class25_singleton_handout

```
-------------------------------------------------------------------
singleton.cpp: MAke sure only ONE object instantiated at a time
-------------------------------------------------------------------

class singleton {
  public:
    static singleton *constructor();
    static void destructor();

  protected:
    singleton() { ; }
    ~singleton() { ; }

  private:
    static singleton *objptr;
    static int Nobjptr;
};

singleton *singleton::constructor() {
  if (objptr == NULL)
    objptr = new singleton;

  Nobjptr++;
  return objptr;
}

void singleton::destructor() {
  if (--Nobjptr == 0) {
    delete objptr;
    objptr = NULL;
  }
}

singleton *singleton::objptr = NULL;
int singleton::Nobjptr = 0;

int main() {
  singleton *ptr1, *ptr2;

  ptr1 = singleton::constructor();
  cout << "ptr1 " << ptr1 << "\n";

  ptr2 = singleton::constructor();
  cout << "ptr2 " << ptr2 << "\n";

  ptr1->destructor();
  ptr2->destructor();
}
```

```
-------------------------------------------------------------------
Hint: Singleton constructor and destructor are protected meaning
they cannot be executed from outside the class.

Hint: Static class data members are shared among all objects of
that class.

Hint: First call to singleton::constructor() causes constructor
to be executed. Subsequent calls return address to same object.

Hint: Last call to singleton::destructor() causes destructor to
be executed. Prior calls do nothing as other objects depend on
the object remaining alive.
-------------------------------------------------------------------
```