### why in System.out.println("\*"); character ‘S’ is capital, is there any convension in java for it?

Great question! Yes, the **capital "S"** in System.out.println("\*"); follows **Java naming conventions** — and it's not just about style; it reflects how Java is structured. Let's break it down:

**✅ Why is System capitalized?**

* System is a **class** in the **java.lang** package.
* By **Java naming conventions**, **class names start with a capital letter**.

So, System is capitalized because it's a class — just like String, Math, Integer, etc.

**✅ Why is out lowercase?**

* out is a **static field** (a public static member variable) inside the System class.
* It's a reference to a PrintStream object (usually connected to the console).
* By convention, **fields (variables) in Java start with a lowercase letter**.

**✅ What about println?**

* println is a **method** of the PrintStream class.
* Methods also follow the **camelCase convention**, starting with a lowercase letter.

**🧠 TL;DR - Java Naming Conventions**

| **Element Type** | **Convention** | **Example** |
| --- | --- | --- |
| Class/Interface | PascalCase | System, String, Scanner |
| Variable/Field | camelCase | out, inputStream, bufferedReader |
| Method | camelCase | println(), readLine(), calculateSum() |
| Constant (static final) | ALL\_CAPS | PI, MAX\_VALUE |

So, System.out.println("\*"); fits perfectly into the conventions:

* System = class
* out = static field
* println = method

Let me know if you want a visual breakdown or example of how System.out works internally!

### Java is a Typed Language, which means that for any variable, we only need to write the variable type beforehand.

### **Data Types**:

1. **Primitive**:  
   byte (1 byte),  
   boolean (1 byte),  
   char (2 byte),  
   int (4 byte),  
   short (2 byte),  
   long (8 byte),  
   float (4 byte),  
   double (8 byte)
2. **Non-primitive**:  
   String, Array, Class, Object, Interface

🡪 chatgpt:  
Because **Java is platform-independent**, it defines the size of primitive types in the Java Language Specification, not based on the underlying architecture.  
  
Perfect! You've listed all the **Java primitive data types** with their **standard sizes**. Here's a quick, formatted reference with a tiny clarification for boolean, which isn’t strictly defined in size:

**✅ Java Primitive Data Types and Their Sizes**

| **Data Type** | **Size** | **Description** |
| --- | --- | --- |
| Byte | 1 byte | 8-bit signed integer (-128 to 127) |
| boolean | ~1 byte\* | true or false (JVM dependent) |
| Char | 2 bytes | 16-bit Unicode character |
| Short | 2 bytes | 16-bit signed integer |
| Int | 4 bytes | 32-bit signed integer |
| Long | 8 bytes | 64-bit signed integer |
| Float | 4 bytes | 32-bit floating point (IEEE 754) |
| Double | 8 bytes | 64-bit floating point (IEEE 754) |

**🧠 Special Note on boolean:**

Java does not explicitly define the size of a boolean in the Java Language Specification (JLS). It's often **1 byte** in practice (for array storage or memory alignment), but it **can vary depending on the JVM**.

So your list is **spot on** — just with the small footnote on boolean.

Want a cheat sheet image or Java code snippet that prints max/min values for these?

###