Java doesn't allow operator overloading yet, + is overloaded for class String.

Important points

1. When you add a non string operand such as an integer or char to a String, the non-string operand is converted to a string and string concatenation happens.
2. If both operands are char literals, the + operator performs addition rather than string concatenation by promoting each of the char-valued operands to int values through widening primitive conversion.

Let us consider some possibilities through examples.

Examples

Example: + operator overloading precedence check

```
public static void main(String[] args) {
      final String first = "length: 10";
      final String second = "length: " + first.length();
      System.out.println("first and second are equal: " + first == second);
   }
```

- It prints false and nothing else.
- The + operator, whether used for addition or string concatenation, has more precedence than the == operator. Therefore, the parameter of the println method is evaluated like:
    - System.out.println(("first and second are equal: " + first) == second);

Example: String concatenation with short hand operator

Find output:

```
String str = "420";

str += 42;

System.out.print(str);
```

- We can expand str += 42 as str=str+42. When you add a non string operand such as an integer or char to a String, the non-string operand is converted to a string and string concatenation happens. Therefore “42” is concatenated to the “420” giving "42042".

Example: String concatenation and characters

```
public static void main(String args[]) {
        System.out.println('a');
        System.out.println('a' + 'b');
        System.out.println("H" + 'a' + 'b');
        System.out.println('a' + 'b'+"H");
        System.out.println('a' + 'b'+"H" + 'a' + 'b');
}
```

This will print:

a

195

Hab

195H

195Hab

- **Line 1:** System.out.println('a');
    - System.out.println() is overloaded for character and it prints the character it represents and hence it **prints a here**.
- **Line 2 :** System.out.println('a' + 'b');
    - Here both operands are char literals. Because neither operand is of type String, the + operator performs addition rather than string concatenation. The compiler evaluates the constant expression 'a' + 'b' by promoting each of the char-valued operands ('a' and 'b') to int values through widening primitive conversion. Widening primitive conversion of a char to an int zero extends the 16-bit char value to fill the 32-bit int. In the case of 'a', the char value is 97 and in the case of 'b', it is 9b, so the expression 'a' + 'b' is equivalent to the int constant 97 + 98, or **195**.
- **Line 3:** System.out.println("H" + 'a' + 'b');
    - Java doesn't allow operator overloading yet + is overloaded for class String. When you add a String to an integer or char it is converted to a string and hence string concatenation happens. So output is **Hab**.
- **Line 4:** System.out.println('a' + 'b'+"H");
    - Java evaluates operands from left. So it adds ‘a’ and ‘b’ as string literals and then concatenates the result to the string “H” to get **195H.**
- **Line 5:** System.out.println('a' + 'b'+"H" + 'a' + 'b');}
    - Java evaluates operands from left. So it adds ‘a’ and ‘b’ as string literals and then concatenates the result to the string “H” to get 195H. Now this string 195H is added to ‘a’ and ‘b’, which will be string concatenation to get **195Hab**.