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("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(Path.of("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: | s_0 | s_1 | s_2 | s_3 | s_4 | s_5 | s_6 | s_7 | s_8 | s_9 | s_{10} | s_{11} | s_{12} |

```
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 that all values in an ArrayList satisfy some condition
  - eg: field containing an ArrayList of marks: check all between 0 and 100

```
public boolean checkValidMarks(){
    boolean ans = true;
    for (int i = 0; i < this.marks.length ; i++){
        if ( this.marks.get(i) < 0  ||  this.marks.get(i) > 100 ) {
            ans = false;
        }
    }
    return ans;
}
```

default: answer if all pass

the answer if one fails.

```
public boolean checkValidMarks(){
    for (double mk : this.marks){
        if (  mk < 0  ||  mk > 100) ) {   return false;  }
    }
    return true;
}
```

Works on array or ArrayList

"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:

51	67	68	68	79	78	82	90

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

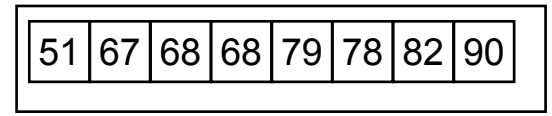
Working on multiple items at once

- "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:

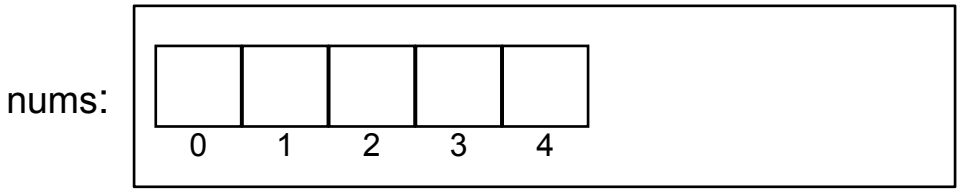
data:




```
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;
}
```

Patterns with ArrayLists of numbers

```
public void doubleAll(ArrayList<Double> nums){
    for (int i = 0 ;  i < nums.size();  i++) {
        nums.set(i, nums.get(i)*2);
    }
}
```

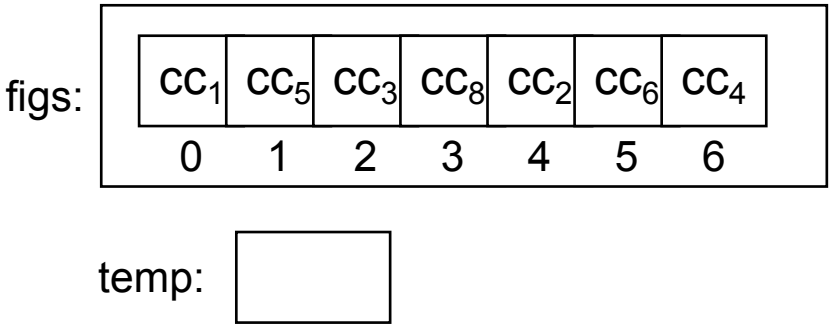



```
public boolean allRising(ArrayList<Double> nums){
    for (int i = 1 ;  i < nums.size();  i++) {
        if ( nums.get(i) <= nums.get(i-1)) {  return false;  }
    }
    return true;
}
```

```
public void reflect (ArrayList<Double> nums){
   int numItems= nums.size();
   for (int i = 0 ;  i < numItems;  i++) {
       nums.add(nums.get(numItems-i);
   }
   }
```

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;
}
```