Data Structures

W3134

Administration

- five assignments, 10 points each
  - lowest dropped, 50% points added as extra credit
  - due 9:00am on date posted, one point off per hour late
- three quizzes, 12 points each
- one final exam, 24 points
- details, syllabus, and assignments on courseworks

Abstract Data Types

- model of a class of objects
- description of the operations on these objects
- implementation independent

Data Structures

- specific mechanism for storing a class of objects
- implementation of the operations that act on those objects

Algorithms

- method for computing a function
- composed of a finite list of operations
- efficiency measured by time and space complexity

Java Representation of Abstract Data Types

- interface construct
- list of operations that can be performed on an object in such a class
- method signatures
- abstract class construct partially implements the abstract data type without commitment to underlying data structure

Comparable and Iterator Interfaces

- comparable interface specifies the requirement to implement

  public int compareTo(Object obj);

- iterator interface specifies the requirement to implement

  public boolean hasNext ();
  public Object next ();

```
public interface Set {
    public int size ();
    public boolean isEmpty ();
    public boolean isMember (Object e);
    public Set union (Set that);
    public Set intersection (Set that);
    public Set copy ();
    public void add (Object e);
    public void remove (Object e);
    public Iterator iterator ();
    public Set empty ();
}
```

```
public abstract class AbstractSet implements Set {
.
.
.
    public int size () {
        int count = 0;
        Object o;
        for (Iterator i = this.iterator();
             i.hasNext();
             o = i.next())
            count++;
        return count;
    }
.
.
.
}
```

```
public abstract class AbstractSet implements Set {
.
.
.
    public boolean isEmpty () {
        return this.size() == 0;
    }
.
.
.
}
```

```
public abstract class AbstractSet implements Set {
.
.
.
    public boolean isMember (Object o) {
       for (Iterator i = this.iterator(); i.hasNext();)
           if (o.equals(i.next())) return true;
      return false;
    }
.
.
.
}
```

```
public abstract class AbstractSet implements Set {
.
.
.
    public Set union (Set that) {
        Set result = this.copy();
        for (Iterator i = that.iterator(); i.hasNext();)
            result.add(i.next());
        return result;
    }
.
.
.
}
```

```
public abstract class AbstractSet implements Set {
.
.
.
    public Set intersection (Set that) {
        Set result = this.empty();
        Object o;
        for (Iterator i = that.iterator();
             i.hasNext();
             o = i.next())
            if (this.isMember(o)) result.add(o);
       return result;
    }
.
.
.
}
```

```
public abstract class AbstractSet implements Set {
.
.
.
    public Set copy () {
        Set result = this.empty();
        Object o;
        for (Iterator i = that.iterator();
             i.hasNext();
             o = i.next())
            result.add(o);
       return result;
    }
.
.
.
}
```

```
public abstract class AbstractSet implements Set {
.
.
.
    public String toString () {
       String result = "{";
       for (Iterator i = this.iterator();
            i.hasNext();
            result += " " + i.next());
       return result + " }";
    }
.
.
.
}
```

```
public abstract class AbstractSet implements Set {
.
.
.

    public abstract void add (Object o);
    public abstract void remove (Object o);
    public abstract Iterator iterator ();
    public abstract Set empty ();

.
.
.
}
```