The ArrayList<E> Class

An ArrayList is a list of objects that expands as objects are added.

- You can think of an ArrayList is an array that expands as

  needed ArrayList is a **Java** class **in java.util**

- Notice the **<E>** here. This is a ***generic***. It is a place holder for a class.

- When we create or instantiate an ArrayList we will substitute a real class name for E.

  Constructors:

  public ArrayList<E>() // sets initial capacity to 1

  ArrayList<String> x = new ArrayList<String>(); //String is substituted for E

  ArrayList«Animal> y= new ArrayList<Animal>(); // an ArrayList of Animal

Or

public ArrayList<E> (n) // n is the initial capacity

Example

ArrayList<lnteger> nums = new ArrayList<Integer>(50);

Some Methods of the ArrayMst<E> class

- boolean add(E x) - adds x to the end of the list
  nums. Add(5); // autoboxing same as num. add(new Integer(5))

- void add(int index, E x) —adds x at position index and moves entries at positions greater than index down .
  nums.add(0, 7)

- boolean contains (E x) -true if x is in the list
  boolean a = nums.contains(5) ;//true in this case

- E get(int index) - returns the object in position x
  Integer b = nums.get(0);// returns 7

- boolean isEmpty() - true if no items are in the list
  boolean a = nums.isEmpty() // returns false

- boolean remove (E x) - removes the first occurrence of *x,* returns true if successful. If x is not in the list returns false
  boolean nums.remove(154) // does nothing and returns false

- E remove(int index) - removes and returns the item at position index, shifts all items below index up one position
  Integer b = nums.remove (0); // returns 7 move the 5 up one position

- E set(int index, E x) - replaces the item at position index with x. Returns the removed item
  Integer b = set(0, 34);// places 34 in position 0

- Int size() — returns the number of items in the list

Example:

```
import java.util.*;
public class DemoArrayList

public static void main(String[]  args)


   ArrayList<Integer> x = new ArrayList<Integer»(50);
   for (int i = 1; i < 20; i++)
        x.add(new Integer(i));    ///could also say x.add(i)...autoboxing

   ArrayList<String> y = new ArrayList<5tring>();
   y.add("Homer“);
   y.add(0,"Marge");
   y.add(1, "Lisa");
   y.add(“Bart");
   y.add(0,"Maggie");
   y.add(2,"Krusty”);

   for (int i = 0; i < x.size(); i++)
        System.out.print(x.get(i) + "  “);
   System.out.println();System.out.println();

  for (int i = 0; i < y.size(); i++)
       System.out.print(y.get(i) + " ");
 System.out.println();System.out.println();

   y.set(1,"Flanders");
   String z = y.remove(2);

     for (int i = 0; i < y.size(); i++)
           System.out.print(y.get(i) + " ");
     System.out.println();System.out.println();
     System.out.println("The removed item was "+z);
     boolean success = y.remove("Homer");

     for (int i = 0; i < y.size(); i++)
      System.out.print(y.get(i) + " ");
      System.out.println();System.out.println();

     y.add(3, "Selma");
     for (int i = 0; i < y.size(); i++)
          System.out.print(y.get(i) + " ");
   System.out.println();System.out.println();

   Object[] array -- y.toArray();

    for (int i = 0; i < array.length; i++)
          System.out.print(array[i] + " ");
      System.out.println();System.out.println();
```

///////////////////////////Box Class///////////////////////////

```
// We used this class before, it compares boxes based on volume


public class Box implements Comparable

 int length, width, height;
 public Box()

  length = width = height = 0;
```

```
public Box( int l, int w, int h)

 length  =  l;
 width = w;
 height = h;

public int volume()

 return  length”width”height;

public int area()

 return  2”length”width + 2*length”height + 2*width*height;



public int compareTo(Object o)  // based on volume

 if(volume()  < ((Box)o).volume())
      return -1;
 if(volume() » ((Box)o).volume())
       return 1;
  return 0;

public String tostring() // inherited from Object and overridden

 return  "Length:  "+ length+ "\t\tWidth:  "+ width + "\t\tHeight: ”+ height+"\t\tVolume:  "+
             volume();

public boolean equals(Object o) // inherited from Object and overridden

 return volume() == ((Box)o).volume();



///////////////////////////////////////////////////////////////////////////////////////////

import java.util.*;
public class SortBoxes

 public static void main(String[]  args)
```

```
ArrayList<Box> b = new ArrayList<Box>();
b.add(new Box(2,5,3));
b.add(new Box(1,5,4));
b.add(new Box(6,2,5));
b.add(new Box(9,1,3));
b.add(new Box(8,9,1));
b.add(new Box(1,2,7));
Collections.sort(b); // sort from parent Collections class
for (int i = 0; i <b.size(); i++)
      System.out.println(b.get(i));
```

Suppose you have a class Animal and Dog, Cat, and Pig all extend animal.

Assume each subclass has a one argument constructor that accepts a String.
For example
Dog d = new Dog("Einstein");

You can take advantage of upcasting with ArrayList<E>

```
Arraylist<Animal> zoo = new ArrayList<Animal>()
zoo.add(new Dog("Fido");
zoo.add(new Cat("Sylvester");
zoo.add(new Pig ("Porky");
```

If you instantiate an array list without specifying a class, the ArrayList will the an arrayLiist of Object

Example:

```
ArralList x= new ArrayList();
a.add("Seldon"); // a String is-a Object
a.add(new Integer (5)); // an Integer is-a Object
a.add(New Dog("Fido")); // a Dog is-a Object
a.add(new Box (3,4,5))); // a Box is-a Object
```

ArrayList and Generics

127

```
importjava.util.”;
class Student

        private String name;
        private double GPA;
        public Student ()

                name =   ;
                GPA = 0.0;

        public Student (String name, double GPA)

                this.name = name;
                this.t3PA = oPA;

        public String getName()

                retum name;

        public double getGPA()

                retum GPA;
```

What will happen here?

```
public class PńntStudents

        public static void main(Stńng[] args)

                ArrayList list = new ArrayList(); // can hold any Object
                list.add( new Student("Bart", 2.1));
                list.add( new Student("Lisa", 3.8));
                list.add( new String(“Homer 2.0"));
                list.add( new Integer (345));
                for (int i = 0; i< list.size(); i++)
                      System.out.println( ((Student)list.get(i)).getName());
```

```
public class PrintStudents1

          public static void main(String[]  args)

                    ArrayList<Student> list = newArrayList<Student>{}; //Students  only
                    list.add( new Student("Bart", 2.1));
                    list.add( new Student("Lisa", 3.8));

                    list.add( new String("Homer 2.0")); //No Strings
                    lis£add(345); //  No Integers allowed

                    for (int i = 0; i< list.size(); i++)
                              System.out.println( ((Student)list.get(i)).getName());
```

From the Compiler:

```
C:\CS 104 programs\PrintStudents1,java:34: cannot find symbol
symbol  : method add(java.lang.Stñng)
location: class java.util.ArrayList<Student»
                    list.add( new String("Homer 2.0"));


C:\CS 104 programs\PrintStudents1.java:35: cannot find symbol
symbol  : method add(int)
location: class java.util.ArrayList<Student»
                    list.add(345);

2 errors
```

The compiler caught the error. ...It did not occur at runtime

xercise:

1. Write a class PhoneBook That has two string entries

    String name // First name only
    String number //Cell number

    The class should have constructors, getters, setters, toString() and implement comparable based on name.
    (Hint: String already has a compareTo method. That should make things simple)
    toString() should return the name and number with the proper labels.

2. Write a program that creates 5 PhoneBook objects, stores them in an ArrayList, and then prints the entries in alphabetical order.

```
public class PhoneBook implements Comparable
{
  private String name, number;
  public PhoneBook()
  {
    name = "";
    number = "";
  }
  public PhoneBook(String name, String number)
  {
    this.name = name;
    this.number = number;
  }

  public int compareTo(Object o)
  {
    return    name.compareTo(((PhoneBook)o).name);
  }
  public String getName()
  {
    return name;
  }
```

```
public String getNumber()
  {
    return number;
  }

  public void setName(String n)
  {
    name = n;
  }
  public void setNumber(String n)
  {
    number = n;
  }
  public String toString()
  {
    return "Name: "+ name+ " Number: " + number;
  }
}
```

```
import java.util.*;
public class TestPhoneBook
{
  public static void main(String[] args)
  {
    ArrayList<PhoneBook> phoneBook= new ArrayList<PhoneBook>();
    phoneBook.add(new PhoneBook("Homer", "456-348-6999"));
    phoneBook.add(new PhoneBook("Bart", "456-666-6789"));
    phoneBook.add(new PhoneBook("Marge", "456-458-6789"));
    phoneBook.add(new PhoneBook("Lisa", "456-228-3689"));
    phoneBook.add(new PhoneBook("Maggie", "456-444-6909"));
    Collections.sort(phoneBook);
    for (int i = 0; i < phoneBook.size(); i++)
       System.out.println(phoneBook.get(i));
  }
}
```