CSE 143
Lecture 5

More `ArrayIntList`:

pre/postconditions; exceptions; interfaces

slides adapted from Marty Stepp

http://www.cs.washington.edu/143/

Preconditions

- **precondition**: Something your method *assumes is true* at the start of its execution.
  - Often documented as a comment on the method's header:

```
// Returns the element at the given index.
// Precondition: 0 <= index < size
public void remove(int index) {
    return elementData[index];
}
```

  - Stating a precondition doesn't "solve" the problem, but it at least documents our decision and warns the client what not to do.

Postconditions

- **postcondition**: Something your method *promises will be true* at the *end* of its execution.
  - Often documented as a comment on the method's header:

```
// Makes sure that this list's internal array is large
// enough to store the given number of elements.
// Postcondition: elementData.length >= capacity
public void ensureCapacity(int capacity) {
    // double in size until large enough
    while (capacity > elementData.length) {
        elementData = Arrays.copyOf(elementData,
                                    2 * elementData.length);
    }
}
```

  - If your method states a postcondition, clients should be able to rely on that statement being true after they call the method.

Throwing exceptions (4.5)

`throw new` **ExceptionType**`();`

`throw new` **ExceptionType**`("`**message**`");`

- Causes the program to immediately crash with an exception.

- Common exception types:
  - ArithmeticException, ArrayIndexOutOfBoundsException, FileNotFoundException, IllegalArgumentException, IllegalStateException, IOException, NoSuchElementException, NullPointerException, RuntimeException, UnsupportedOperationException

- Why would anyone ever *want* a program to crash?

Exception example

```
public void get(int index) {
    if (index < 0 || index >= size) {
        throw new ArrayIndexOutOfBoundsException(index);
    }
    return elementData[index];
}
```

– Exercise: Modify the rest of `ArrayIntList` to state preconditions and throw exceptions as appropriate.

Interfaces

Interfaces (9.5)

- **interface**: A list of methods that a class can promise to implement.
  - Inheritance gives you an is-a relationship *and* code sharing.
    - A `Lawyer` can be treated as an `Employee` and inherits its code.
  - Interfaces give you an is-a relationship *without* code sharing.
    - A `Rectangle` object can be treated as a `Shape` but inherits no code.
  - Analogous to non-programming idea of roles or certifications:
    - "I'm certified as a CPA accountant.
      This assures you I know how to do taxes, audits, and consulting."
    - "I'm 'certified' as a Shape, because I implement the Shape interface.
      This assures you I know how to compute my area and perimeter."

Interface syntax

```
public interface **name** {
    public **type** **name**(**type** **name**, ..., **type** **name**);
    public **type** **name**(**type** **name**, ..., **type** **name**);
    ...
    public **type** **name**(**type** **name**, ..., **type** **name**);
}
```

Example:

```
public interface Vehicle {
    public int getSpeed();
    public void setDirection(int direction);
}
```