Mandatory Copy Elision → There is no copy, no need to have copy/move constructor defined.

Named Return Value Optimization → Compiler dependent, generally clang is better for now

auto A = myClass{MyClass{MyClass{}}}; → No copy constructor is called
→ Only 1 constructor/destructor called

MyClass foo(int x){ Return myClass{x};} }
auto m = foo(123)

↳ This enable us to write factory function for objects without copy/move constructor

template <typename T, typename ...Args>
auto create(Args&& ...args){ return T(std::forward<Args>(args)...);
Always work, even

Move constructor must be noexcept.

When local variable is returned (and not NRVO) it is automatically moved.

If return value is ternary op, then no copy elision or move happens, copy
→ with returns of named values
constructor is called.

If base class is noexcept, then derived must be noexcept,
" " " " not noexcept, " " can be either case

All $\overset{\text{default}}{\text{special}}$ member functions are noexcept; but it is smart enough to consider member variable noexcept status, if it has string then copy members can throw.

Destructors are (should) be noexcept

Exception level → All functions should at least provide basic guarantee

Basic Guarantee: If function throws, it should be in valid state and does not leak resources

Stron Guarantee: Basic + No change in program state, if cant finish job goes back to state before starting task

Noexcept Guarantee: Does not ~~t~~throw, if noexcept function throws, program is terminated.

Duration types (used for benchmark/timing)

↳ With C++20 there is insertion functions (<<), so can be printed, without using .count()