Dealing with lots of values

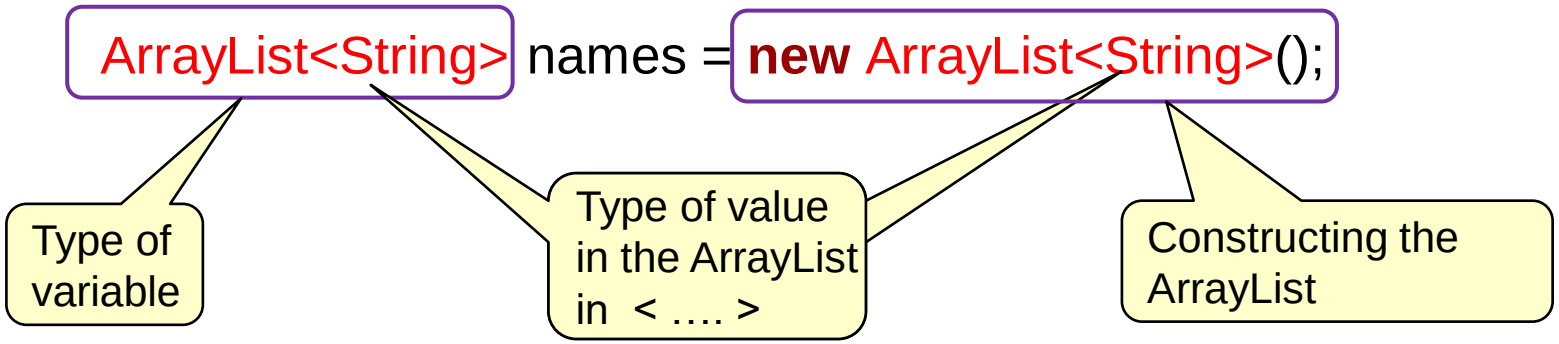
- We've used ArrayLists:
  - Pollution Analyser,
  - GraphPlotter, WordSearcher, SalesVisualiser, FileEditor,
- ArrayLists of numbers, Strings, other objects.
- Created by methods
  - UI.askNumbers(…) and UI.askStrings(…)
  - Files.readAllLines(Path.of(filename)) (actually, gave us a List, not ArrayList)
- Used **for** each loops to step through items in an ArrayList
- What more can you do with an ArrayList?

Using ArrayLists

- How can we create a new ArrayList?
- How can we add items to an ArrayList?
- How can we access items in an ArrayList without having to step through the whole list.
- How can we modify an ArrayList?

Using ArrayList: create new

- Creating empty ArrayList:



  - Must specify the type of value
  - Can't be int or double or boolean!!!
    Have to use Integer or Double or Boolean ("wrapper" objects)

    ```
    ArrayList<Integer> counts = new ArrayList<Integer>();
    ```

  - Must have the ( ) - standard constructor, with no arguments.

Using ArrayList: adding to the end

- Adding an item at the end of an ArrayList: .add(….) method

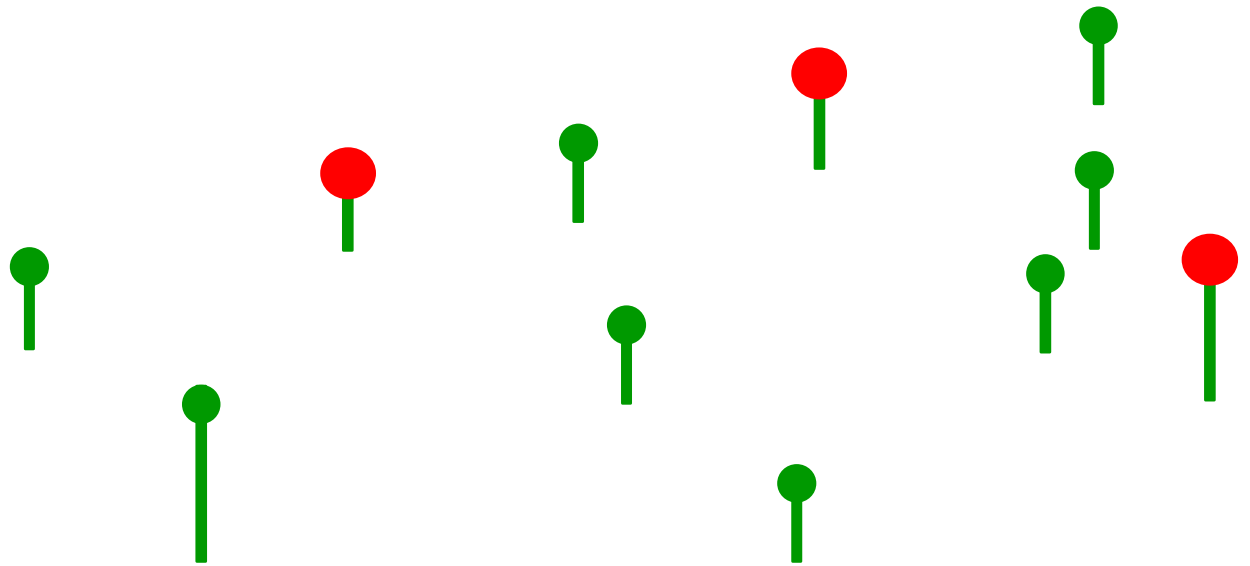
```
ArrayList<Double> values = new ArrayList<Double>();
for (int i = 0;  i < 10; i++){
    values.add (UI.askDouble("Enter next number"));
}
```

- value to be added must be of the right type for the list.

Garden Program: Flower class

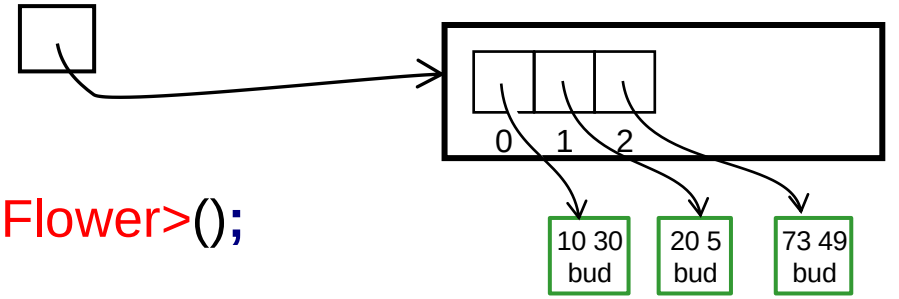
Flower(double x, double y)

void	bloom()	Make the flower bloom, if it is in the "bud" stage
void	grow(int v)	make the flower taller, if it is in the "bud" or "bloom" stage.
void	pick()	Make the flower half its height, if in the "bud" or the "bloom" stage,
boolean	touching(double x, double y)	Is the point x,y on top of the flower?

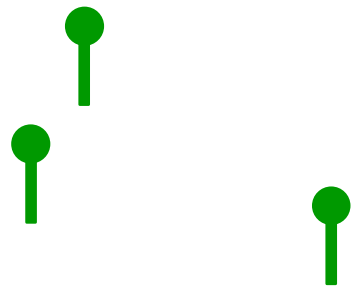


Garden Program

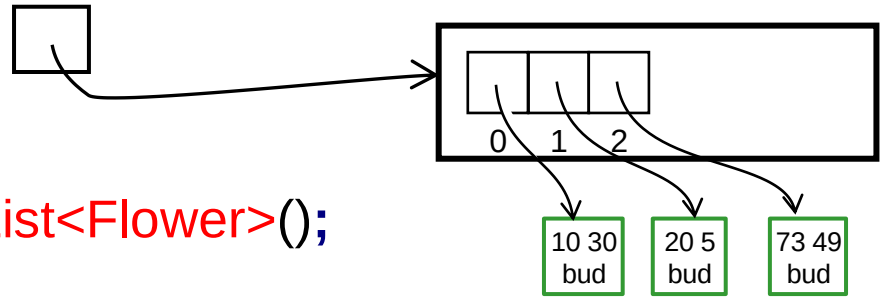
- Needs to keep track of all the flowers that have been planted
  - needs and ArrayList of Flowers:




```
public class Garden {
    private ArrayList<Flower> flowers = new ArrayList<Flower>();
```



Garden Program

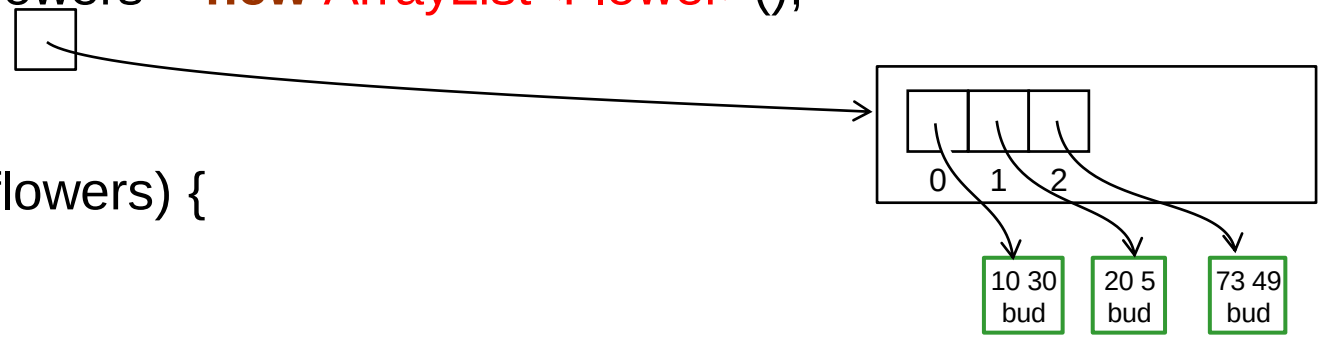


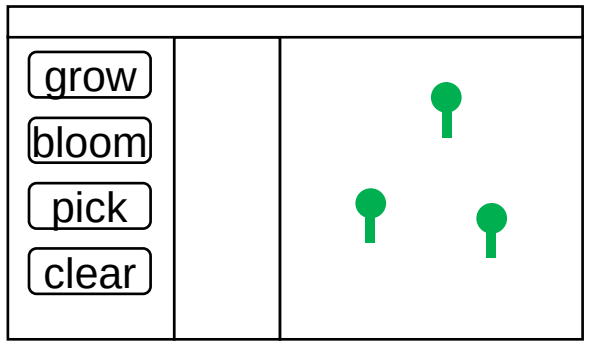

```
public class Garden {
    private  ArrayList<Flower> flowers = new  ArrayList<Flower>();

    public void setupGUI(){
        UI.setMouseListener( this ::doMouse);
        UI.addButton("Grow",   this::doGrow);
        UI.addButton("Bloom", this::doBloom);
        UI.addButton("Pick",     this::doPick);
        UI.addButton("Clear",   this::doClear);
    }

    public  void doMouse(String action, double x, double y){
        if (action.equals("released")){
            Flower fl = new Flower(x, y);          // or    this.flowers.add(new Flower(x,y));
            this.flowers.add(fl);
        }
    }
```

Garden program

- Operate on the ArrayList of Flowers:

```
private ArrayList<Flower> flowers = new ArrayList<Flower>();

public void doGrow(){
    for (Flower flower : this.flowers) {
        flower.grow();
    }
}
public void doBloom(){
    for (Flower flower : this.flowers) {
        flower.bloom();
    }
}
public void doClear(){
    this.flowers = new ArrayList<Flower>();
    UI.clearGraphics();
}
```

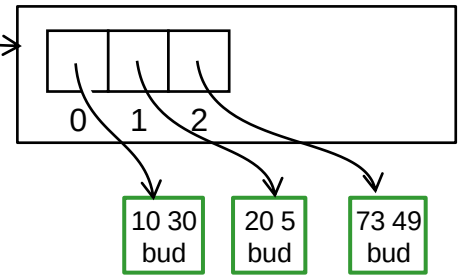


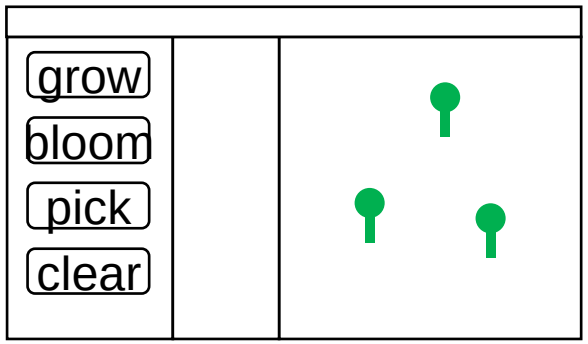


Garden program: Selecting Flowers

- Operate on a *selected* Flower:

```
private ArrayList<Flower> flowers = new ArrayList<Flower>();

private Flower selectedFlower;


public void doGrowOne(){
    this.selectedFlower.grow(10);
}
public void doBloomOne(){
    this.selectedFlower.bloom();
}
public void doPickOne(){
    this.selectedFlower.bloom();
}
```

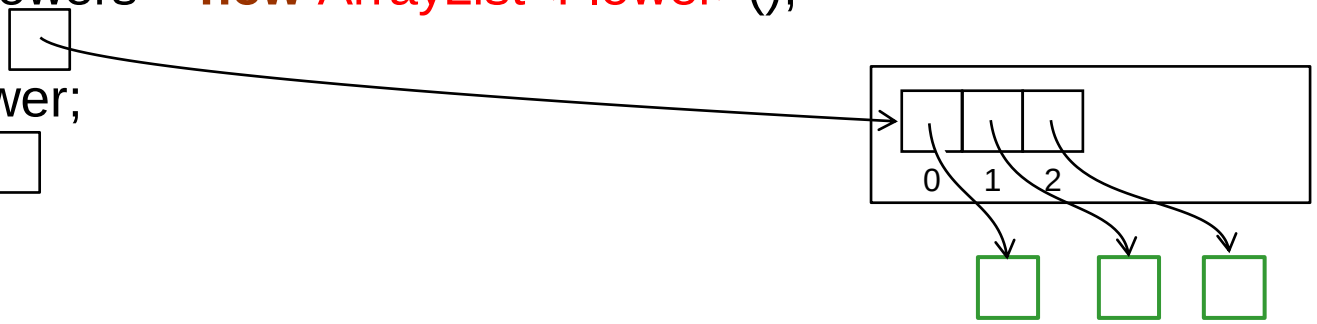




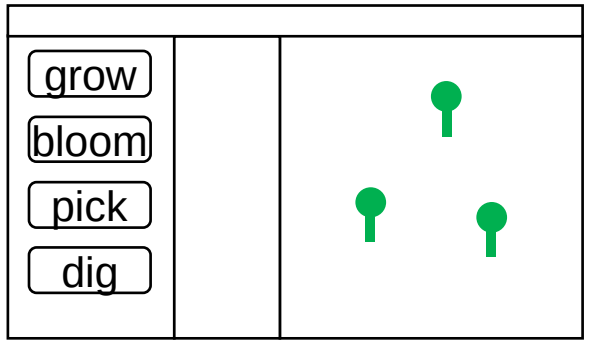
Garden program : Selecting Flowers

- Operate on a *selected* Flower:

```
private ArrayList<Flower> flowers = new ArrayList<Flower>();

private Flower  selectedFlower;

    :
```



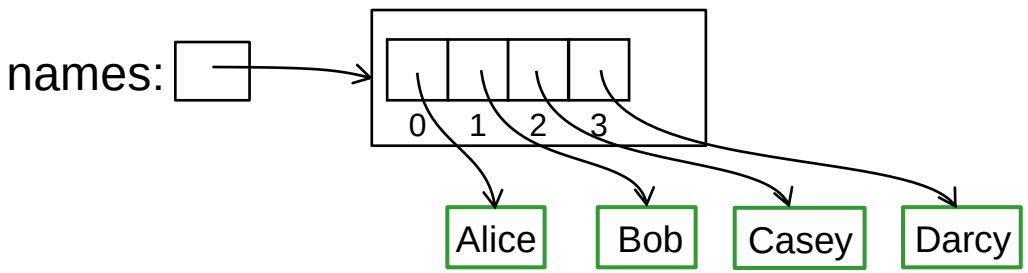

```
public void doMouse(String action, double x, double y){
    if (action.equals("released")){
        for (Flower flower : this.flowers) {
            if  (flower.touching(x, y) ) {
                this.selectedFlower = flower ;
                return ;
            }
        }   // if not touching a flower, plant one
        this.flowers.add(new Flower(x, y));
    }
}
```



Doing more with ArrayLists

- How do we
  - add a new value in the middle of an ArrayList?
  - access the values in the ArrayList, except by stepping through the whole list?
  - change the value at an index in the ArrayList?
  - remove a value from the ArrayList?
  - find out how big the ArrayList is?
  - clear the ArrayList?

- Items in an ArrayList are numbered: with an index starting at 0:



Using ArrayList

- For all actions, call methods on the ArrayList:
  - size(), add(…), get(…), set(…), remove(…), clear(), contains(…), indexOf(…), isEmpty(), …

```
ArrayList<String> names = new ArrayList<String>();

names.add("Jim");

names.add("Jan");

if (names.get(0).equals("Jim") {
    names.set(0, "Jane");
    …

 names.add(1, "Bob");

 names.remove("Bob");

names.remove(0);
```

Adds an item at the end of the list

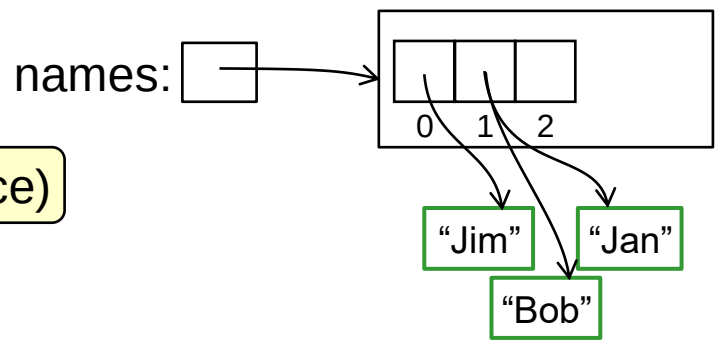
Gets the item at a position

Replaces the item at a position with a new item

Adds an item at a position

Removes item (1st occurrence)

Removes item at a position



ArrayLists Methods: size & add

mylist.**size**()

- returns the number of items in mylist.
- Note contrast to String: .size() vs .length()

```
UI.printf("The garden had %d flowers\n", this.flowers.size());
```

mylist.**add**(item)

item must be of the right type

- adds the item at the end of the ArrayList

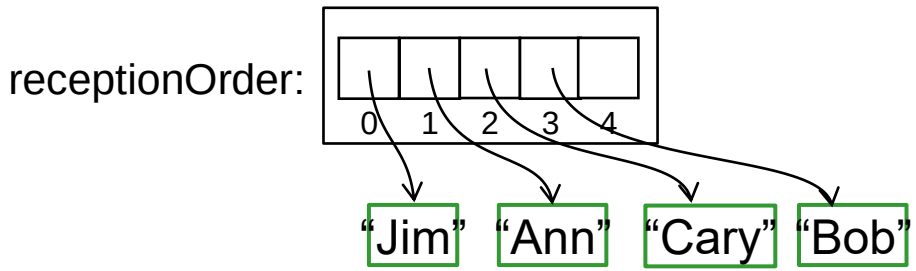
```
balloons.add( new Balloon(x, y, color) );
```

item must be of the right type

mylist.**add**(index, item)

- inserts the item at position index (0 .. size)

```
int pos = UI.askInt("Where do you want " +name);
receptionOrder.add(pos, name);
```



ArrayLists Methods: get & set

mylist.**get**(index)

- returns the item at position *index* (0 .. size-1)

mylist.**set**(index, item)

- replaces the current value at position *index* with *item*
- returns the old value at *index*
- *index* must be 0 .. size-1

```
UI.print("Which two units do you want to swap?");
int first = UI.askInteger("first");
int scnd = UI.askInteger("second");
```

```
if (first>=0 && first< attackList.size() &&
    scnd>=0 && scnd<attackList.size() &&
    first!=second){
```

```
String temp = attackList.get(first);
attackList.set(first, attackList.get(scnd));
attackList.set(scnd, temp);
```

Or

```
attackList.set(first, attackList.set(scnd, attackList.get(first)));
```

ArrayLists Methods: isEmpty & clear

mylist.**isEmpty**()

- returns true iff there are no items in mylist.

```
// Make each unit advance, if there are any units
if ( ! attackList.isEmpty() ) {
    for (Unit unit : attackList) {
        unit.checkPath();
        unit.advance(3);
    }
}
```

mylist.**clear**()

- removes all values from the list.

```
// Restart and clear the list of all elements.
public void doRestart() {
    UI.clearGraphics();
    this.flowers.clear();
}
```

ArrayLists Methods: contains & remove[1]

mylist.**contains**(item)

- returns true if the item is somewhere in mylist

mylist.**remove**(item)

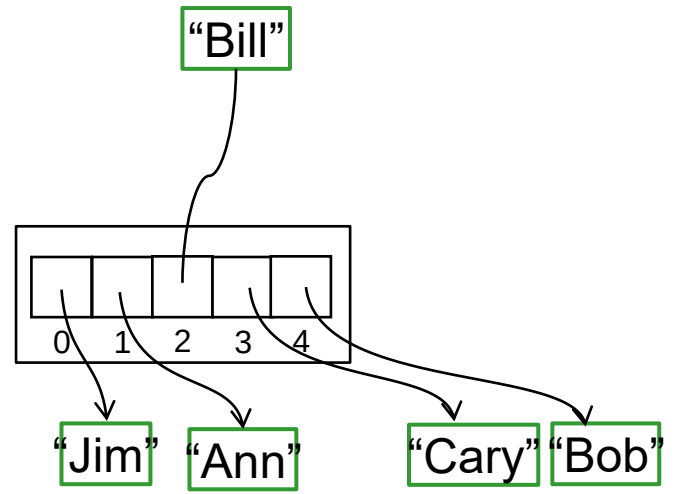
- removes the item, if it is present, and shuffles later items down
- returns true iff item was removed

first occurrence of item if item is at several places in the list

```
// Respond to a "Remove Person" button
String name = UI.askString("Person to remove");
if (receptionOrder.contains(name)){
    receptionOrder.remove(name);
}
else {
    UI.println("That person is not in the reception order");
}
```

Or

```
if ( ! receptionOrder.remove(name) ){
    UI.println("That person is not in the reception order");
}
```



ArrayLists Methods: indexOf & remove²

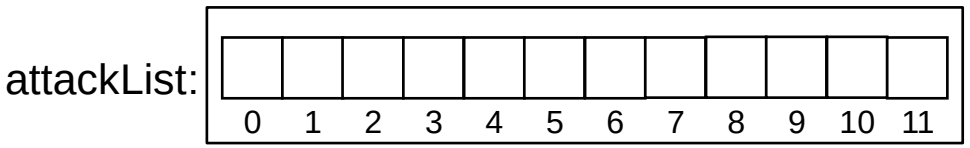
mylist.**indexOf**(item)

- returns the position of item in mylist
- returns –1 if the item is not present

```
// Report position on waiting list
String name = UI.askString("Your name:");
int index = waitingList.indexOf(name);
if (index == –1) {  UI.println("You are not on the waiting list"); }
else             {  UI.println("You are number " +  index + " in order"); }
```

mylist.**remove**(index)

- removes the item at position *index* (0 .. size-1)
- returns the value that was removed




```
// Remove every third unit from attackList
for (int index=2;  index<attackList.size(); index =index +2){
    attackList.remove(index);
}
```

Removing many items from an ArrayList

Remove balls past the right edge:

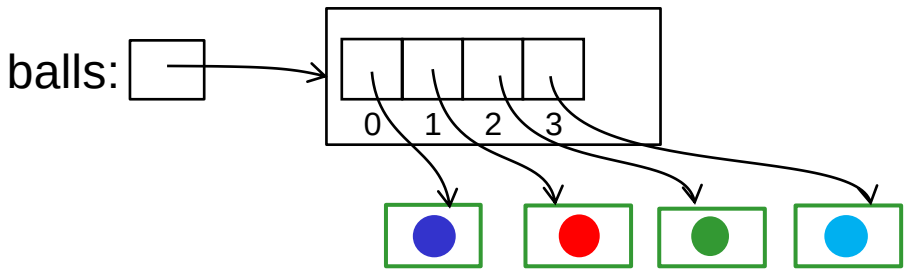
```
for (Ball ball : this.balls) {
    if (ball.getX()>RIGHT) {
        balls.remove(ball);
    }
}
```

Program will CRASH!
Not allowed to change the list while iterating with a "for each" loop

Must use a standard for loop

```
for (int index = 0 ;  index < this.balls.size();  index++) {
    if (this.balls.get(index).getX()>RIGHT) {
        this.balls.remove(index);
    }
}
```

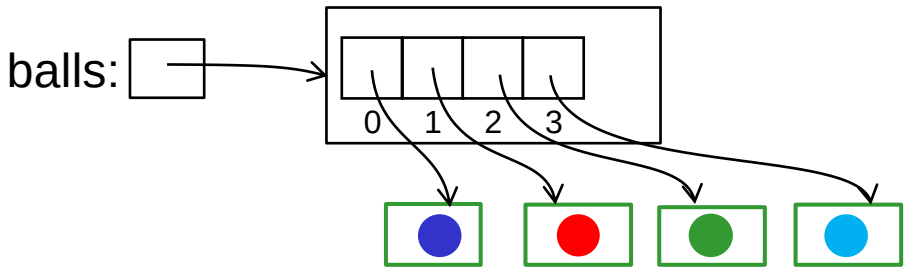
Won't crash,
But still won't work properly!



BE CAREFUL WHEN MODIFYING THE LIST IN A LOOP!

Removing items from ArrayList in a loop

1) Step back after removing item




```
for (int num = 0 ;  num < balls.size();  num++) {
        if (balls.get(num).getX() > RIGHT) {
          balls.remove(num);
          num = num - 1;
      }
}
```

Or

2) iterate backwards from the end.

```
for (int num = balls.size()-1 ;  num >= 0;  num--) {
        if (balls.get(num ).getX() > RIGHT) {
          balls.remove(num);}
}
```

ToDo list Program

```
import ecs100.*;    import java.util.*;
public class ToDo{
    private ArrayList<String> items = new ArrayList<String>();
    public static  void main(String[ ] args){
       ToDo td = new ToDo();
       UI.addButton("New", td::doNewList);
       UI.addTextField("Add", td::doAddItem);
       UI.addButton("List", td::displayList);
       UI.addTextField("Contains", td::doContains);
       UI.addTextField("Remove", td::doRemove);
       UI.addButton("Load", td::doLoad);
       UI.addButton("Save", td::doSave);
       UI.addButton("Quit", UI::quit);
    }
    public void doNewList(){
        this.items = new ArrayList<String>();
        this.displayList();
    }
    public void doAddItem(String item){
        this.items.add(item);
        this.displayList();
    }
    public void displayList(){
        UI.clearText();
        UI.printf("List has %d items:\n", this.items.size());
        for (String item : items){
            UI.println(item);
        }
    }
    public void doContains(String item){
        if (this.items.contains(item)){
            UI.println(item+"  is in the list");
        }
        else {
            UI.println(item+" is not in the list");
        }
    }
    public void doRemove(String item){
        if (this.items.remove(item)) {
            UI.println(item+" was removed");
        }
        else {
            UI.println(item+" was not present");
        }
    }
```

ToDo list Program

```
public void doSave(){
    try{
        PrintStream ps = new PrintStream(new File("todolist.txt"));
        for (String item : items){
            ps.println(item);
        }
        ps.close();
    }
    catch(IOException e){UI.println("todolist.txt is broken"+e);}
}

public void doLoad(){
    this.items.clear();
    try{
        Scanner sc = new Scanner(new File("todolist.txt"));
        while (sc.hasNext()){
            this.items.add(sc.nextLine());
        }
        sc.close();
    }
    catch(IOException e){UI.println("todolist.txt is broken"+e);}
}
```

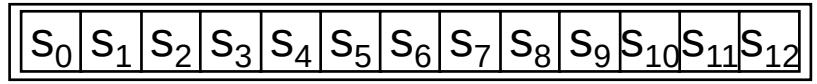
Patterns

- There are many common patterns for working with ArrayLists.
  - Do something with each value
  - Search for any item that satisfies some property
  - Check if all items have some property
  - Do something with every adjacent pair of items.
  - Do something with every possible pair of items.

"Any" pattern: Finding a value

- Search ArrayList to see if it contains a given value
  - eg: ArrayList of names: finding if name is present in any element

(field)

names: 


```
public boolean containsName(String name){
    boolean ans = false;
    for (String n : this.names){
        if ( n.equalsIgnoreCase(name) ) {
            ans = true;
        }
    }
    return ans;
}
```

default answer if no such element

the answer if you find one

no else clause! why not?

```
public boolean containsName(String name){
    for (String n : this.names){
        if ( n.equalsIgnoreCase(name) ) {   return  true;   }
    }
    return false;
}
```

why keep searching once you know the answer?

default answer: if no element was equal

"All" pattern: Check valid

- Check ArrayList of names: all must have at least one space in them

```
public boolean checkValidNames(ArrayList<String>  names){
    boolean allOK= true;
    for (int i = 0; i < names.size() ; i++){
        if ( ! names.get(i).contains("  ") ) {
            allOK = false;
            break;
        }
    }
    return allOK;
}
```

default: answer if all pass

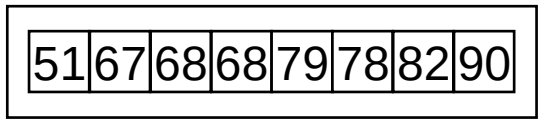
the answer if one fails.

```
public boolean checkValidNames(ArrayList<String>  names){
    for (String nm : names){
        if ( ! nm.contains("  ") ) { return false; }
    }
    return true;
}
```

Working on multiple items at once

- Doing something to each *adjacent* pair
  - Given an ArrayList of numbers, check if in order:

data:



Can't easily use 'for each'!

```
public boolean checkOrdered(ArrayList<Double> data){
    for ( int i = 0;  i<data.size()–1;  i++){
        if ( data.get( i ) > data.get(i+1)){
            return  false;
        }
    }
    return true;
}
```

compare with next item:
stop before size-1

```
public boolean checkOrdered(ArrayList<Double> data){
    for (int i = 1; i<data.size(); i++){
        if ( data.get( i-1 ) > data.get( i ) ){
            return  false;
        }
    }
    return true;
}
```

compare with previous item:
start at 1

Working on multiple items at once

- "Smooth" the data:

$$smooth_i = 0.25\ data_{i-1} + 0.5\ data_i + 0.25\ data_{i+1}$$

  - Given a list of numbers, generate a new list:

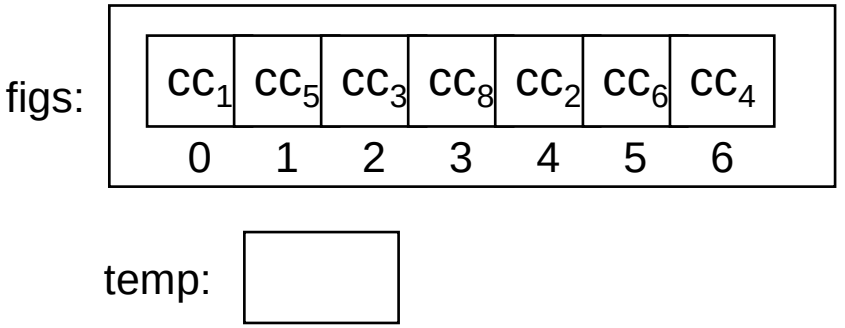
data:

51	67	68	68	79	78	82	90

```
public ArrayList<Double> smooth(ArrayList<Double> data){
    ArrayList<Double> answer = new ArrayList<Double>();
    answer.add( 0.67*data.get(0) + 0.33*data.get(1) );          // first element is special
    for (int i = 1; i<data.size() –1; i++){
        answer.add( 0.25 * data.get( i-1) + 0.5 * data.get( i) + 0.25 * data.get( i+1) );
    }
    answer.add( 0.33 * data.get(data.size() – 2) + 0.67 * data.get(data.size()-1);  // last element
    return answer;
}
```

Changing the order: reverse

```
public void reverseList(){
    for (int i = 0; i < this.figs.size()/2;  i++){
        CartoonCharacter temp = this. figs.get( i );
        this.figs.set( i,  this.figs.get(this.figs.size() – 1 – i);
        this.figs.set (this.figs.size() – 1 – i , temp);
}
```



```
public void reverseList(){
    for (int i = 0;  i<this.figs.size()/2;  i++){
        this.figs.set( i,  this.figs.set( this.figs.size() – 1 – i),  this.figs.get( i) ) );
    }
}
```

set returns the old value at the position!!

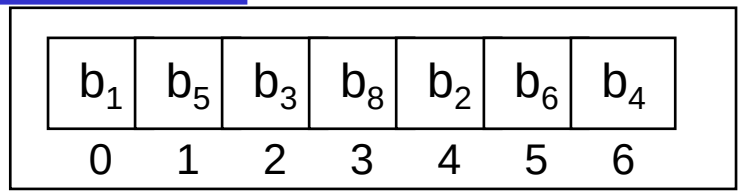
```
public void reverseList(){
    ArrayList<CartoonCharacter> ans = new ArrayList<CartoonCharacter>();
    for (int i = this.figs.size()-1;   i >= 0;  i- -){
        ans.add(this.figs.get( i) ) ;
    }
    this.figs = ans;
}
```

Acting on *every* pair of values

Given ArrayList with Ball objects

check if any two Balls are colliding with each other:

(assume Ball class has a method called isTouching)




```
for ( Ball b1  :  this.balls){
    for ( Ball b2  :  this.balls ){
        if (b1 != b2 && b1.isTouching(b2) ){
            //do something to b1 and b2
        }
    }
}
```

What's the problem?

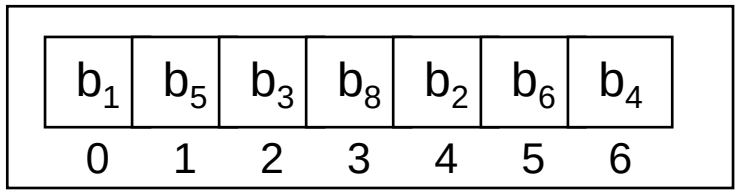
```
for ( int i=0;  i<this.balls.size();  i++ ){
    for ( int j=0;  j< this.balls.size();  j++ ){
        if (i != j  &&  this.balls.get( i ).isTouching(this.balls.get( j ) ){
            //do something to this.balls.get( i )  and   this.balls.get( j )
    }
}
```

Acting on *every* pair of values

Check if any two Ball objects are colliding with each other:

```
for (int i=0;   i<this.balls.size();   i++){
    for (int j=i+1;   j<this.balls.size();   j++){
        if ( this.balls.get( i ).isTouching(this.balls.get( j )) ){
            //do something to  this.balls.get( i )  and   this.balls.get( j )
        }
    }
}
```

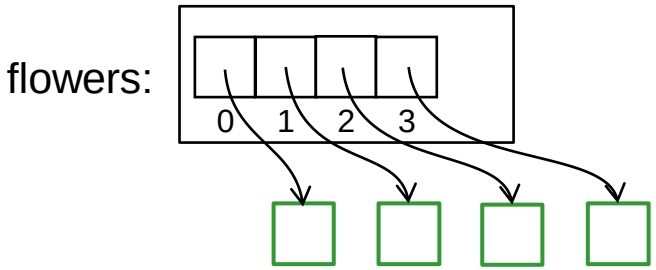
this.balls:



```
for (int i=0;   i<this.balls.size();   i++){
    Ball b1 = this.balls.get( i );
    for (int j=i+1;   j<this.balls.size();   j++){
        Ball b2 = this.balls.get( j );
        if (b1.isTouching(b2) ){
            //do something to  b1  and  b2
        }
    }
}
```

Saving and loading lists of objects

- Save a garden of flowers to a file to load later:
  - Get file name and open PrintStream to file
  - step through list of flowers, printing info to file



- Load a garden of flowers from a file
  - Make an empty list
  - Get file name and read all the lines.
  - for each line,
    - extract the values
    - create new Flower
    - add to the list.

```
104  268  40  bud
287 132  70  bloom
524 245  20  picked
274  83  50  bud
```

Saving and Loading objects

- Turn Object into text in a file that can be read in order to reconstruct the Object.

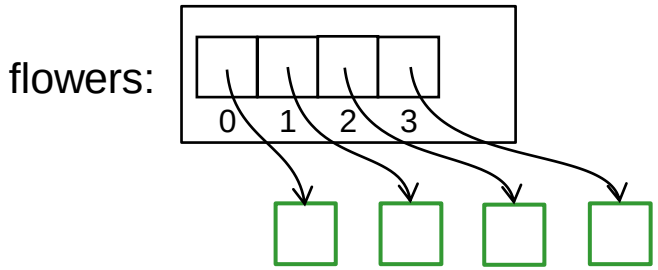
eg, for the Flower class:

toString()
- a standard method for all Objects.
- println knows about it.

```
/** Returns a String representation of the Flower, suitable for saving to a file */
public String toString(){
    return ""+this.baseX+" "+this.baseY+" "+this.height+" "+ this.stage;
}

/** Constructor #2 : Makes a new Flower with the specified values.  */
public Flower(double x, double y, double ht, String st){
    this.baseX = x;
    this.baseY = y;
    this.height = ht;
    this.stage = st;
    this.draw();
}
```

Saving Garden to a file

- Save a garden of flowers to a file to load later:
  - Get file name and open file
  - step through flowers, printing to file



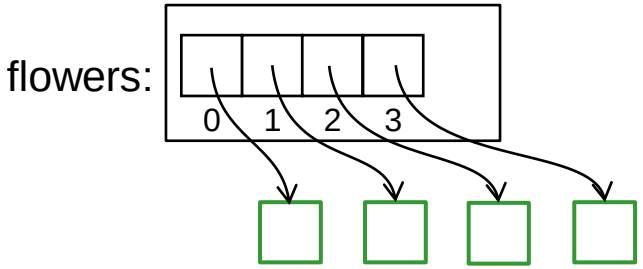

```
public void doSave(){
    try{
        PrintStream out = new PrintStream(UIFileChooser.save("File for Garden"));
        for (Flower flower : this.flowers) {
            out.println(flower.toString());        or just        out.println(flower);
        }
        out.close();
    }
    catch(IOException e) { UI.println("File saving failed: "+e); }
}
```

toString() method called automatically

Loading Garden from a file

- Get file name and open file
- Step through file, reading

```
104 268  40  bud
287 132  70  bloom
524 245  20  picked
274  83  50  bud
          :
```




```
public void load(){
    ArrayList<String> lines = Files.readAllLines(Path.of(UIFileChooser.open("Garden File")));
    this.flowers.clear();              // or  this.flowers = new ArrayList<Flower>();
    for (String line : lines){
        Scanner sc = new Scanner(line);
        double x = sc.nextDouble();
        double y = sc.nextDouble();
        double height = sc.nextDouble();
        String stage = sc.next();
        this.flowers.add(new Flower(x, y, height, stage) );
    }
}
```