Containment

```
#include <iostream>
#include <fstream>
#include <vector>
using namespace std;

class Person {
    friend ostream& operator<<(ostream& os, const Person& someone) {
        os << "Person: " << someone.name << ", " << someone.age;
        return os;
    }
public:
    Person(const string& name, int age) : name(name), age(age) {}
    const string& getName() const { return name; }
private:
    string name;
    int age;
};

int main() {
    ifstream ifs("stooges.txt");
    vector<Person> group1;
    string name;
    int age;
    while (ifs >> name >> age) {
        Person someone(name, age);
        group1.push_back(someone);
    }
    for (Person& p : group1) {
        cout << p.getName() << endl;
    }

    for (Person& p : group1) {
        cout << p << endl;
    }
}
```

```
Moe
Larry
Curly
Shemp
Person: Moe, 77
Person: Larry, 72
Person: Curly, 48
Person: Shemp, 60
```

- We created a vector of “Person” ← **containment**
- **But what about if** we want:
  - Group 1 to contain all persons
  - Group 2 contains persons at or below age 62
  - Group 3 contains persons above age 62
- Then we must use **association** (pointers) instead of containment..

Association

```
#include <iostream>
#include <fstream>
#include <vector>
using namespace std;

class Person {
    friend ostream& operator<<(ostream& os, const Person& someone) {
        os << "Person: " << someone.name << ", " << someone.age;
        return os;
    }
public:
    Person(const string& name, int age) : name(name), age(age) {}
    const string& getName() const { return name; }
private:
    string name;
    int age;
};

int main() {
    ifstream ifs("stooges.txt");
    vector<Person*> group1;
    string name;
    int age;

    while (ifs >> name >> age) {
        Person someone(name, age);
        group1.push_back(&someone);
    }

    for (Person* p : group1) {
        cout << p->getName() << endl;
    }

    for (Person* p : group1) {
        cout << *p << endl;
    }
}
```

Stooges.txt:

Moe 77
Larry 72
Curly 48
Shemp 60

- What will be the output?

Association

```
#include <iostream>
#include <fstream>
#include <vector>
using namespace std;

class Person {
    friend ostream& operator<<(ostream& os, const Person& someone) {
        os << "Person: " << someone.name << ", " << someone.age;
        return os;
    }
public:
    Person(const string& name, int age) : name(name), age(age) {}
    const string& getName() const { return name; }
private:
    string name;
    int age;
};

int main() {
    ifstream ifs("stooges.txt");
    vector<Person*> group1;
    string name;
    int age;

    while (ifs >> name >> age) {
        Person someone(name, age);
        group1.push_back(&someone);
    }

    for (Person* p : group1) {
        cout << p->getName() << endl;
    }

    for (Person* p : group1) {
        cout << *p << endl;
    }
}
```

```
Person: , 60
Person: , 60
Person: , 60
Person: , 60
```

Stooges.txt:

Moe 77
Larry 72
Curly 48
Shemp 60

- All names are empty strings!
- All ages are 60!!

Association

```cpp
#include <iostream>
#include <fstream>
#include <vector>
using namespace std;

class Person {
    friend ostream& operator<<(ostream& os, const Person& someone) {
        os << "Person: " << someone.name << ", " << someone.age;
        return os;
    }
public:
    Person(const string& name, int age) : name(name), age(age) {}
    const string& getName() const { return name; }
private:
    string name;
    int age;
};

int main() {
    ifstream ifs("stooges.txt");
    vector<Person*> group1;
    string name;
    int age;

    while (ifs >> name >> age) {
        Person someone(name, age);
        group1.push_back(&someone);
    }

    for (Person* p : group1) {
        cout << p->getName() << endl;
    }

    for (Person* p : group1) {
        cout << p << ": " << *p << endl;
    }
}
```

```
0000005D3D8FF398: Person: , 60
0000005D3D8FF398: Person: , 60
0000005D3D8FF398: Person: , 60
0000005D3D8FF398: Person: , 60
```

Stooges.txt:

Moe 77
Larry 72
Curly 48
Shemp 60

- All names are empty strings!
- All ages are 60!!

Association

```cpp
#include <iostream>
#include <fstream>
#include <vector>
using namespace std;

class Person {
    friend ostream& operator<<(ostream& os, const Person& someone) {
        os << "Person: " << someone.name << ", " << someone.age;
        return os;
    }
public:
    Person(const string& name, int age) : name(name), age(age) {}
    void setname(string in) { name = in; }
    void setage(int in) { age = in; }
    const string& getName() const { return name; }
private:
    string name;
    int age;
};

int main() {
    ifstream ifs("stooges.txt");
    vector<Person*> group1;
    string name;
    int age;
    Person someone("Unnamed", 5);
    while (ifs >> name >> age) {
        someone.setname(name); someone.setage(age);
        group1.push_back(&someone);
    }
    for (Person* p : group1) {
        // All the group1 are showing the same name.
        cout << p->getName() << endl;
    }

    for (Person* p : group1) {
        // And they are all displaying the same address!!!
        cout << p << ": " << *p << endl;
    }
}
```

Association

```cpp
#include <iostream>
#include <fstream>
#include <vector>
using namespace std;

class Person {
    friend ostream& operator<<(ostream& os, const Person& someone) {
        os << "Person: " << someone.name << ", " << someone.age;
        return os;
    }
public:
    Person(const string& name, int age) : name(name), age(age) {}
    void setname(string in) { name = in; }
    void setage(int in) { age = in; }
    const string& getName() const { return name; }
private:
    string name;
    int age;
};

int main() {
    ifstream ifs("stooges.txt");
    vector<Person*> group1;
    string name;
    int age;
    Person someone("Unnamed", 5);
    while (ifs >> name >> age) {
        someone.setname(name); someone.setage(age);
        group1.push_back(&someone);
    }
    for (Person* p : group1) {
        // All the group1 are showing the same name.
        cout << p->getName() << endl;
    }

    for (Person* p : group1) {
        // And they are all displaying the same address!!!
        cout << p << ": " << *p << endl;
    }
}
```

```
Shemp
Shemp
Shemp
Shemp
0000005AF251F928: Person: Shemp, 60
0000005AF251F928: Person: Shemp, 60
0000005AF251F928: Person: Shemp, 60
0000005AF251F928: Person: Shemp, 60
```

Stooges.txt:

Moe 77
Larry 72
Curly 48
Shemp 60

- Every class has a destructor function that is called when it's deallocated
- By creating "someone" outside of the while loop, it is not deallocated at the end of that while's code block, but rather persists till end of the main routine.