```
1  ///////// Lecture 69: Type Conversions III (User to
   •   Primitive Type)
2
3  // If x is a primitive type, and a1 is an integer
   •   class and if you do a standard assignment operator,
   •   this will NOT compile. The compiler does not know
   •   how to convert this object into the primitive type,
4  int main(){
5      Integer a1 {5};
6      int x = a1;  // Compiler Error: No suitable
   •       conversion function from "Integer" to "int"
   •       exists.
7  }
8  // Thankfully, C++ provides us with functionality
   •   through which you can implement a type conversion
   •   operator in the Integer class and that tyype
   •   conversion operator will convert this object into a
   •   primitive type.
9
10 // See slides for syntax.
11
12 // Type Conversion Operator
13
14 // First edit your object class: Integer.h
15 class Integer{
16     ...
17 public:
18     ...
19     operator int();
20 };
21
22 // Next edit your Integer.cpp implementation:
23 Integer::operator int(){
24     return *m_pInt;
25 }
26
27 // Now you don't even need to change your code in
   •   main()
28 int main(){
29     Integer a1 {5};
30     int x = a1; // Now the compiler can use the type
   •       conversion operator function to convert the
```

```
        Integer object into a primitive type. Note that
        the compiler IMPLICITLY invokes the operator
        function for type conversion.
    // Although you can write it like this,
    int x = static_cast<int>(a1); // it is redundant.
    // Here, we are performing an EXPLICIT CAST, so we
        can remove the static cast, it's not required,
        because the compiler implicitly invokes the type
        conversion operator function.
}
// This can cause confusion in some code, therefore
    C++11 allows us to use the explicit keyword on the
    operator function, so that the compiler can never
    use implicit conversion. So if you want to perform a
    type conversion, you'll have to explicitly mention
    the cast. in this case, use static_cast.

// So using the type conversion operator, you can
    convert any user defined type into any other type.

```