CSE 12 – Basic Data Structures
Prof. Christine Alvarado
[Slides borrowed/adapted from slides by Cynthia Lee]
Announcements

1. HW2 posted tomorrow. When it is, DON’T WAIT to read it, though you’ll need Wednesday’s material to really dig in
2. Office hours tomorrow 9-11am
3. Seating charts coming Monday
Review quiz

- Questions handed out on paper
- You must have an iClicker2. If you have an iClicker1, we will grade your answers on paper this week only
- EVERYONE should circle their answers on paper anyway. We will collect these papers separately.
Today’s Topics

1. The List ADT
2. Using Generics in Lists
3. Iterators
public interface Collection<E>
extends Iterable<E>

What does the <E> mean in the above code?

A. That this collection can only be used with objects of a built-in Java type called E

B. That the object can be instantiated to work with any object type

C. That a single collection can hold objects of different types
Java generics: Using parameterized types in class definitions

```java
public class RecentRememberer<T> {
    private T lastElement;
    private int numElements;

    public RecentRememberer() {
        numElements = 0;
        lastElement = null;
    }

    public void add(T element) {
        lastElement = element;
        numElements++;
    }

    ...
```
Java generics: Using parameterized types in class definitions

```java
public class RecentRememberer<T> {
    private T lastElement;
    private int numElements;

    public RecentRememberer() {
        numElements = 0;
        lastElement = null;
    }

    public void add(T element) {
        T prevLast = lastElement;
        lastElement = element;
        numElements++;
    }

    ... // More code here
}
```

Type parameter

Is this line legal Java code?
A. Yes
B. No
Java generics: Using parameterized types in class definitions

```java
public class RecentRememberer<T> {
    private T lastElement;
    private int numElements;

    public RecentRememberer() {
        numElements = 0;
        lastElement = null;
    }  

    public T add(T element) {
        T prevLast = lastElement;
        lastElement = element;
        numElements++;
        return prevLast;
    }
```

T can be used to stand for a type (to be specified later anywhere in this class (and its inner classes!))
Exceptions

public class RecentRememberer<T> {
    ...
    public void add(T element) {
        if (element == null) {
            // code goes here
            lastElement = element;
            numElements++;
        }
    }
}

What code could I add if I want to prevent null elements from being added to this class?

A. throw new NullPointerException();
B. return -1;
C. Either A or B
D. None of the above
Next time

- Developing an object that implements the List ADT: Linked Lists