**When we say:**

Days arr[] = Days.values();

**1. What is values() ?**

* For every enum, the **compiler automatically generates** a static method:

public static Days[] values();

* This method returns an **array of all enum constants** in the order they are declared.

So here, values() returns:

new Days[] { Days.Friday, Days.Saturday, Days.Sunday }

**2. What is the type of this array?**

* Type is Days[] (array of enum constants).
* Each element in this array is an **object of type Days**, and since every enum extends java.lang.Enum, they also inherit methods from Enum.

**3. How can we call .name() and .ordinal() ?**

* Both name() and ordinal() are **instance methods defined in java.lang.Enum**:
* public final String name(); // returns the exact name of the constant
* public final int ordinal(); // returns its position (0-based)
* Since each element in the array is a Days object, and Days extends Enum<Days>, they have access to these methods.

**So our loop:**

for(int i=0;i<arr.length;i++) {

System.out.println(arr[i].name()+"\t"+arr[i].ordinal());

}

* Iterates over an array of Days objects.
* For each object, calls:
  + name() → the constant’s identifier ("Friday", "Saturday", "Sunday")
  + ordinal() → the constant’s position (0, 1, 2)

**Summary :**

* values() returns an **array of enum objects** (Days[]).
* Yes, when you traverse that array, each element is an **object** and you are invoking the name() and ordinal() methods of that object (inherited from Enum).