### Java Strings:

1. **StringBuilder vs. StringBuffer**:
   Both `StringBuilder` and `StringBuffer` are used for string manipulation. The key difference is that `StringBuilder` is not thread-safe, while `StringBuffer` is thread-safe due to synchronized methods.

   ```java
   StringBuilder stringBuilder = new StringBuilder();
   StringBuffer stringBuffer = new StringBuffer();
   ```

2. **String Pool**:
   Java maintains a special memory area called the "string pool" to store unique string literals, which helps conserve memory by reusing identical strings.

   ```java
   String s1 = "Hello";
   String s2 = "Hello"; // Reuses the same string from the pool
   boolean areEqual = s1 == s2; // true
   ```

### Java Math:

1. **BigDecimal for Precise Decimal Calculations**:
   While `double` and `float` are suitable for most floating-point calculations, for precise decimal calculations (e.g., financial calculations), you should use the `BigDecimal` class to avoid rounding errors.

   ```java
   import java.math.BigDecimal;

   BigDecimal num1 = new BigDecimal("0.1");
   BigDecimal num2 = new BigDecimal("0.2");
   BigDecimal sum = num1.add(num2); // Result: 0.3
   ```

2. **Mathematical Constants and Functions**:
   The `Math` class provides various mathematical constants like `Math.PI` and `Math.E`. It also offers methods like `Math.sinh()`, `Math.cosh()`, `Math.tanh()` for hyperbolic trigonometric functions.

From math import *
System.out.println(Math.sinh(90));
System.out.println(`Math.cosh(90));

### Java Booleans:

1. **Boolean Streams and Collections**:
   Java 8 introduced the Stream API, which can handle boolean values as well. Streams provide a concise and functional way to work with collections.

   ```java
   boolean[] flags = { true, false, true };
   long countTrue = Arrays.stream(flags).filter(flag -> flag).count(); // Result: 2
   ```

2. **Boolean.valueOf()**:
   The `Boolean` class has a `valueOf()` method that can be used to get `Boolean` objects from boolean values.

   ```java
   Boolean boolObject = Boolean.valueOf(true); // Returns a Boolean object
   ```

3. **Boolean Logical Operations on Sets**:
   You can perform logical operations on sets using methods like `retainAll()`, `addAll()`, `removeAll()`, etc.

   ```java
   Set<String> set1 = new HashSet<>(Arrays.asList("A", "B", "C"));
   Set<String> set2 = new HashSet<>(Arrays.asList("B", "C", "D"));
   set1.retainAll(set2); // set1 now contains "B" and "C"
   ```

4. **BooleanSupplier Functional Interface**:
   Java 8 introduced the `BooleanSupplier` functional interface, which represents a supplier of boolean values. It can be useful for lazy evaluation.

   ```java
   BooleanSupplier boolSupplier = () -> someMethod();
   boolean result = boolSupplier.getAsBoolean();
   ```

These advanced concepts should provide you with a deeper understanding of Java strings, mathematical operations, and boolean logic. As you become more familiar with these concepts, you'll be able to write even more sophisticated and efficient Java programs.