memory efficient (i.e., it saves memory).

**Static and non static methods**

* \* from one static method to we can call another static method
* \*\*\*we cant call non static method from static method
* \* we can call a static method from non static methods
* \* we can call a non static method to non static method

**Static Block**

**Ex:-**

**Static {**

**Static block**

**}**

* Why static :- If you wish to load the content by default

**Concepts and Method**

* Concepts and Method
* Concepts does not have the return type
* Where method has return type
* Concepts name must be class matches the class name
* Method should not have same as class as class name
* Concepts will be executed when you create object for class.
* Method will be executed by create object and call that method with help of object referece
* Status method will be executed by default .

**Class – 4**