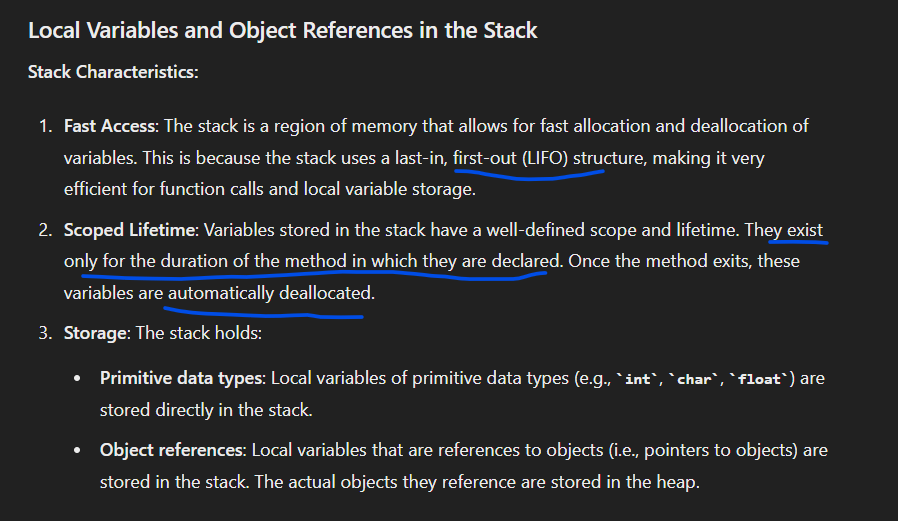
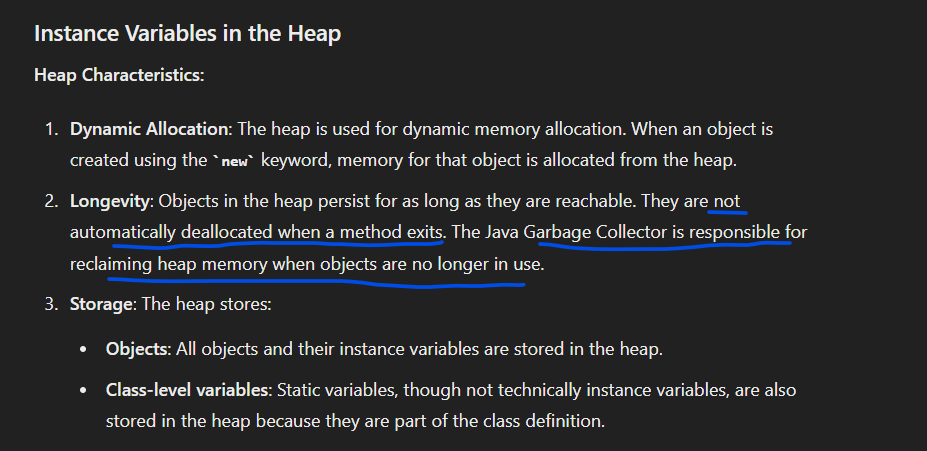


A blackboard with white text and green arrows

Description automatically generated

Actual memory area for method is stack.





A screenshot of a computer

Description automatically generated

A screen shot of a computer code

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

In Java, an abstract class does not provide 100% abstraction. An abstract class can have both abstract methods (methods without an implementation) and concrete methods (methods with an implementation). This means that an abstract class can include both fully defined behavior and placeholders for behavior that must be defined in subclasses.

On the other hand, an interface in Java is designed to provide 100% abstraction. In Java, an interface can only contain abstract methods (until Java 8, which introduced default and static methods). All the methods in an interface are implicitly abstract (unless they are default or static methods introduced in later versions of Java).

A black screen with white text

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A yellow background with black text

Description automatically generated

Enums in Java are similar to classes in several ways because they can have their own fields, methods, and constructors, making them powerful tools for representing a fixed set of constants with additional functionality. Here are the key reasons why enums in Java are like classes:

**1. Fields and Methods**

Enums can have fields and methods, just like regular classes. This allows enums to carry additional data and behavior beyond just being constants.

**Example:**

java

Copy code

public enum Day {

MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, SATURDAY, SUNDAY;

public boolean isWeekend() {

return this == SATURDAY || this == SUNDAY;

}

}

public class Main {

public static void main(String[] args) {

Day day = Day.SATURDAY;

System.out.println(day.isWeekend()); // Output: true

}

}

**2. Constructors**

Enums can have constructors, which can be used to initialize fields. However, the constructors must be private or package-private (default) because enums are implicitly final and you cannot create instances of them using the new keyword.

**Example:**

java

Copy code

public enum Season {

WINTER("Cold"), SPRING("Warm"), SUMMER("Hot"), FALL("Cool");

private String description;

private Season(String description) {

this.description = description;

}

public String getDescription() {

return description;

}

}

public class Main {

public static void main(String[] args) {

for (Season season : Season.values()) {

System.out.println(season + ": " + season.getDescription());

}

}

}

**3. Implementing Interfaces**

Enums can implement interfaces, allowing you to define a common set of behaviors for different enum constants.

**Example:**

java

Copy code

interface Displayable {

void display();

}

public enum TrafficLight implements Displayable {

RED {

@Override

public void display() {

System.out.println("Stop");

}

},

YELLOW {

@Override

public void display() {

System.out.println("Caution");

}

},

GREEN {

@Override

public void display() {

System.out.println("Go");

}

}

}

public class Main {

public static void main(String[] args) {

for (TrafficLight light : TrafficLight.values()) {

System.out.print(light + ": ");

light.display();

}

}

}

**4. Static Methods**

Enums can have static methods, just like classes. This allows you to define utility methods related to the enum type.

**Example:**

java

Copy code

public enum MathOperation {

ADD, SUBTRACT, MULTIPLY, DIVIDE;

public static MathOperation fromString(String operation) {

switch (operation.toUpperCase()) {

case "ADD":

return ADD;

case "SUBTRACT":

return SUBTRACT;

case "MULTIPLY":

return MULTIPLY;

case "DIVIDE":

return DIVIDE;

default:

throw new IllegalArgumentException("Unknown operation: " + operation);

}

}

}

public class Main {

public static void main(String[] args) {

MathOperation op = MathOperation.fromString("add");

System.out.println(op); // Output: ADD

}

}

**5. Enum Constants Are Instances**

Enum constants are implicitly public, static, and final instances of the enum type. This means that each enum constant is a unique instance of the enum class.

**Example:**

java

Copy code

public enum Singleton {

INSTANCE;

public void show() {

System.out.println("Singleton instance method called");

}

}

public class Main {

public static void main(String[] args) {

Singleton singleton = Singleton.INSTANCE;

singleton.show(); // Output: Singleton instance method called

}

}

**Summary**

Enums in Java are similar to classes because they can have fields, methods, constructors, and static methods. They can also implement interfaces. This makes enums much more powerful than simple constants, as they can encapsulate related data and behavior in a type-safe manner.

Enums provide a robust way to define a fixed set of constants, and their class-like capabilities allow them to be used in sophisticated programming scenarios, adding clarity and maintainability to your code.

A screenshot of a computer

Description automatically generated

A black screen with white text

Description automatically generated

A screenshot of a computer

Description automatically generated

PUBLIC:

A screen shot of a computer program

Description automatically generated

PROTECTED:  
A screenshot of a computer program

Description automatically generated

A screenshot of a video

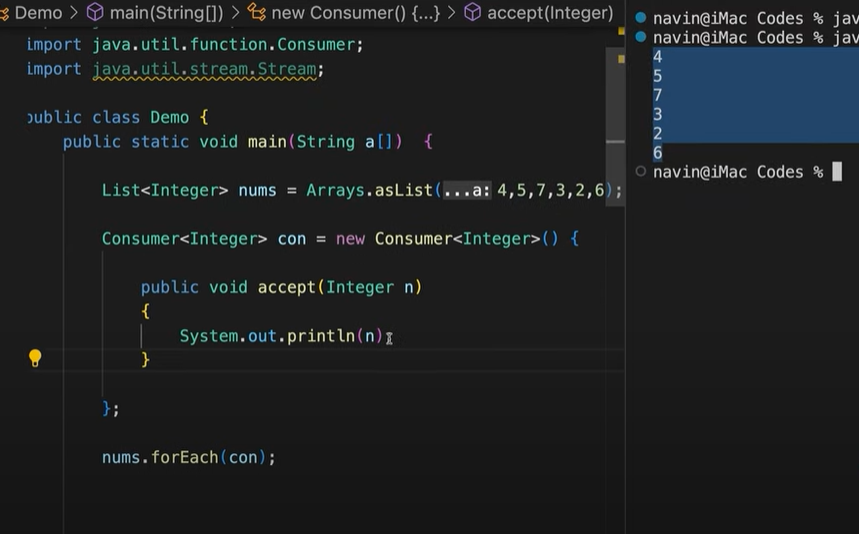
Description automatically generated

A screen shot of a computer program

Description automatically generated

A white and black text on a white background

Description automatically generated



A screenshot of a computer program

Description automatically generated

