A copy constructor may be called when:

1. An object of the class is returned by value:

   When a function returns an object by value, a copy of the object is created for the caller. The copy constructor is invoked to initialize this new object with the properties of the object being returned.

```
class MyClass {
public:
    int data;
    // Constructor
    MyClass(int value) : data(value) {}
    // Copy constructor
    MyClass(const MyClass& source) {
        data = source.data;
        std::cout << "Copy constructor called\n";
    }
};

MyClass createObject(int value) {
    MyClass obj(value);
    return obj; // Copy constructor is called when returning by value
}

int main() {
    MyClass newObj = createObject(42);
    //createObject function is called true the function copy
    //constructor is called and than the object is returned
    return 0;
}
```

2. An object of the class is passed (to a function) by value as an argument:

   When an object is passed by value to a function, a copy of the object is created within the function's scope. Again, the copy constructor is called to initialize this copy.

```
void processObject(MyClass obj) {
    // Copy constructor is called when obj is passed by value
    // ...
}

int main() {
    MyClass original(42);
    processObject(original);
    // ...
    return 0;
}
```

3. An object is constructed based on another object of the same class:

   As you've correctly stated, when an object is created based on another object of the same class (using assignment or initialization) the copy constructor is called to perform the initialization.

```
MyClass obj1(42);
MyClass obj2 = obj1; // Copy constructor is called during initialization
```

4. The compiler generates a temporary object:

   In some cases, the C++ compiler may generate temporary objects, especially during expressions and evaluations. For example, in complex expressions or when passing function arguments, the compiler may create temporary objects to hold intermediate results. In such cases, the copy constructor is called to initialize these temporary objects.

```
void processObjects(MyClass obj1, MyClass obj2) {
    // obj1 and obj2 are initialized using the copy constructor
    // ...
}

int main() {
    processObjects(MyClass(10), MyClass(20));
    // Temporary objects are created during the function call
    // ...
    return 0;
}
```

Understanding when the copy constructor is called is crucial for proper resource management and ensuring that objects are correctly copied or moved when needed.