```cpp
#include "FibonacciSequenceGenerator.h"
#include <stdexcept>
#include <climits>


FibonacciSequenceGenerator::FibonacciSequenceGenerator(const std::string& aID) noexcept : fID(aID), fPrevious(0), fCurrent(1) {
    //Construct the FibonacciSequenceGenerator
}

const std::string& FibonacciSequenceGenerator::id() const noexcept {
    //Get the ID of the FibonacciSequenceGenerator
    return this->fID;
}

const long long& FibonacciSequenceGenerator::operator*() const noexcept {
    //Get the current value, which will be use as the main value of FibonacciSequenceGenerator (using operator*())
    return this->fCurrent;
}

FibonacciSequenceGenerator::operator bool() const noexcept {
    //Return true if there are any next available number (not reach limit of long long). Depends on hasNext() function
    return this->hasNext();
}

void FibonacciSequenceGenerator::reset() noexcept {
    //Reset the previous and current to 0 and 1 consecutively
    this->fPrevious = 0;
    this->fCurrent = 1;
}

bool FibonacciSequenceGenerator::hasNext() const noexcept {
    //Check that current value must greater than 0 and lower than the long long limit (which is 2^64 - 1)
    return this->fCurrent >= 0 && this -> fPrevious <= (LLONG_MAX - this->fCurrent);
}

void FibonacciSequenceGenerator::next() noexcept {
    //If value is greater than limit of long long -> Raise overflow_error
    if (!this->hasNext())
        throw std::overflow_error("Fibonacci sequence overflow");
    else {
        //Set a temporary value as previous + current
        long long temporary = this->fCurrent + this->fPrevious;
        //Previous be the current value
        this->fPrevious = this->fCurrent;
```

```
45          //Current value be the temporary value
46          this->fCurrent = temporary;
47
48      }
49  }
```