Java Strings:

1. **String Immutability**:
   Strings in Java are immutable, which means that once a string is created, it cannot be changed. Any operation that seems to modify a string actually creates a new string.

   ```java
   String original = "Hello";
   String modified = original + ", World!";
   // 'original' still holds "Hello", and 'modified' holds "Hello, World!"
   ```

2. **String Comparison Methods**:
   The `compareTo()` method compares two strings lexicographically. It returns a negative value if the first string is "less than" the second, zero if they are equal, and a positive value if the first string is "greater than" the second.

   ```java
   String str1 = "apple";
   String str2 = "banana";
   int result = str1.compareTo(str2); // Result: negative value
   ```

Java Math:

1. **Trigonometric and Exponential Functions**:
   The `Math` class provides methods for various trigonometric and exponential operations, such as `sin()`, `cos()`, `tan()`, `exp()`, and `log()`.

   ```java
   double angle = Math.toRadians(45);
   double sineValue = Math.sin(angle); // Result: 0.7071067811865475
   double expValue = Math.exp(2);      // Result: 7.3890560989306495
   ```

Java Booleans:

1. **De Morgan's Laws**:
   De Morgan's laws are fundamental principles in Boolean algebra. They describe how to negate complex conditions involving multiple logical operators.

   - `!(A && B)` is equivalent to `!A || !B`
   - `!(A || B)` is equivalent to `!A && !B`

These laws are used to simplify and optimize complex boolean expressions.

2. **Boolean Short-Circuiting**:
Short-circuiting can be advantageous in situations where evaluating the second operand of a logical operator could result in an error or unnecessary computation.

```java
boolean result = isNotNull(obj) && obj.isValid(); // 'obj.isValid()' won't be called if 'obj' is null
```

3. **Ternary Operator**:
The ternary operator (`? :`) allows you to express a simple conditional operation in a concise way.

```java
int x = 10;
String result = (x > 5) ? "Greater than 5" : "Less than or equal to 5";
```

These additional aspects provide further insight into the nuances of Java strings, mathematical operations, and boolean logic. By mastering these concepts, you can create more sophisticated and efficient Java programs.

Certainly, let's delve even deeper into these concepts:

Java Strings:

1. **String Formatting (printf)**:
Java offers the `System.out.printf()` method to format strings using placeholders and format specifiers.

```java
String name = "Alice";
int age = 30;
System.out.printf("My name is %s and I am %d years old.", name, age);
// Output: My name is Alice and I am 30 years old.
```

2. **String Interning**:
Java has a concept called string interning, where string literals are stored in a common pool to conserve memory. You can explicitly intern a string using the `intern()` method.

```java
String s1 = "Hello";
String s2 = new String("Hello");
String s3 = s2.intern();
boolean areEqual = s1 == s2; // false
boolean areEqualInterned = s1 == s3; // true
```

Java Math:

1. **Math Class Constants**:
   The `Math` class provides useful constants like `Math.PI` (pi) and `Math.E` (Euler's number).

   ```java
   double circumference = 2 * Math.PI * radius;
   ```

2. **Trigonometric Inverses**:
   The `Math` class offers inverse trigonometric functions like `asin()`, `acos()`, and `atan()`, which return angles based on provided ratios.

   ```java
   double angle = Math.atan(1); // Returns the angle in radians for which tan(angle) = 1
   ```

Java Booleans:

1. **Boolean Arrays and Collections**:
   Boolean values can be used in arrays and collections to represent binary states efficiently.

   ```java
   boolean[] daysOfWeek = new boolean[7]; // Represents if each day is available (true) or not (false)
   ```

2. **Boolean Short-Circuiting in Ternary Operator**:
   Ternary operators can also take advantage of short-circuiting.

   ```java
   int x = 10;
   String result = (x > 5) && (someMethod()) ? "Greater than 5" : "Not greater than 5";
   // 'someMethod()' won't be called if 'x' is not greater than 5
   ```

3. **Bitwise Operations on Booleans**:
   While not commonly used, you can perform bitwise operations on boolean values, although they might not have the same intuitive behavior as they do on integers.

   ```java
   boolean b1 = true;
   boolean b2 = false;
   boolean bitwiseAnd = b1 & b2; // Result: false
   boolean bitwiseOr = b1 | b2;  // Result: true
   boolean bitwiseXor = b1 ^ b2; // Result: true
   ```

These additional insights should give you a comprehensive understanding of Java strings, mathematical operations, and boolean logic. By mastering these concepts, you'll be well-equipped to write robust and efficient Java programs.