Return Value Optimization

RVO

- Return Value optimization is used by compilers for optimizing away temporary copies of objects.
- This is typically enabled in *optimized* mode of a compiler.
- You should check if your compiler supports this (hopefully should by now).
- The standard says that the compiler can do optimizations as long as the behavior of the generated code is same.
- However, for return value optimization, the standard allows a compiler to optimize away a copy constructor call that would have been made as a result of a return from a function, even if the copy constructor has some side effects.

Unnamed RVO

- This is when unnamed temporaries are involved.

```
ABC  Foo ( )
{
            // … some code
            return ABC ();  // or  return  ABC( arg1, arg2 );
}
```

In the above example, a temporary object is created at the end and returned.

Unnamed RVO

- If there is no RVO applied, then that object is returned by value (gets copied to the receiving object, resulting in a copy constructor call).

```
void Func()
{
        ABC    obj1( Foo() );
}
```

With RVO, the copy constructor is not invoked for creating obj1.

Named RVO

- This is when RVO is applied to named variables.

```
ABC  Foo ( )
{
        ABC   a1;
        // … some code
        return  a1;
}
```

- In this example, a1 is the named variable.

```
void Func()
{
        ABC    obj1( Foo() );
}
```

With named RVO, the copy constructor is not invoked for creating obj1.

Mixing …

- If your function has return statements with named and unnamed temporaries, RVO may not be applied by the compiler.

```
ABC MixedRVO(int arg)
{
        if (arg < 0)
                return ABC();
        else
        {
                ABC a1;

                return a1;
        }
}
```

move constructor?

- If a move constructor is defined for the class (C++ 11), then that will be used instead of RVO.