Department of Computer Science
Undergraduate Events

Events this week
Schlumberger Info Session
Date: Mon., Feb 8
Time: 5:30 pm
Location: HENN Rm 201

Finding a Summer Job or Internship Info Session
Date: Wed., Feb 10
Time: 12 pm
Location: X836

Masters of Digital Media Program Info Session
Date: Thurs., Feb 11
Time: 12:30 – 1:30 pm
Location: DMP 201

Reminder: Co-op Deadline
Date: Fri., Feb 12
Submit application to Fiona at Rm X241 by 4:30 pm

Olympics Opening Ceremonies
Date: Fri., Feb 12
Time: 6:00pm

Administrivia
• Lecture slides (day by day) are on the web:
  • http://www.cs.ubc.ca/~norm/211/2009W2
• Assignment #2 is due this week
  • Due Thursday February 11, 10:00pm
Where are we?

- We are looking at:
  - The Java Collections Framework

Java Collections Framework

Consists of 3 components:

- Interfaces
  - provide specifications for the behaviour of the collections
  - form inheritance hierarchies
- Implementations
  - provide specific structures that store the elements and relevant operations on those structures
  - each interface may have multiple implementations that differ only by which optional operations they implement and by the efficiency of the operations
- Algorithms
  - polymorphic algorithms that manipulate data stored in collections
  - are not members of any collection
Some Collection Interfaces

- **Iterable**
- **Collection**
- **List**
- **ArrayList**

Collection Interfaces

- The **Collection** interface specifies methods that are applicable to all collections (lists, sets and queues – more later).

- The **List** interface specifies methods that are particular to lists (e.g., the ability to add an element at a specific location in the list).
Iterable Objects

• The Iterable interface has a single method and is defined in java.lang as:
  
  ```java
  public interface Iterable<T>
  {
    // Returns an iterator over a set of elements of type T.
    Iterator<T> iterator();
  }
  ```

• Each iterable object can return an Iterator:
  • An iterator is an object that allows us to visit the items in a collection
  • This is another example of a generic type. The type T is a generic type that will be specified when the iterator is declared and instantiated.

The Iterator Interface

• Is defined in java.util as:
  
  ```java
  public interface Iterator<E> {
    boolean hasNext();
    E next();
    void remove(); // Optional
  }
  ```

• hasNext() returns true if there is another element to visit
• next() returns the next object in the collection and advances the iterator to another object that has not been visited
• remove() removes the object that was returned by the last next() operation
  • can be called only once per call to next()
  • otherwise IllegalStateException is thrown.
The **Iterator** Interface (cont'd)

- Some notes on the *optional* `remove()` method:
- Methods in an interface that are documented to be optional:
  - provide flexibility
  - allow for a reduction in the number of interfaces needed
  - **must** be implemented by classes that implement the interface although those implementations may do nothing more than throw an `UnsupportedOperationException`.

```java
public interface Collection<E> extends Iterable<E> {
    int size();
    boolean isEmpty();
    boolean contains(Object o);
    boolean add(E o); // Optional
    boolean remove(Object o); // Optional
    Iterator<E> iterator();
    // Bulk Operations
    boolean containsAll(Collection<?> c);
    // Array Operations
    Object[] toArray();
    <T> T[]toArray(T[] a);
    // Object operations; allow collections to customize
    boolean equals(Object o);
    int hashCode();
}
```
The **Collection** Interface (cont’d)

- Provides a general set of methods applicable to all collections
- Used as a base for more specific sub-interfaces (e.g. `List` and `Set`)
- **Note**: the `contains()` method uses `equals()` for comparison.

---

**Collection-Iterator - Example**

- Complete the following method that prints out all the elements in a collection of strings:

```java
public static void print( Collection<String> col )
{
    Iterator<String> itr = col.iterator();
    while ( itr.hasNext() )
    {
        String s = itr.next();
        System.out.println(s);
    }
}
```
Collection-Iterator – Generic Example

• Complete the following method that prints out all the elements in a collection of any type:

```java
public static <T> void print( Collection<T> col )
{
    Iterator<T> itr = col.iterator();
    while ( itr.hasNext() )
    {
        System.out.println( itr.next() );
    }
}
```