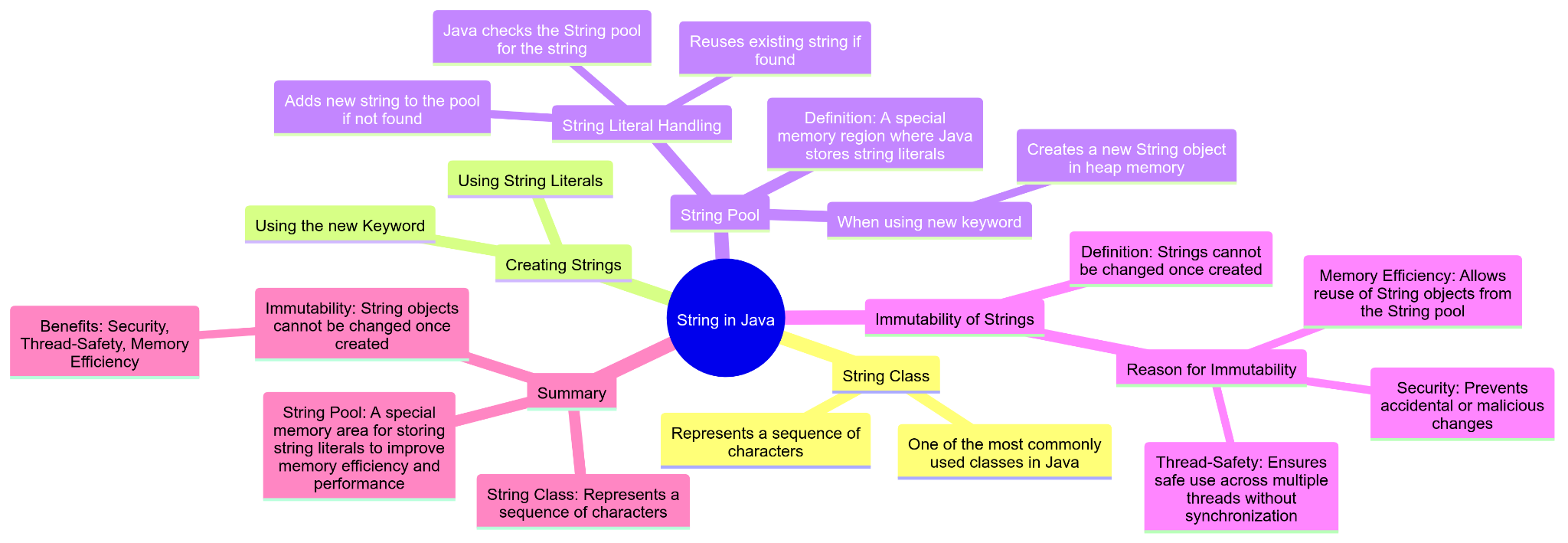
### **Visual Overview (Mindmap)**

****

### **≠––≠**

### **String in Java**

* **String Class: In Java, String is a class that represents a sequence of characters. It's one of the most commonly used classes in Java.**
* **Creating Strings: You can create String objects in two ways:**

**1. Using String Literals:  
String s1 = "Hello";**

**2. Using the new Keyword:  
String s2 = new String("Hello");**

### 

### 

### **String Pool**

* **Definition: The String pool (or intern pool) is a special memory region where Java stores string literals.**
* **String Literal Handling:**
  + **When you create a string using a literal (e.g., String s1 = "Hello";), Java checks the String pool to see if the string already exists. If it does, it reuses the existing string. If not, it adds the new string to the pool.**
  + **String s2 = "Hello";**
  + **When you create a string using the new keyword (e.g., String s2 = new String("Hello");), it creates a new String object in the heap memory, even if an identical string exists in the String pool.**

#### **Example:**

**String s1 = "Hello";**

**String s2 = "Hello";**

**String s3 = new String("Hello");**

**System.out.println(s1 == s2); // true, both refer to the same object in the String pool**

**System.out.println(s1 == s3); // false, s3 refers to a new object in the heap**

### 

### **Immutability of Strings**

* **Definition: Strings in Java are immutable, meaning once a String object is created, it cannot be changed.**
* **Reason for Immutability:**
  + **Security: Prevents accidental or malicious changes.**
  + **Thread-Safety: Ensures safe use across multiple threads without synchronization.**
  + **Memory Efficiency: Allows reuse of String objects from the String pool.**

#### **Example:**

**String s1 = "Hello";**

**s1 = "World"; // This creates a new String object and s1 now refers to "World"**

**String s2 = "Hello";**

**s2.toUpperCase(); // This doesn't change s2, it creates a new String object**

**System.out.println(s1); // Outputs "World"**

**System.out.println(s2); // Outputs "Hello"**

**In the example above, when s1 is reassigned to "World", a new String object is created and s1 now refers to this new object. The original "Hello" string remains unchanged. When toUpperCase is called on s2, it returns a new String object ("HELLO") but does not change the original s2.**

### **Summary**

1. **String Class: Represents a sequence of characters.**
2. **String Pool: A special memory area for storing string literals to improve memory efficiency and performance.**
3. **Immutability: String objects cannot be changed once created, which provides benefits like security, thread-safety, and memory efficiency.**

**Prepared by : Rishi patel**