Chapter 14 (Collection object)
Chapter 15 (Singleton)

Collection object (Ch 14)

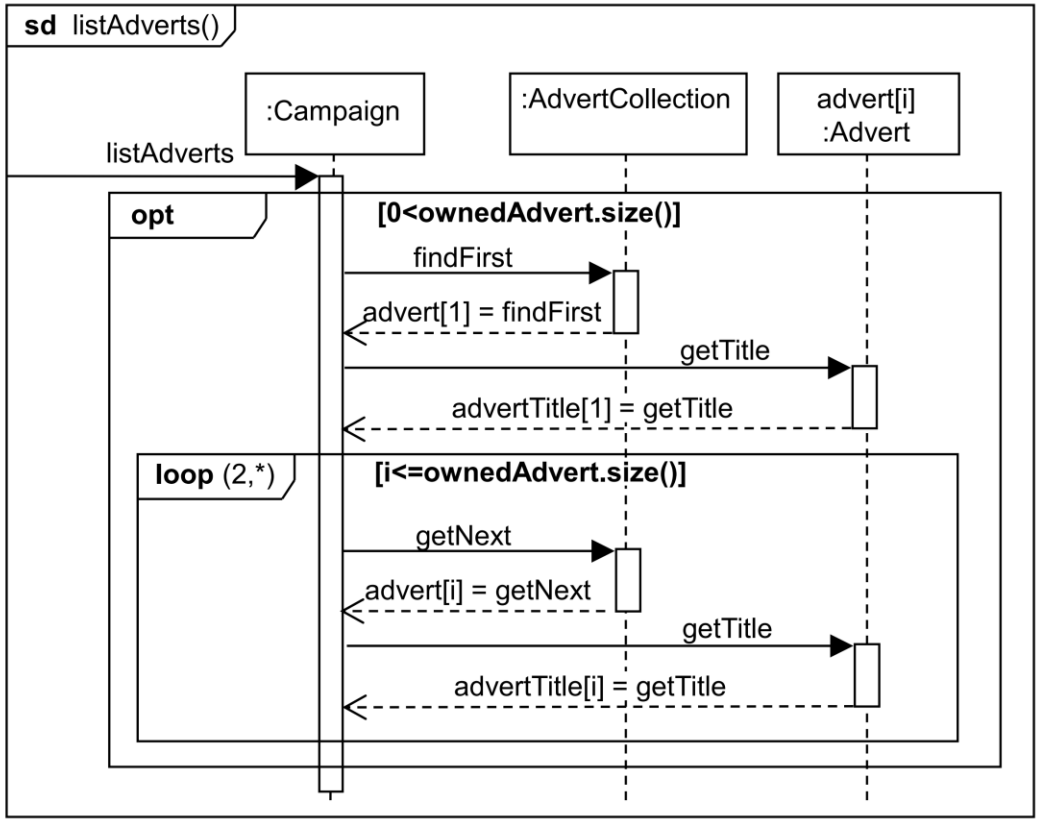
Collection Classes

- Collection classes are used to separately hold object identifiers when message passing is required from one to many along an association
- OO languages provide support for these requirements. Frequently the collection class may be implemented as part of the sending class (e.g. `Campaign`) as some form of list

Collection object (Ch 14)

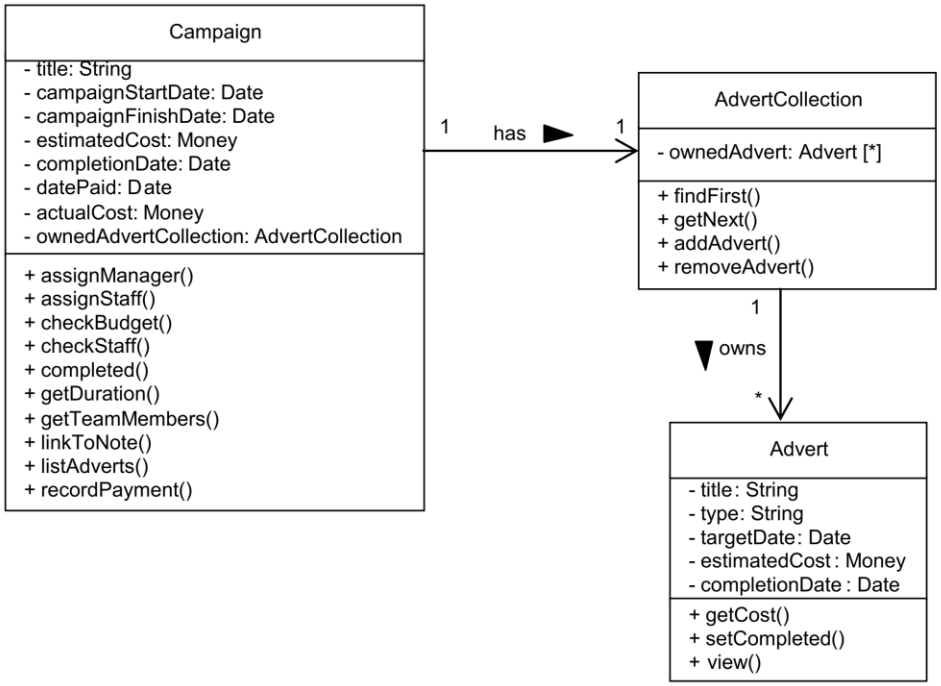
Sequence diagram for `listAdverts()`

This sequence diagram shows the interaction when using a collection class

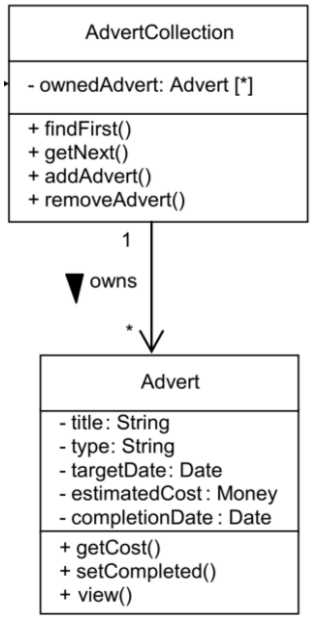


Collection object (Ch 14)

One-to-many associations using a collection class



Collection object (Ch 14)

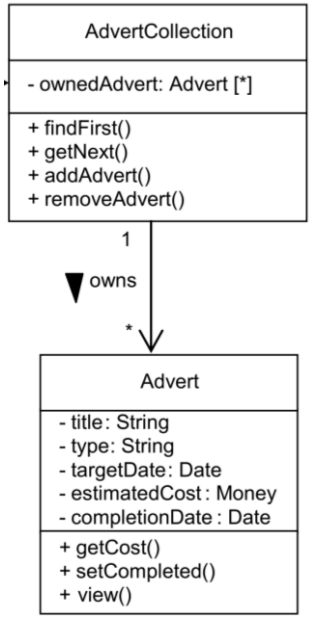


Advert.h

akeCollectionExample - x64-Debug (de (Global Scope)

```
1  #ifndef __ADVERT_H__
2  #define __ADVERT_H__
3
4  #include <string>
5
6  using namespace std;
7
8  class Advert {
9  private:
10     string _title;
11 public:
12     Advert();
13     string getTitle();
14     void setTitle(string title);
15 };
16
17 #endif
```

Collection object (Ch 14)

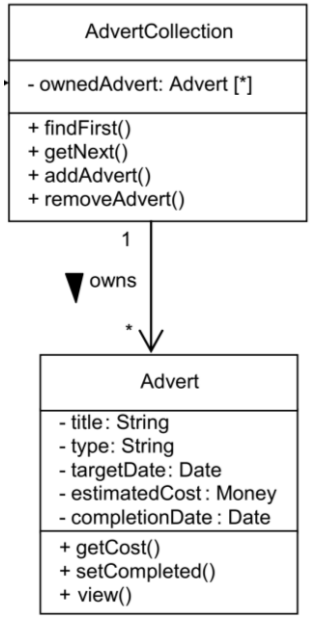


Advert.cpp | Advert

akeCollectionExample - x64-Debug (de | (Global Scope)

```
1   #include "Advert.h"
2
3   Advert::Advert()
4   {
5       _title = "";
6   }
7
8   string Advert::getTitle()
9   {
10      return _title;
11  }
12
13  void Advert::setTitle(string title)
14  {
15      _title = title;
16  }
17
```

Collection object (Ch 14)

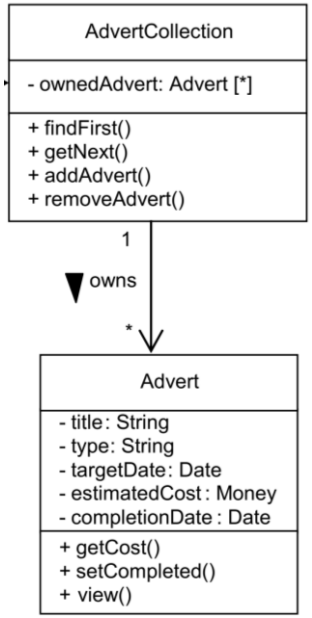


AdvertCollection.h

CMakeCollectionExample - x64-Debug (de | (Global Scope)

```
1  #ifndef __ADVERT_COLLECTION_H__
2  #define __ADVERT_COLLECTION_H__
3
4  #include "Advert.h"
5  #include <vector>
6
7  class AdvertCollection {
8  private:
9      vector<Advert *> _ownedAdvert;
10
11 public:
12     vector<Advert*>::size_type getSize();
13
14     Advert *findFirst();
15     Advert* get(vector<Advert*>::size_type index);
16
17     void addAdvert(string title);
18     //bool removeAdvert(string title);
19 };
20
21 #endif
```

Collection object (Ch 14)

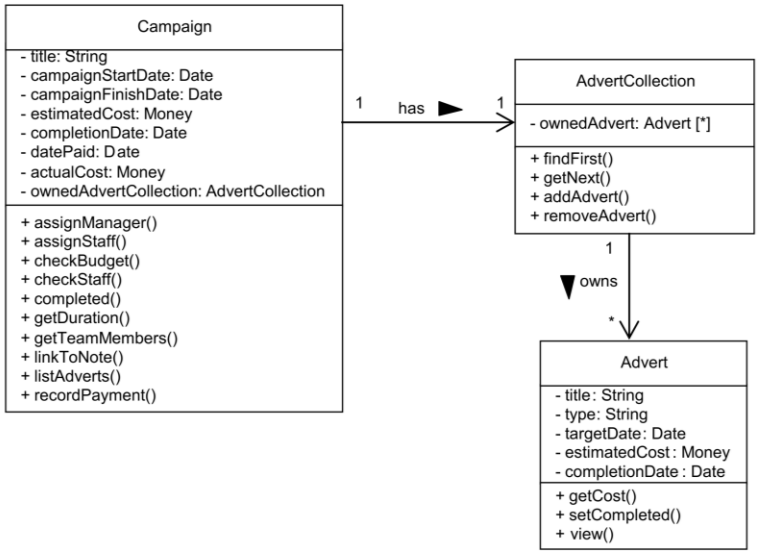


Collection.h | AdvertCollection.cpp | CMakeCollectionExample.h | CMakeCollectionExample.cpp

akeCollectionExample - x64-Debug (de | (Global Scope)

```
1   #include "AdvertCollection.h"
2
3   vector<Advert*>::size_type AdvertCollection::getSize() {
4       return _ownedAdvert.size();
5   }
6
7   Advert* AdvertCollection::get(vector<Advert*>::size_type index) {
8       return _ownedAdvert.at(index);
9   }
10
11  Advert* AdvertCollection::findFirst() {
12      return _ownedAdvert.at(0);
13  }
14
15  void AdvertCollection::addAdvert(string title) {
16      Advert *newAdvert = new Advert();
17      newAdvert->setTitle(title);
18      _ownedAdvert.push_back(newAdvert);
19  }
20
21  //bool AdvertCollection::removeAdvert(string title);
22
```

Collection object (Ch 14)

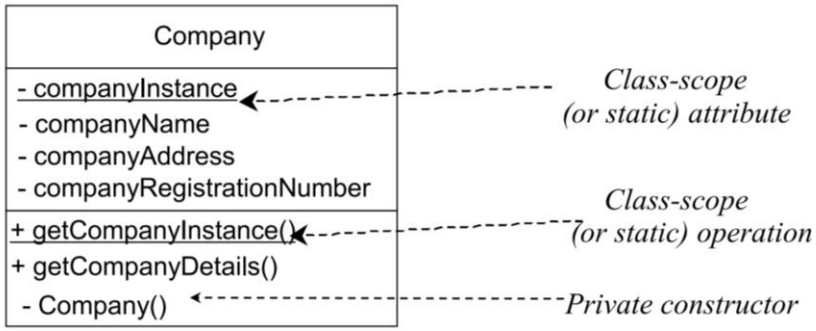



```cpp
#include <iostream>
#include <string>
#include "CMakeCollectionExample.h"
#include "Advert.h"
#include "AdvertCollection.h"

using namespace std;

int main()
{
    AdvertCollection ownedAdvertCollection;
    ownedAdvertCollection.addAdvert("title1");
    ownedAdvertCollection.addAdvert("title2");
    ownedAdvertCollection.addAdvert("title3");
    ownedAdvertCollection.addAdvert("title3-2");

    if (ownedAdvertCollection.getSize() > 0) {
        Advert *a = ownedAdvertCollection.findFirst();
        cout << a->getTitle() << "\n" << endl;

        vector<Advert*>::size_type i;
        for (i = 1; i < ownedAdvertCollection.getSize(); i++) {
            a = ownedAdvertCollection.get(i);
            cout << a->getTitle() << "\n" << endl;
        }
    }

    cin.get();
    return 0;
}
```

```
title1

title2

title3

title3-2
```

Singleton (Ch 15)

```cpp
#include <iostream>

class Company {
    private:
        static Company* companyInstance;
        int companyRegistrationNumber;

    public:
        static Company* getCompanyInstance();
        int getCompanyRegistrationNumber();

    private:
        Company();
};


Company* Company::companyInstance = NULL;
Company* Company::getCompanyInstance() {
    if (companyInstance == NULL) {
        companyInstance = new Company();
    }
    return companyInstance;
}
int Company::getCompanyRegistrationNumber() {
    return companyRegistrationNumber;
}
Company::Company() {
    std::cout << "Company object is created.\n";
    companyRegistrationNumber = 100;
}


int main(void) {
    Company *a = Company::getCompanyInstance();
    std::cout << "a's number is " << a->getCompanyRegistrationNumber() << "\n";
    Company *b = Company::getCompanyInstance();
    std::cout << "b's number is " << b->getCompanyRegistrationNumber() << "\n";
}
```



The use of class-scope operations allows global access