Generics
ArrayList
/*
 * Removes the element at the specified position in this
 * list. Shifts any subsequent elements to the left
 * (subtracts one from their indices).
 */

public void remove(int index) throws IndexOutOfBoundsException
{
    if (index < 0)
        throw new IndexOutOfBoundsException("Cannot retrieve from negative location");
    arra[index] = null;
    for (int i = index; i < count; i++)
    {
        arra[i] = arra[i + 1];
    }
    count--;
Reading from Files

- In, StdIn classes are not part of Java 11
  - do not use them
- Scanner
  - could use for reading files
    - DO NOT
  - DO USE for reading from keyboard
- BufferedReader
  - Use this for reading from files (lab 2, EchoCount)
Generics

• A way to write classes and methods that can operate on a variety of data types without being locked into specific types at the time of definition
  • When objects are created, they are locked to a class
    • Allows for code checking at compile time
    • Automatic casting to appropriate types
• General Idea: Write definitions & implementations with “Generic” parameters
Generic Alternative

- ArrayList class from Tuesday
  - It is defined to store Object
  - So it can store anything

- It is generally a BAD idea to store different types of data in a single data structure.
import java.util.Random;
/
@Author gTowell
* Created: August 28, 2019
* Modified: Jan 24, 2019
* Purpose:
* Generic Methods
/
public class NormalClassWithGenericMethod {
    public static void main(String[] args) {
        Integer[] jj = {1, 2, 3, 4, 5, 6, 7, 8, 9}; // NOTE AUTOBOXING!!
        new GenericMethod().randomize(jj);
        for (int j : jj)
            System.out.println(j);
        String[] ss = {"A", "B", "c", "d", "E", "F"};
        new GenericMethod().randomize(ss);
        for (String s : ss)
            System.out.println(s);
    }

    // The Fisher–Yates shuffle
    public <T> void randomize(T[] data) {
        Random r = new Random();
        for (int i = data.length; i>0; i--)
            { int tgt = r.nextInt(data.length);
             T temp = data[i];
             data[i]=data[tgt];
             data[tgt]=temp;
            }
}
/**
 * Simple generic class example
 * @author gtowell
 *
 * @param <A>
 */
public class GenericClass<A> {
    /** A non-generic value */
    private double amount;
    /** A generic value */
    private A otherValue;
    /**
     * Constructor.
     * @param other the generic value
     * @param amt a double value.
     */
    public GenericClass(A other, double amt) {
        this.otherValue = other;
        this.amount = amt;
    }

    public static void main(String[] args) {
        GenericClass<String> gString = new GenericClass<String>("ASDF", 24.5);
        System.out.println(gString);
        GenericClass<Double> gDouble = new GenericClass<Double>(99.5, 44.5);
        System.out.println(gDouble);
        GenericClass<BufferedReader> gBR = new GenericClass<BufferedReader>(
            new BufferedReader(new StringReader("When in the course")), 99.8);
        System.out.println(gBR);
    }
}
Generics Restrictions

- No instantiation with primitive types
  - \texttt{Genre<Double> ok, but Genre<double> is NOT}
- Can not declare static instance variables of a parameterized type
- Can not create arrays of parameterized types
  - but you can create an array of \texttt{Object} then cast
    - \texttt{T[]} \texttt{array = new T[10]}
    - \texttt{T[]} \texttt{array = (T[])new Object[10]}
Creation with Type Parameters

• When constructing an class with generics, you must specify the type of elements via $<\>$

```java
ArrayList<String> l1 = new ArrayList<>((100);
ArrayList<Integer> l2 = new ArrayList<>();
```
Reimplementation of ArrayList

• Mostly, just change “Object” to “T”

```java
class ArrayListInterface<T> {
    boolean add(T t);
    boolean add(int index, T t) throws IndexOutOfBoundsException;
    void remove(int index) throws IndexOutOfBoundsException;
    T get(int index) throws IndexOutOfBoundsException;
    boolean set(int index, T t) throws IndexOutOfBoundsException;
    int size();
    int indexOf(T t);
    void clear();
}
```
Problem ... how can you compare equality of two generics

- The only functions you can assume exist for a generic are those with Object.

Solution: Override Equals!!!
public class Place {
    private String zip, city, state;
    public String getZip() { return zip; }
    // Other stuff not shown
    @Override
    public boolean equals(Object o) {
        if (o instanceof Place) {
            return zip.equals(((Place) o).getZip());
        }
        return super.equals(o);  // return false;
    }
    public static void main(String[] args) {
        Place p1 = new Place("19380", "a", "b");
        Place p2 = new Place("19380", "aaaa", "bbbb");
        Place p3 = new Place("19385", "a", "b");
        Integer ii = 10;
        System.out.println("p1 p2: " + p1.equals(p2));
        System.out.println("p1 p3: " + p1.equals(p3));
        System.out.println("p1 ii: " + p1.equals(ii));
        System.out.println("p2 p3: " + p2.equals(p3));
    }
}
Groups

• For the ArraList class write:

```java
/**
 * Returns the index of the first occurrence of the specified element in this
 * list, or -1 if this list does not contain the element. More formally, returns
 * the lowest index i such that (o==null ? get(i)==null : o.equals(get(i))), or
 * -1 if there is no such index.
 *
 * @param t the item to be found
 * @return the index of the first occurrence of the specified element in this
 * list, or -1 if this list does not contain the element
 */
int indexOf(T t);
```

A toString method that returns a print representation of every object in the ArraList.
2d ArrayList

```java
public class AL2d {
    public static void main(String[] args) {
        ArrayList<ArrayList<String>> al2d = new ArrayList<>();
        al2d.add(new ArrayList<String>());
        // etc
        al2d.get(0).add("Hello");
        al2d.get(0).add(1);
    }
}
```

- **Not legal!**
- **Add an AL to the “outer” AL**
- **a real mouthful!**
- **add a string to the inner AL**
Using an ArrayList

- **Problem**: Write a program to collect then print all unique words in a file
- **Difficulty**: you do not know the number of distinct words!
- **Solutions**
  - Bad: allocate a really big array
  - Use ArrayList!
WordCounter —
Count the unique words in file!

WordCounter.java
java.util.ArrayList

• Implements much the same interface as ours
  • Their implementation has a few more functions
• Theirs is probably more more more efficient.
• Part of Java collections framework
• import java.util.ArrayList
• Use ArrayList rather than ArraList (ours) for Homework 3.
Collections

TreeSet
LinkedHashSet
ArrayDeque
Vector
PriorityQueue
Stack

Set
SortedSet
NavigableSet

List
AbstractList
LinkedList

Queue
Deque
AbstractQueue

Collection

Iterable

AbstractCollection

Interface
Abstract Class
Class
Lab

• Go to Park 231 (The CS computer labs!!!!)
  • read my document about using the computers in the lab (link near the top of class web page).
• Log into a Unix machine (You may do this in pairs) but there should be just about enough to work solo in the two rooms.
• Start a terminal
• execute ls in the terminal
• take a picture of your screen
• send me the picture
• log out