Introduction to ArrayLists

(based on Ch. 4, Objects First with Java - A Practical Introduction using BlueJ, © David J. Barnes, Michael Kölling)

Produced by: Ms. Mairéad Meagher
Dr. Siobhán Drohan
Ms Siobhan Roche

Ollscoil Teicneolaíochta an Oirdheiscirt
South East Technological University

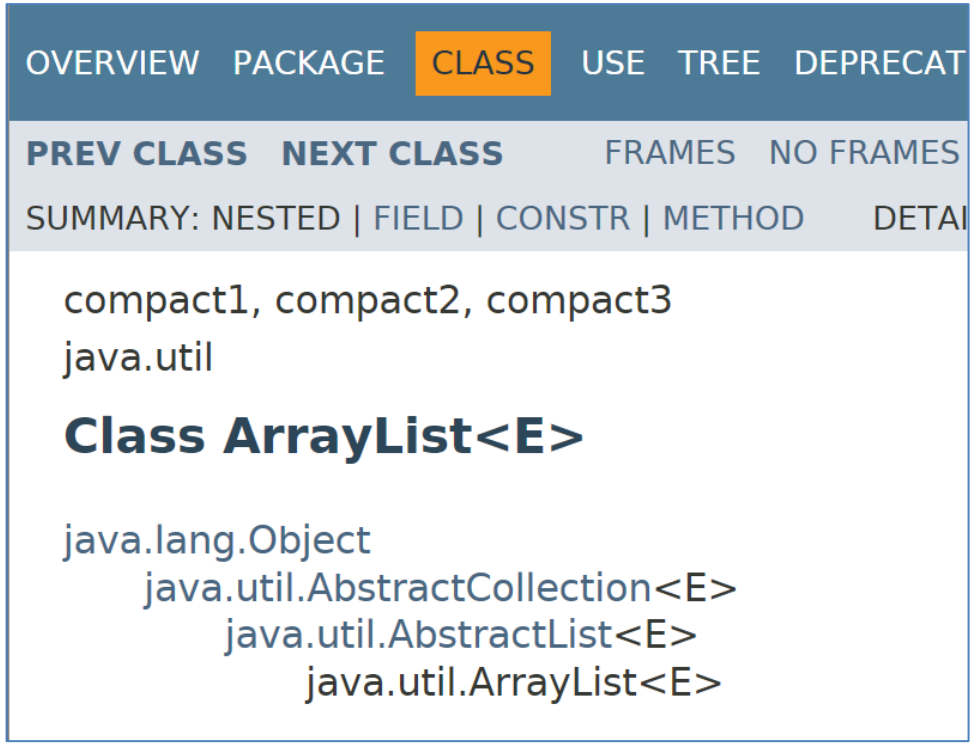
Department of Computing and Mathematics
http://www.setu.ie/

Topic list

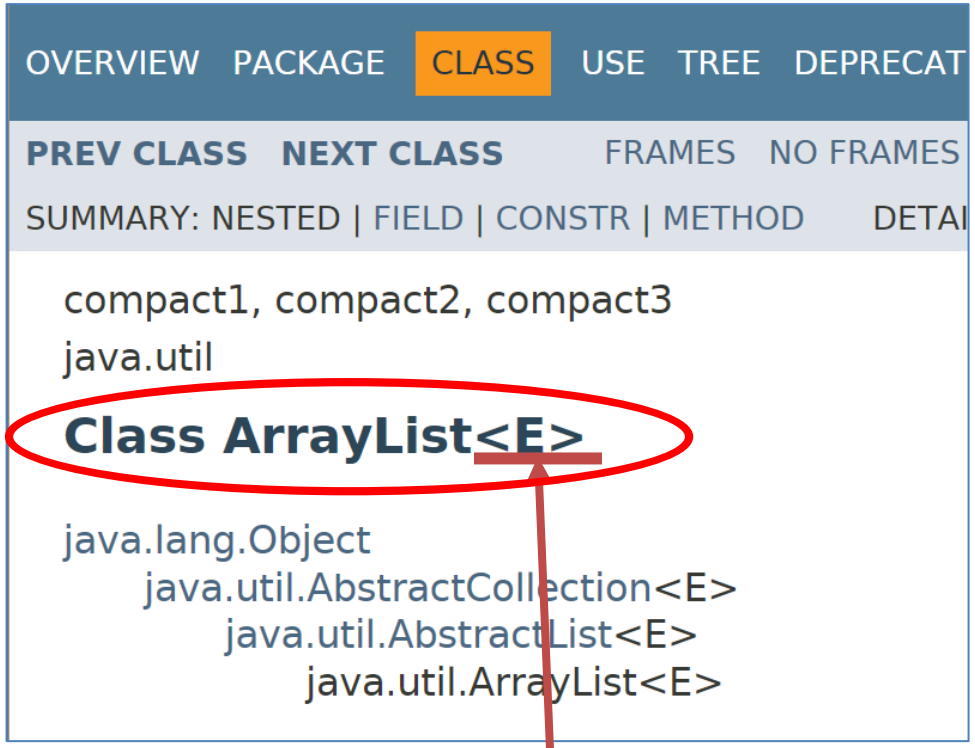
1. Grouping Objects
   - Developing a basic personal notebook project using **Collections** e.g. **ArrayList**
2. **Indexing** within Collections
   - Retrieval and removal of objects
3. **Generic / Parameterized classes**
   - e.g. ArrayList
4. **Iteration**
   - Using the for loop
   - Using the while loop
   - Using the **for each** loop

- Next SlideDeck:
  coding a Shop Project that stores an ArrayList of Products.

Generic/Parameterized Classes



Generic/Parameterized Classes



Collections are known as ***parameterized*** or ***generic*** types.

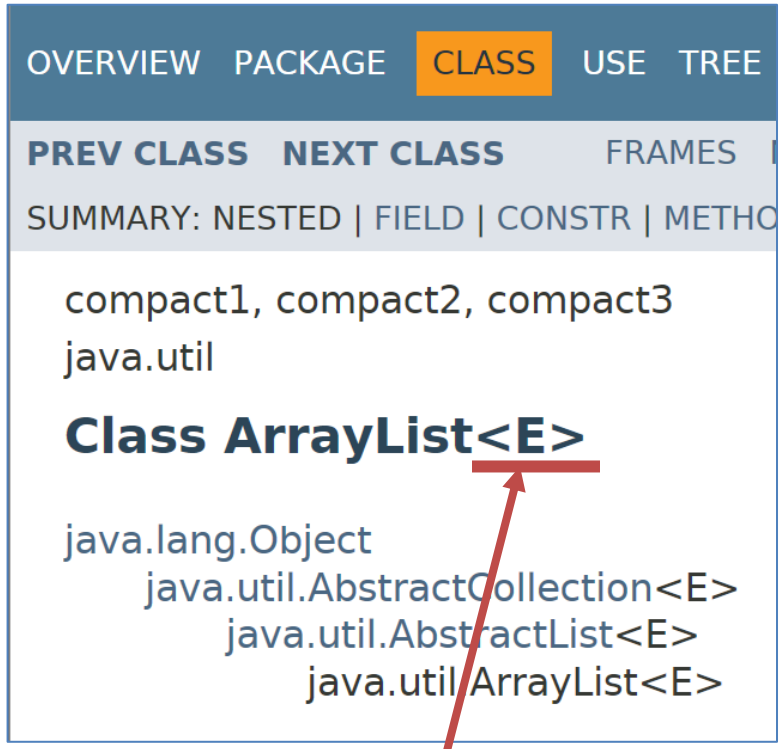
Note **<E>** is the parameter.

E gets replaced with some Class or Type

String is not parameterized.

Generic/Parameterized Classes

String is not parameterized.



ArrayList is parameterized.

The **type parameter** **<E>**
says what we want a list of e.g.:

```
ArrayList<Person>
ArrayList<TicketMachine>
ArrayList<String>
```

etc.

Topic list

1. Grouping Objects
   - Developing a basic personal notebook project using **Collections** e.g. **ArrayList**
2. **Indexing** within Collections
   - Retrieval and removal of objects
3. **Generic classes**
   - e.g. ArrayList
4. **Iteration**
   - Using the for loop
   - Using the while loop
   - Using the **for each** loop

- Next SlideDeck:
  coding a Shop Project that stores an ArrayList of Products.

Processing a whole collection (**iteration**)

- We often want to perform some actions an **arbitrary** number of times.
  - E.g.,
    Print all the notes in the notebook.
    How many are there?
    Does the amount of notes in our notebook vary?

- Most programming languages include ***loop statements*** to make this possible.

- **Loops** enable us to **control how many times we repeat** certain actions.

Loops in Programming

- There are three types of standard loops in (Java) programming:
  - **while**
  - **for**
  - **do while**

- You typically use **for** and **while** loops to iterate over your ArrayList collection,

OR

- you can use another special construct associated with Collections:
  - **for each**

Topic list

1. Grouping Objects
   - Developing a basic personal notebook project using **Collections** e.g. **ArrayList**
2. **Indexing** within Collections
   - Retrieval and removal of objects
3. **Generic classes**
   - e.g. ArrayList
4. **Iteration**
   - Using the for loop
   - Using the while loop
   - Using the **for each** loop

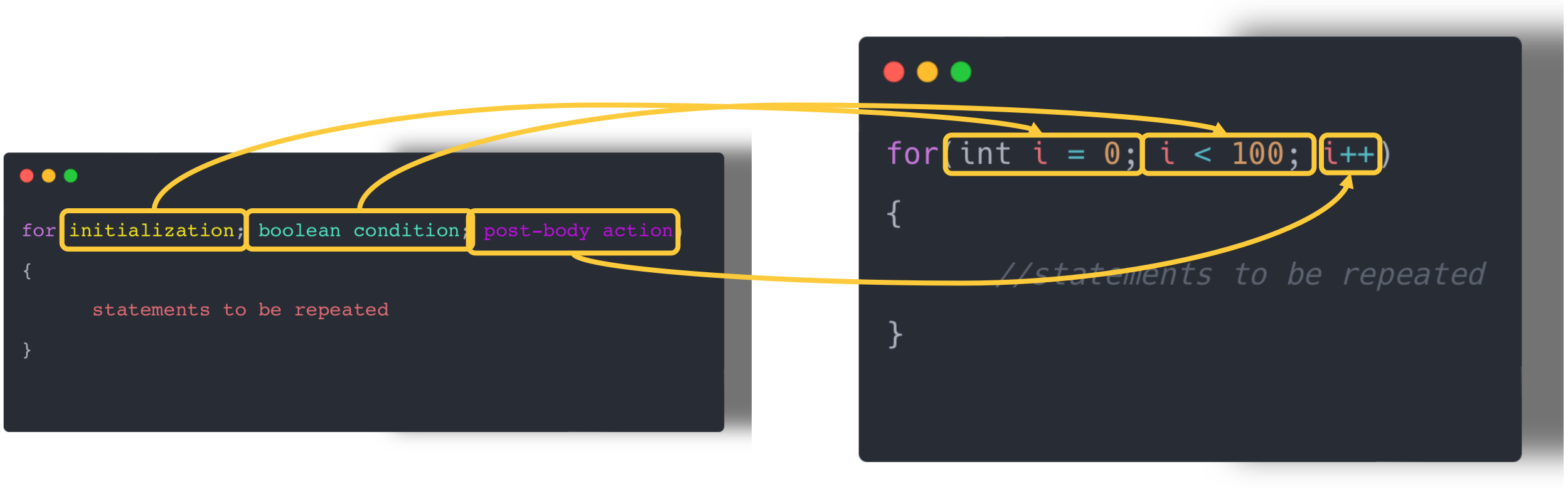
- Next SlideDeck:
  coding a Shop Project that stores an ArrayList of Products.

Recap: For Loop pseudo-code

- General structure of a for loop

```
for(initialization; boolean condition; post-body action)
{
      statements to be repeated
}
```

Recap: For Loop syntax

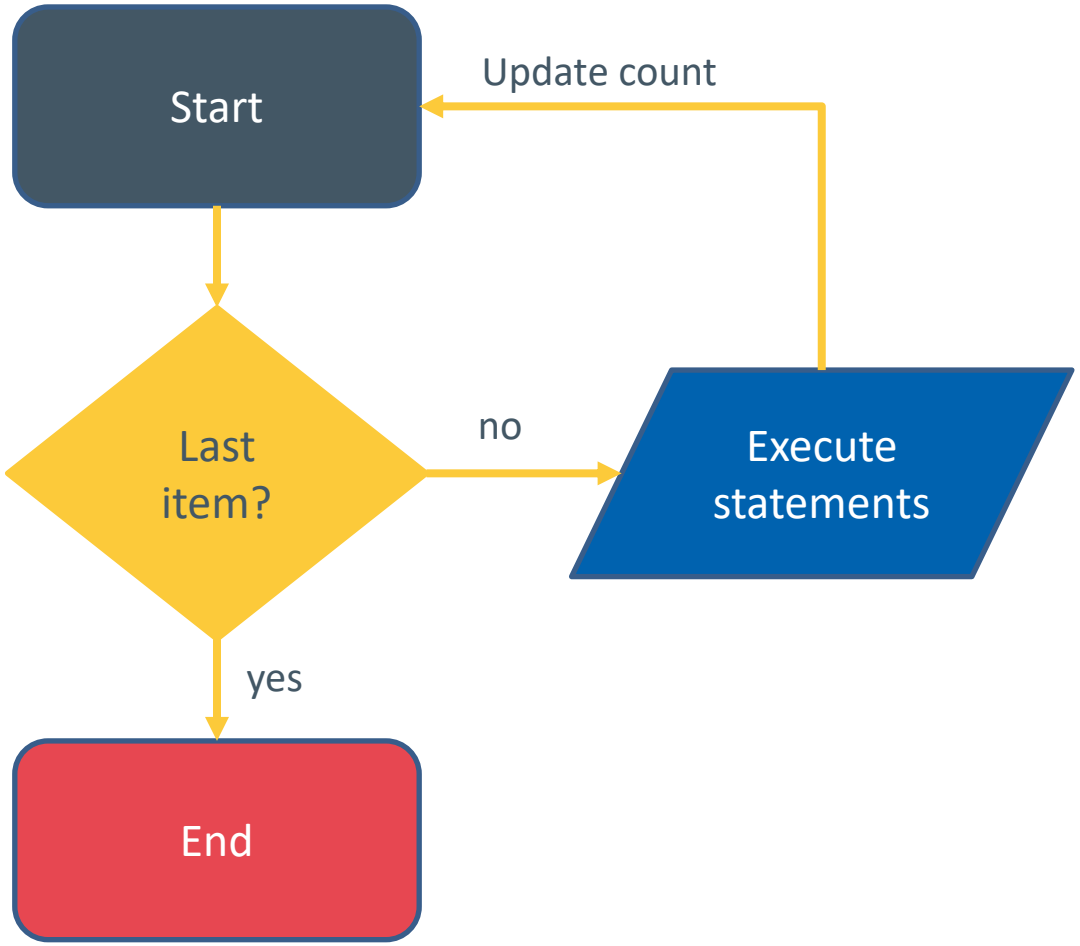


RECAP: FOR LOOP SYNTAX

stage	code	Description
initialization	**int i = 0**	Initialise a loop control variable (LCV) e.g. i It can include a variable declaration.
Boolean condition	**i < 100**	Is a valid boolean condition that typically tests the loop control variable (LCV).
Post body action	**i++**	A change to the loop control variable (LCV). Contains an assignment statement.

```
for(int i = 0; i < 100; i++)
{
        //statements to be repeated
}
```

For Loop Flowchart



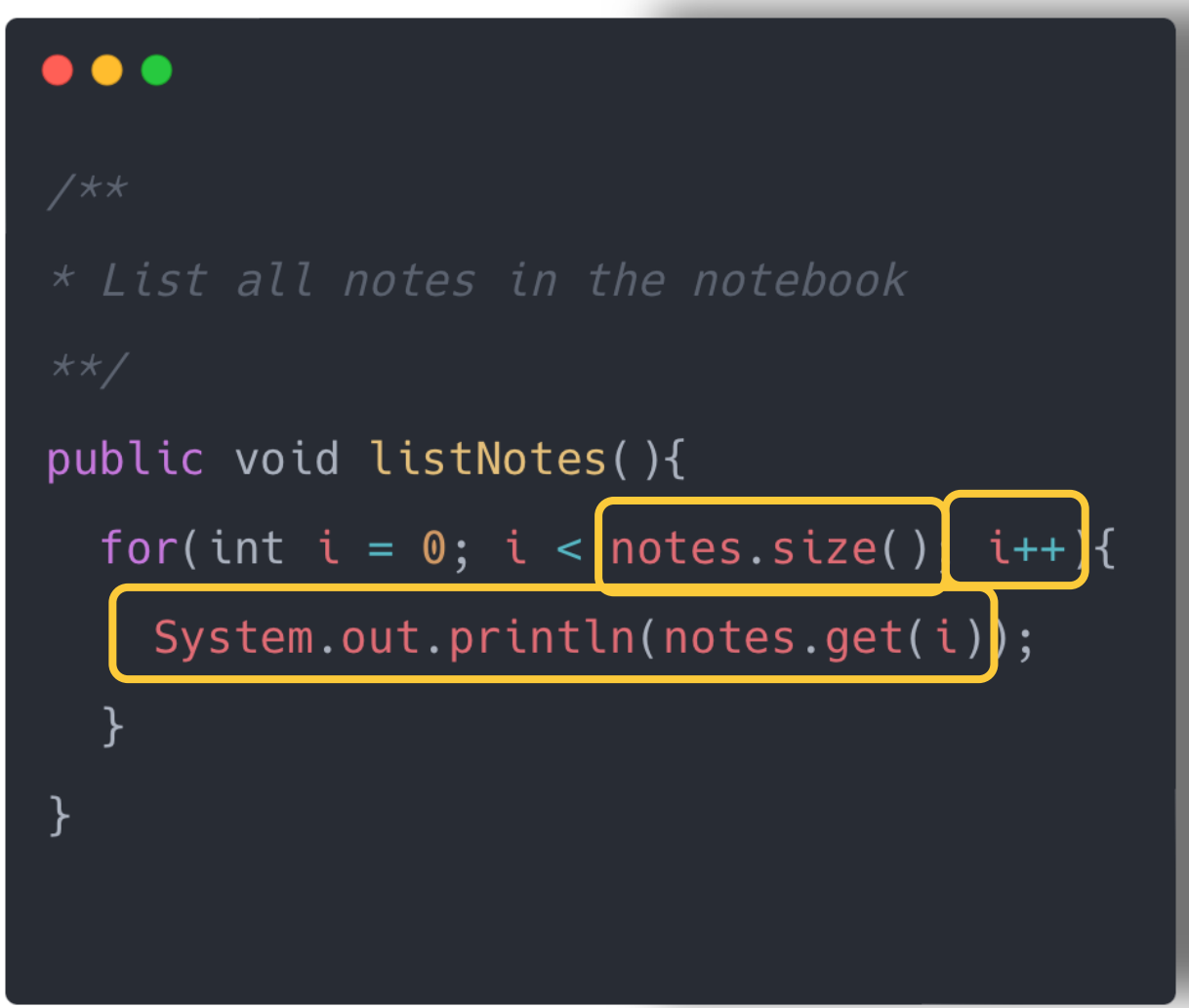

```java
for(int i = 0; i < 4; i++)
{
    System.out.println(i);
}
```

```
/Users/dave/Library/Java/JavaVirtualMachines/
0
1
2
3

Process finished with exit code 0
```

For Loop with ArrayLists (Collections)

- for each value of i less than the size of the collection;
- print the next note;
- and then increment i.

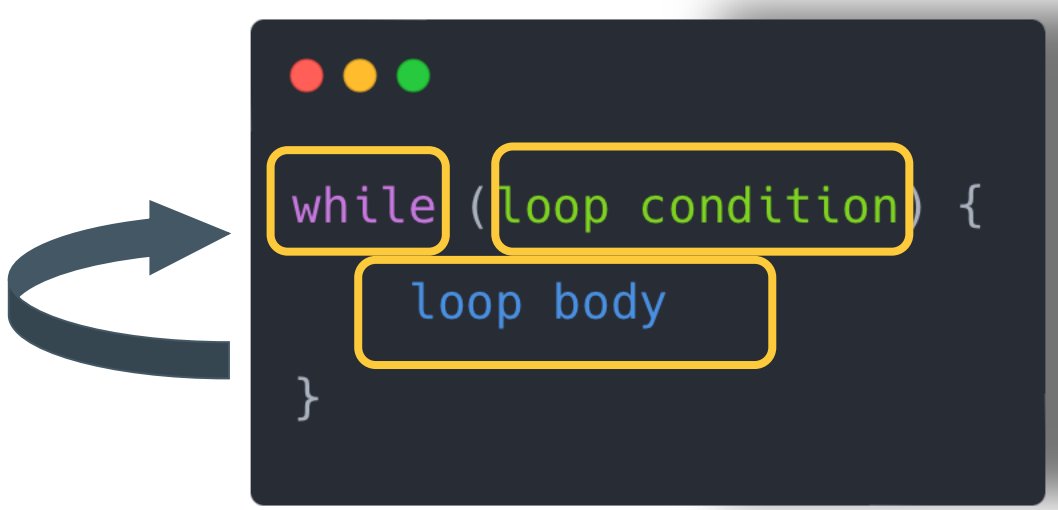


```
/**
* List all notes in the notebook
**/
public void listNotes(){
  for(int i = 0; i < notes.size(); i++){
    System.out.println(notes.get(i));
  }
}
```

While Loop

Recap: While Loop pseudo-code

- General form of a while loop
  - while we wish to continue, do the things in the loop body
- while keyword
- Boolean condition
- Statements to be repeated

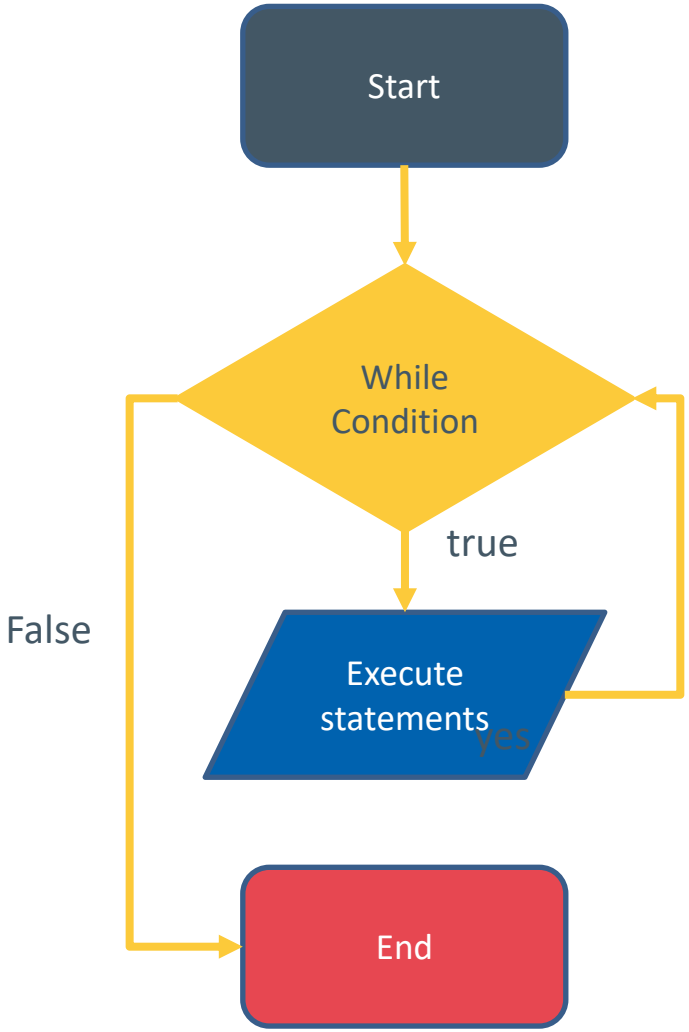


Recap: While Loop Construction

This structure should always be used

```
Declare and initialise loop control variable (LCV)
 while(condition based on LCV)
 {
        “do the job to be repeated”
        “update the LCV”
 }
```

While Loop Flowchart




```java
int i = 1;
while (i <= 10)
{
        System.out.println(i);
        i++;
}
```

While Loop with ArrayLists

- while the value of **i** is less than the size of the collection;
- print the next note;
- and then increment **i** by 1.

```
/**
* List all notes in the notebook
**/
public void listNotes(){
    int i = 0;
    while (i < notes.size())
    {
        System.out.println(notes.get(i));
        i++;
    }
}
```

For vs While -

Variable i is the Loop Control Variable (LCV).

It must be initialised, tested and changed

int i = 0 is the **initialisation**.

i < notes.size() is the **test**.

i++ is the post-body action i.e. the **change**.

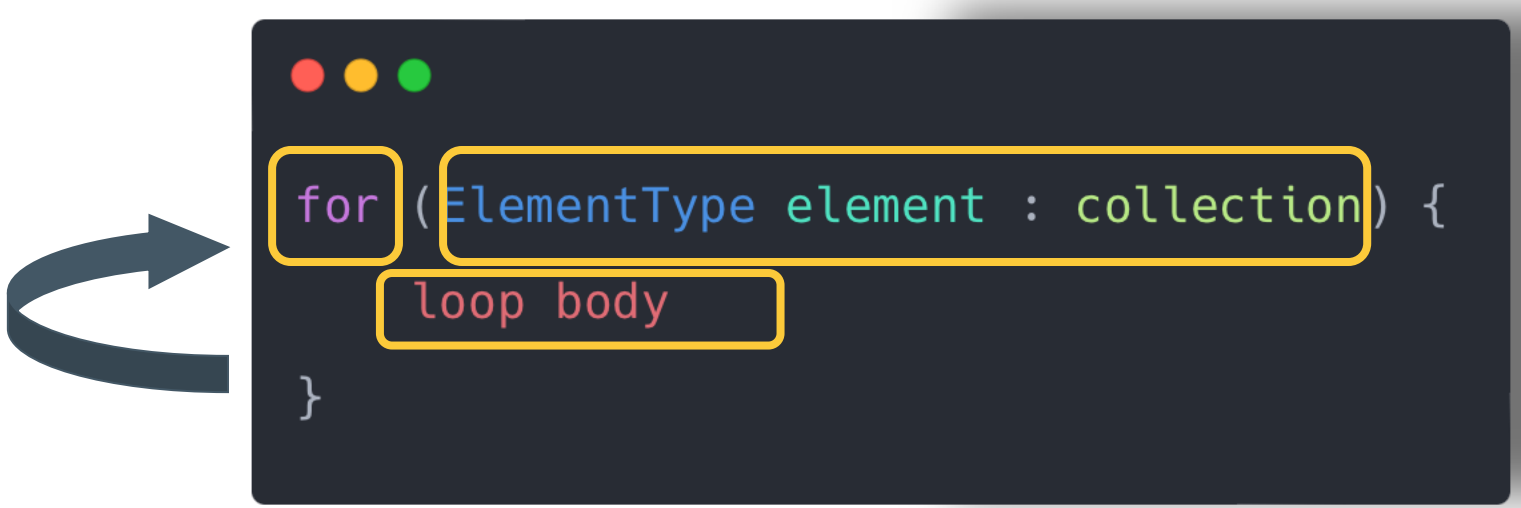
```
/**
* List all notes in the notebook
**/
public void listNotes(){
  for(int i = 0; i < notes.size(); i++){
    System.out.println(notes.get(i));
  }
}
```

```
/**
* List all notes in the notebook
**/
public void listNotes(){
    int i = 0;
    while (i < notes.size())
    {
        System.out.println(notes.get(i));
        i++;
    }
}
```

For Each Loop

FOR EACH LOOP: PSEUDO-CODE

- General form of a for-each loop
    - For each element in collection, do the things in the loop body.
- for keyword
- Loop header
- Statements to be repeated



For each loop with Arraylists

- for each note (of type String) in the notes collection;
  - print out note

```
/**
* list all notes in the notebook
**/
public void listNotes(){
    for (String note : notes) {
        System.out.println(note);
    }
}
```

For each loop

- Can only be used for access
  - you can’t remove the retrieved elements.
- Can only loop forward in single steps.
- Cannot use to compare two collections.

For each vs while

- for-each:
  - **easier to write**.
  - **safer**: it is guaranteed to stop.

- while:
  - we **don't *have* to process the whole collection**.
  - doesn't even have to be used with a collection.
  - take care: could be an *infinite loop*.

ArrayList Collection

- We specify:
  - the type of collection
  - e.g.: notes
  - the type of objects it will contain
    - e.g.: <String>
- We say
  - “notes is an ArrayList of String”

ArrayList Summary

- Java **Collections** Framework
  - **ArrayList**
  - (part of the ***Java util package***)
  - `import java.util.ArrayList;`
  - `private ArrayList <String> notes;  // declares notes as an ArrayList of Strings`
  - `notes = new ArrayList <String>();  // Initialises notes`
  - `notes.add(note);  // add a note to the list`
  - `notes.size();  // returns how notes in collection`
  - `notes.get(noteNumber)  // returns specific element of collection`
  - `notes.remove(noteNumber);  // deletes specific element from collection`
- **Iterating collections**
  - **for each**

```
myArray.length;
```

Comparison

Array

- **Initialisation**
  - String[] names= new String[5];
- **Adding an element**
  - names[0] = “Siobhan”;
- **Accessing an element**
  - System.out.println(names[0]);
- **Modifying an Element**
  - names[0] = “Mary”;
- **Number of elements**
  - names.length; //returns how many elements //array can store
- **Remove an element**
  - names[0] = “”; //delete of element in array can //mean moving elements

ArrayList

- **Initialisation**
  - ArrayList<String> names = new ArrayList<>();
- **Adding an element**
  - names.add(“Siobhan”);
- **Accessing an element**
  - System.out.println(names.get(0));
- **Modifying an Element**
  - names.set(0, “Mary");
- **Number of elements**
  - names.size(); //returns how many elements are //stored
- **Remove an element**
  - names.remove(“Mary"); // Remove by value
    names.remove(0); // Remove by index

References

- https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/lang/String.html
- https://docs.oracle.com/javase/8/docs/api/java/util/ArrayList.html
- http://www.angelikalanger.com/GenericsFAQ/FAQSections/ParameterizedTypes.html#FAQ001
- https://www.programiz.com/java-programming/library/arraylist

Questions?