Item5: Know what functions C++ silently writes and calls

Class Empty{};

🡺

class Empty {

public:

Empty() { ... } // default constructor

Empty(const Empty& rhs) { ... } // copy constructor

~Empty() { ... } // destructor

Empty& operator=(const Empty& rhs) { ... } // copy assignment operator

};

Generated destructor is non-virtual unless inheriting from a base class that declares a virtual destructor.

Example:

template<typename T>

class NamedObject {

public:

NamedObject(const char \*name, const T& value);

NamedObject(const std::string& name, const T& value);

private:

std::string nameValue;

T objectValue;

};

🡺a constructor of NamedObject is declared, so compiler won’t generate a default constructor. NamedObject does not declare copy constructor or copy assignment operator, so compiler will generate these functions (if they are needed)

Ex. of copy constructor:

NamedObject<int> no1(“small”,2);

NamedObject<int> no2(no1); //calls copy constructor

🡺no2.nameValue will be initialized by calling string copy constructor with no1.nameValue

no2.objectValue will be initialized by copying the bits in no1.objectValue(built-in type)

Consider:

template<typename T>

class NamedObject {

public:

NamedObject(std::string& name, const T& value);

... // as above, assume no operator= is declared

private:

std::string& nameValue; // this is now a reference

const T objectValue; // this is now const

};

std::string newDog("Persephone");

std::string oldDog("Satch");

NamedObject<int> p(newDog, 2);

NamedObject<int> s(oldDog, 36);

p = s;

🡺What should happen to the data members in p?

Should the string object to which p.nameValue refers be modified, thus

affecting other objects that hold pointers or references to that string?

🡺C++ refuses to compile the code. If you want to support copy assignment in a class containing a reference member, you must define the copy assignment operator yourself.

Conclusion: Compilers may implicitly generate a class’s default constructor, copy

constructor, copy assignment operator, and destructor.