1. double num = 2.\_718;
2. float f2 = 5.0; // c4……….to convert a double (5.0) to a float
3. String car, bus = "petrol";

car = car + bus;

Compilation error

1. double num1, int num2 = 1; // C1 Compilation error
2. for(int i=0; 0; i++) Compilation error

if (true)

break;

}

break statement can be used within loop like for, while or do while loop or within switch statement.

Integer num1 = 400;

Integer num2 = 400;

Integer class support the range of -128 to 127.

Hence, String is more specific than Object class. Therefore, null is matched as a string object.

Null is not an object in Java.

class Code {

public static void main(String args[]) {

System.out.println(value());

}

static int value() {

static int data = 0;

return data;

}

}

int y = 08;

thread1.start();

thread1.stop();

thread1.start();

Answer: Runtime error

**public static void main() {**

**int odd = 1;**

**if (odd) {**

**System.out.println("odd");**

**} else {**

**System.out.println("even");**

**}**

**}**

**}**

**.** Type mismatch error

String name = null; // This would cause a NullPointerException System.out.println(name.length()); // To prevent the exception, add a null check if (name != null) { System.out.println(name.length()); }

// Interface example

interface Vehicle {

void drive();

}

class Car implements Vehicle {

public void drive() {

System.out.println("Driving a car");

}

}

// Abstract class example

abstract class Shape {

int x;

int y;

Shape(int x, int y) {

this.x = x;

this.y = y;

}

abstract void draw();

}

class Circle extends Shape {

int radius;

Circle(int x, int y, int radius) {

super(x, y);

this.radius = radius;

}

void draw() {

System.out.println("Drawing a circle");

}

}

finalize method garbage

**ChatGPT said:**

ChatGPT

The finalize() method in Java is associated with garbage collection. Here's an explanation:

**What is the finalize() Method?**

* The finalize() method is a special method defined in the Object class.
* It is called by the **Garbage Collector (GC)** when it determines there are no more references to an object, just before the object is removed from memory.

**Syntax:**

java

Copy code

protected void finalize() throws Throwable {

// Cleanup code

}

* The method can be overridden in your class to define custom cleanup actions before the object is destroyed.

**Usage:**

The finalize() method is typically used to release resources such as closing files, database connections, or cleaning up unmanaged resources (e.g., native code).

java

Copy code

class Example {

@Override

protected void finalize() throws Throwable {

System.out.println("Finalize method called!");

// Cleanup code here

}

}

public class Main {

public static void main(String[] args) {

Example obj = new Example();

obj = null; // Dereference the object

// Request garbage collection

System.gc();

// Give the GC some time to run (not guaranteed)

try { Thread.sleep(1000); } catch (InterruptedException e) {}

}

}

**Key Points about finalize():**

1. **Not Guaranteed to Run**:
   * The finalize() method is not guaranteed to execute.
   * It depends on whether and when the garbage collector runs.
2. **Unpredictability**:
   * The garbage collector's execution is non-deterministic; you cannot predict when (or even if) the finalize() method will be invoked.
3. **Deprecated**:
   * Starting from **Java 9**, the finalize() method is marked as **deprecated**.
   * It is considered unreliable and is officially removed in **Java 18**.
4. **Alternatives**:
   * Use **try-with-resources** for automatic resource management.
   * Use **close()** methods or implement the AutoCloseable interface for explicit cleanup.

**Why Avoid finalize()?**

* **Performance Impact**: Garbage collection with finalize() can be slower as the objects are placed in a queue for finalization before being collected.
* **Uncertainty**: Cleanup might not occur promptly, leading to resource leaks.

**Example with try-with-resources (Preferred Way):**

Instead of using finalize(), manage resources using try-with-resources:

java

Copy code

import java.io.FileWriter;

import java.io.IOException;

public class Main {

public static void main(String[] args) {

try (FileWriter writer = new FileWriter("output.txt")) {

writer.write("Hello, World!");

} catch (IOException e) {

e.printStackTrace();

}

}

}

Here, the FileWriter resource is automatically closed after use.

**Summary:**

* The finalize() method is used for cleanup before an object is garbage collected.
* It is unreliable, has been deprecated, and is no longer a best practice.
* Use modern alternatives like try-with-resources or close() for resource management.