**Enum:**

* An enum is a special "class" that represents a group of constants (unchangeable variables, like final variables).
* To create an enum, use the enum keyword (instead of class or interface), and separate the constants with a comma. Note that they should be in uppercase letters:

enum Level {

LOW,

MEDIUM,

HIGH

}

can access enum constants with the dot(.) syntax: ***Level myVar = Level.MEDIUM;***

Enum is short for "***enumerations***", which means "***specifically listed***".

**Enum inside a Class:** we can have an enum inside a class:

public class MyClass {

enum Level {

LOW,

MEDIUM,

HIGH

}

public static void main(String[] args) {

Level myVar = Level.MEDIUM;

System.out.println(myVar);

}

}

**Enum in a Switch Statement**: Enums are often used in switch statements to check for corresponding values:

enum Level {

LOW,

MEDIUM,

HIGH

}

public class MyClass {

public static void main(String[] args) {

Level myVar = Level.MEDIUM;

switch(myVar) {

case LOW:

System.out.println("Low level");

break;

case MEDIUM:

System.out.println("Medium level");

break;

case HIGH:

System.out.println("High level");

break;

}

}

}

**Loop Through an Enum:** The enum type has a values() method, which returns an array of all enum constants. This method is useful when you want to loop through the constants of an enum:

for (Level myVar : Level.values()) {

System.out.println(myVar);

}

**Difference between Enums and Classes**

* An enum can, just like a class, have attributes and methods. The only difference is that enum constants are public, static and final (unchangeable - cannot be overridden).
* An enum cannot be used to create objects, and it cannot extend other classes (but it can implement interfaces).

**Why And When To Use Enums?**

* Use enums when you have values that we know aren't going to change, like month days, days, colors, deck of cards, etc.