Chapter 14

A List ADT
Review the Java interface construct

- A Java interface describes a set of methods:
  - no constructors
  - no instance variables

- interface OurList<E> will be implemented by two different classes using different data structures

OurArrayList<E> implements OurList<E>  ch. 14
OurLinkedList<E> implements OurList<E>  ch. 15
An ADT specified as an interface

/**
 * An interface specifying methods for a generic List ADT.
 * Its 7 methods are a subset of the 25 specified in List<E>.
 */

public interface OurList<E> {

  // Return the number of elements currently in the list
  public int size();

  // Insert an element at the specified location
  // Precondition: insertIndex >= 0 && insertIndex <= size()
  public void add(int insertIndex, E element)
      throws IllegalArgumentException;

  // Get the element stored at a specific index
  // Precondition: insertIndex >= 0 and insertIndex < size()
  public E get(int getIndex) throws IllegalArgumentException;

  // Other methods in Ch 14: set, find, remove, removeElementAt
}
Implement 2 methods at once

• Begin by adding and getting elements

```java
@Test
public void testAddWithGet() {
    OurList<String> list = new OurArrayList<String>();
    list.add(0, "A");
    list.add(1, "B");
    list.add(2, "C");
    assertEquals("A", list.get(0));
    assertEquals("B", list.get(1));
    assertEquals("C", list.get(2));
}
```
Inserting elements almost anywhere

- **We must keep elements together**
  - The 1st must go at index 0
  - The 2nd element can be at index 0 or 1
  - The 3rd at index 0, 1, or 2 only

  \[
  \text{data} \rightarrow |"A"|"B"|"C"|\text{null}|\text{null}|\text{null}|...|\text{null}|
  \]

- **We must shift array elements:** \texttt{add(0, "D")}

  \[
  \text{data} \rightarrow |"D"|"A"|"B"|"C"|\text{null}|\text{null}|...|\text{null}|
  \]

- **We can not leave gaps:** \texttt{add(6, "No go")}
  - **This must throw** \texttt{IllegalArgumentException}
Exceptions

- When programs run, exceptional events occur
  - division by zero
  - invalid input
  - `next()` when a `Scanner` has no more data
  - open a file that does not exist
  - add an element where you shouldn't

- Add and get throw `IllegalArgumentException`
  - Next slide shows when and how to do that, and
  - a test to ensure `get(0)` from an empty list throws it
Handling exceptional events

- Put code that "throws" exceptional event into a try block and add a catch block

```java
try {
    code that cause an exceptional event (throws an exception)
}
catch(Exception anException) {
    code that executes when code in try above throws an exception
}
```

- If code in the try block causes an exceptional event, program control transfers to the catch block
Example

- `Double.parseDouble` may be passed a string that does not represent a valid number

```java
String numberAsString = "x";
double amount = 0.0;
try {
    // the next message may "throw an exception"
    amount = Double.parseDouble(numberAsString);
    System.out.println("This message may not be sent");
} catch (NumberFormatException nfe) {
    System.out.println(numberAsString + " not valid");
}
System.out.println("amount = " + amount);
```
parseDouble

```java
public double parseDouble(String s)
    throws NumberFormatException
```

Returns a number new represented by s

**Parameters:** s - the string to be parsed.

**Returns:** the double value represented by the string argument.

**Throws:** NumberFormatException - if the string does not represent a valid number, "100.0" for example

- Many methods throw an exception
  - Your methods may too, in fact some must
Part of Java's Exception hierarchy

- Code that throws `RuntimeException` need not be in a try block, all others must be "tried"
Examples of Exceptions

- `parseDouble` and `parseInt` when the `String` argument does not represent a valid number
- Integer expressions that result in division by 0
- Sending a message to an object when the reference variable has the value of null
- Indexing exceptions:
  - attempting to access an `List` element with an index that is out of range or a character in a `String` outside the range of 0 through `length()-1`
Two Examples of Exceptions

```java
String str = null;
String strAsUpperCase = str.toUpperCase();

Output
Exception in thread "main" java.lang.NullPointerException at main(OtherRunTimeExceptions.java:6)
```

```java
List<String> stringList = new ArrayList<String>();
stringList.add("first");
String third = stringList.get(1);

Output
IndexOutOfBoundsException: Index: 1, Size: 1
java.util.ArrayList.RangeCheck(ArrayList.java:491)
at java.util.ArrayList.get(ArrayList.java:307)
at main(OtherRunTimeExceptions.java:10)
```
Throwing your own Exceptions

- **throws** and **throw** are keywords
- The object after throw must be an instance of a class that extends the **Throwable** class

```java
public void deposit(double depositAmount) throws IllegalArgumentException {
    // Another way to handle preconditions
    if (depositAmount <= 0.0) {
        throw new IllegalArgumentException();
    }
    balance = balance + depositAmount;
}
```
Testing exception is thrown

/** Return a reference to the element at the given index.
 * Runtime: O(1)
 * @returns Reference to object at index if 0 <= index < size()
 * @throws IllegalArgumentException when index < 0 or index>=size()
 */

public E get(int index) throws IllegalArgumentException {
    if (index < 0 || index > size())
        throw new IllegalArgumentException("" + index);
    return (E)data[index];
}

• Write tests to ensure the exception is thrown watch for bugs!

@Test(expected = IllegalArgumentException.class)
public void testExceptionGetElementAtZeroWhenSizeIsZero() {
    OurList<String> list = new OurArrayList<String>();
    // size()==0 so exception should be thrown from the get method
    list.get(0);
    // If the exception was not thrown in get, this test fails
}