Understanding Objects in Java: A Simplified Explanation  
Hello Coders so today I will be talking about how object is created and where it gets Stored in the memory

In Java, objects are fundamental components of Object-Oriented Programming (OOP). Here’s a brief overview:

What is an Object?

An object in Java is an instance of a class.

It represents a real-world entity or concept, combining data (attributes) and behaviors (methods) into a single unit.

For example, a Car class might have attributes like color and model

We can create an object of class by using the new keyword

Example

Car myCar = new Car();

Here in the above code snippet

I have created a new Car object and assigns it to the variable myCar.  
Now the real question is where does the memory of this object get stored?

So first let's take a code snippet

public class NewExample1 {

    public static void main(String[] args) {

        NewExample1 obj=new NewExample1();

                              int n;

    }

}

Now as you know in the memory section the local objects are in the stack section

So in the above code, we can see that I have created an object of class and simple variable “n”

of **int** datatype

So here the real question is

**Where is the memory of the variable and object stored?**

Here is the tricky part

**Memory Allocation: Stack vs. Heap**

When you create an object using new, the actual object is stored in the **heap** memory area of the Java Virtual Machine (JVM). The variable **mycar** is a **reference** that points to the object in the heap.

**Local Variables and Objects**

Local variables, like n in your example, are typically stored on the **stack**. They have a limited scope and are destroyed when the method or block they're declared in finishes executing.

**Object References and Garbage Collection**

Even if the reference to an object (like **mycar**) goes out of scope, the object itself remains in the heap until it's no longer needed.

**Imagine the garbage collector as a janitor in a classroom.** The janitor's job is to clean up trash and recycle things that are no longer needed. In Java, the garbage collector does a similar job for objects.

**When an object is no longer used** (meaning there's no way to access it from your code anymore), **the garbage collector will eventually come along and "clean it up."** This means it will remove the object from memory, freeing up space for other things.

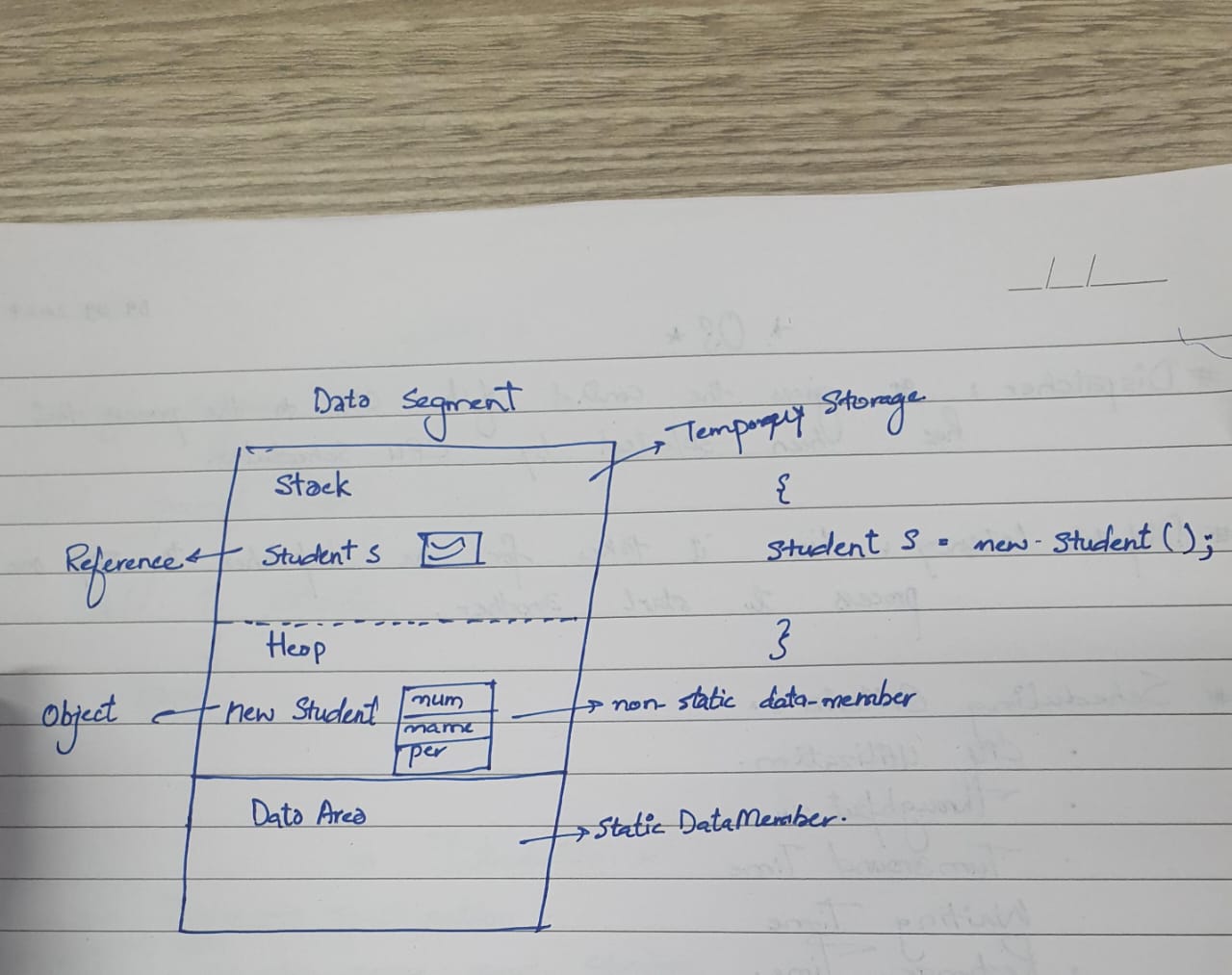
Note

As you know the data member of the class

So in the heap, only non-static members are stored

and in the data area, the static members are stored

Here is the diagrammatic representation of the above point



Here's an interesting fact told by my Professor Dinesh Banswal

He told us that Java never gives us an  object address even if we

convert the address using the toString() method it will provide us with the reference address, not the actual address

**Key Points:**

* Objects are created using the new keyword.
* Objects are stored in the heap memory.
* Variables that reference objects are stored on the stack.