CSE1720

Week 05, **Class Meeting 14** (Lecture 10)

Winter 2013 ◆ Thursday, Feb 06, 2013

YORK UNIVERSITÉ UNIVERSITY

Goals for today's class meeting

Provide feedback and enable discussion about the questions at the end of Chapter 8

YORK UNIVERSITÉ UNIVERSITY

Questions about Collections

- What is a collection? What is an aggregate with variable multiplicity? How are these questions related?
- RQ8.19 What does variable multiplicity mean, and how is it depicted in UML?
- RQ8.20 If a collection is statically allocated, then what should be passed to its constructor? If a collection is statically allocated, then what should be passed to its constructor?
- RQ8.21 Can you add an element to a collection even if it is already in it?

YORK UNIVERSITÉ UNIVERSITY

Questions about Collections

- RQ8.22 What happens if you attempt to add an element to a full, statically allocated collection?
- RQ8.23 What is a traversal?
- RQ8.24 How do you determine the number of elements in a collection if it supports indexed traversals?
- RQ8.25 How do you determine the number of elements in a collection if it supports iterator-based traversals?
- RQ8.26 (a) Explain how a traversal can be used to perform a search. (b) Why are traversal-based searches called exhaustive?

YORK UNIVERSITÉ UNIVERSITY

OK – those are many questions.

Let’s talk about some answers

The first question… What is a collection?

YORK UNIVERSITÉ UNIVERSITY

About Collections…

The course material concerns several topics about collections: e.g., collection traversals, static/dynamic allocation, etc.

These concepts will make a lot **more sense** if you have a crystal clear understanding about what a collection actually is

YORK UNIVERSITÉ UNIVERSITY

So what **is** a collection anyway?

Let's start with what a collection is **NOT**.

A collection is **NOT** a set.

- A set is, by definition, a **collection** that does not contain duplicate elements.

A collection is **NOT** a list.

- A list is, by definition, an ordered **collection**.

You can't use the term you are trying to define in the definition!

7

YORK UNIVERSITÉ UNIVERSITY

So what **is** a collection anyway?

Instead of trying to articulate what a collection **IS** it is better to articulate what a collection **DOES**

This is a Forrest Gump was of defining something:

A collection *is* what a collection *does*

8

YORK UNIVERSITÉ UNIVERSITY

So what does a collection **do**?

A collection does:

- have elements.
  - and these elements are understood to be non-primitive
- allow clients to query its size
- allow clients add/remove elements
- allow clients traverse the elements
  - two possible ways this can be provided

YORK UNIVERSITÉ UNIVERSITY

A diagnostic test:
Is this object a collection?

Does it have elements that I can traverse?

Does it let me add elements?

Does it let me remove elements?

Does it tell me its size?

Then it is a collection.*

*a collection does a few other things, but we will talk about these later

YORK UNIVERSITÉ UNIVERSITY

The JBA way of defining a collection

A collection is an aggregate in which the multiplicity is variable and in which the aggregated parts are called elements.

YORK UNIVERSITÉ UNIVERSITY

ok….

… but how does this jive with the whole “a collection is what a collection does” way of constructing a definition???

YORK UNIVERSITÉ UNIVERSITY

Here's the bridge…

- At run time, a *collection* is usually encapsulated by an **object**
  - It is possible, in principle, for a **class** to encapsulate a collection:
    - the requisite characteristics --- the traversal of elements, querying of size, addition/removal of elements --- would be provided by static methods
  - but this happens rarely, if at all, in practice
- In order for the object to even exist, it must have been **instantiated**
- This **capacity to be instantiated** is provided by a service of a class definition, namely the constructor
- **If the class definition encapsulates a collection, then it MUST BE an aggregation.**

YORK UNIVERSITÉ UNIVERSITY

Why?

- Why must any class that encapsulates a collection be an aggregation?

YORK UNIVERSITÉ UNIVERSITY

Because…

- A class that **encapsulates a collection** will, by definition, represent the elements of the collection
- These elements will have a **type**
  - These elements will be represented using class **attributes**
  - The traversal of elements, querying of size, addition/ removal of elements will be provided by class **methods**
- The collection class requires a HAS-A relationship with the class that encapsulates the type of the elements.
- The number of elements may change, hence the “variable multiplicity”

YORK UNIVERSITÉ UNIVERSITY

What does “traverse” mean?

A traversal can be thought of as a trip that visits each element once and only once.

JBA, p.318

What this means:

- No element can be missed
- No element can be visited more than once.

YORK UNIVERSITÉ UNIVERSITY

What does “traverse” **NOT** mean?

That the elements will be visited in **any particular order**

- Even if you traverse a given collection several times, you should not assume that the elements will be visited in the same order.
- There is **no order** defined over the elements (even if the elements are things that you may commonly think of as having a “natural” order, such as numbers)

17

YORK UNIVERSITÉ UNIVERSITY

What does “traverse” **NOT** mean?

Traverse doesn’t mean you get to do “**partial trips**”

- e.g., visit “every other element” or “the first half of the elements” or any other trip that is anything other than the complete traversal of all the elements
- A collection simply is not defined to provide this.
- This is just a variant of trying to **impose a particular order** on the elements of the collection.

18

YORK UNIVERSITÉ UNIVERSITY

Iterator-Based Traversal 8.2.4

If `collection` is a variable that refers to a collection object, then the following enhanced for loop will be provided:

```
for (ElementType e : collection) {
        // visit element e
}
```

OK, but what is `ElementType`?

…a collection is an aggregate, which means, by definition *a class that has as one of its features an attribute that is non-primitive.*
What is this non-primitive type?
Don't know – let's just call it `ElementType` for the time being…

Iterator-Based Traversal 8.2.4

Another version of iterator-based traversal is…

```
while (collection.hasNext()) {
      ElementType e = collection.next();
}
```

It is equivalent to the enhanced for loop version…

(see the API of the `Iterable` interface)

Indexed Traversal 8.2.3

A sneaky bit of material that has the potential to confuse…

we just emphasized that traverse does not mean visiting the elements in any particular order… but…

Sometimes a collection may, *in addition to its requisite methods*, also support **an indexing scheme for its elements,** such as via this method:

```
public ElementType get(int index)
```


21

Indexed Traversal 8.2.3

If there is an indexing scheme, then we can implement full and even partial traversals…

```
for (int i = 0; i < collection.size(); i++) {
    ElementType e = collection.get(i);
}
```

Since collection is a collection, it must have a size() method

22

Where can I get me a collection?

- the `Portfolio` class implements a collection of `CreditCard` elements; use the static method `getRandom()` to get a randomly-populated collection
- the `Picture` class implements `getPixels()`, which returns an array of `Pixel[]`.
- An array is not a collection (technically speaking), but provides identical behaviours

Where can I get me a collection?

What if I want to create and populate my own collection:

1. use a constructor to create an empty collection
2. add the elements one by one

- its not possible* to create and populate a collection all in one step

*well, someone might argue against this: a collection may provide constructors that allow the client to specify the initial content of a collection by passing a reference to another collection; such constructors are for convenience only and are implemented using the two steps above anyway.

we will do this once we cover section 9.3.3, Generics

A design tradeoff

When a new, empty collection is created, a block of run-time memory will be allocated for this object.

How large should this block be?

- If the block is too small:
  - then the size will be quickly exceed. When this happens, a whole new empty collection will need to be created, using a larger block and all of the elements copied over.
- If the block is too large:
  - a significant amount of memory will sit empty and cannot be used for anything else

The extreme form of the **small block** version is to start with a block so small that is it not big enough to hold even a single element. Then the amount of memory that is used grows and shrinks as the collection grows and shrinks.