Why Java does not support Operator overloading

Unlike C++, Java doesn't support operator overloading. Java doesn't provide freedom to programmer, to overload the standard arithmetic operators e.g. +, -, \* and / etc. If you have worked previously in C++, than you know that, Java has left lot of feature supported in C++ e.g. Java doesn't support multiple inheritance, no pointers in Java, and no pass by reference in Java. Rarely this question asked in Java interviews, to check how programmer thinks about certain features, which is not supported in Java. Another similar questions is regarding Java being pass by reference, which is mostly appear as, whether Java is pass by value or reference. Though I don't know the real reason behind it, I think following observation make sense on, why Operator overloading is not supported in Java.

**1) Simplicity and Cleanliness**

Why operator overloading is not supported in JavaSimple and clear design was one of the goals of Java designers. They, just don't want to replicate the language, but wanted to have a clear, truly object oriented language. Adding Operator overloading would have definitely made design more complex than without it, and it might have lead to more complex compiler or slows the JVM ,because it needs to do extra work to identify the actual meaning of operators, and reduce the opportunity to optimize the language by guarantee behavior of operators in Java.

**2) Avoid Programming Errors**

Java doesn't allow user defined operator overloading, because if you allow programmer to do operator overloading, they will come up with multiple meanings for same operator, which will make the learning curve of any developer hard and things more confusing and messy. Its been observed that, there is increase in programming errors, when language supports operator overloading, which in turn increase e development and delivery time. Since Java and JVM has taken most of developers responsibility, in memory management by providing garbage collector, it doesn't really make sense to left this feature to pollute the code, and as a loop hole for programming errors.

**3) JVM Complexity**

From JVM perspective, supporting operator overloading is more difficult, and if the same thing can be achieved, by using method overloading in more intuitive and clean way, it does make sense to not support operator overloading in Java. A complex JVM, may result in slower JVM, than a relatively simpler JVM ,and reduce the opportunity of optimization by taking out guaranteed behavior of operators in Java.

**4) Easy Development of Tools**

This is an additional benefit of not supporting operator overloading in Java. Omission of operator overloading has kept the language easier to handle and process, which in turn makes it easier to develop the tools, that process the language e.g. IDE or re-factoring tool. Re-factoring tools in Java are far better than C++.

In conclusion many things, which can be achieved by operator overloading, can also be achieved using method overloading using more intuitive and easy way and that might be the reason java designer thought that supporting operator overloading will not be a big benefit for language, but in fact only Java designer can answer real motivation of, why Java doesn't support operator overloading, like some other questions as Why Java doesn't support multiple inheritance or Why String is immutable in Java.