Programming Fundamentals 2

Introduction to ArrayList

Produced by: Dr. Siobhán Drohan
Mairead Meagher

Waterford Institute *of* Technology
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE

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

Topic list

- Grouping Objects
  - Developing a basic personal notebook project using Collections e.g. ArrayList
- Indexing within Collections
  - Retrieval and removal of objects
- Generic classes e.g. ArrayList
- Iteration
  - Using the for each loop
  - Using the while loop
- Coding ShopV3.0 that stores an ArrayList of Products.

The requirement to group objects

- Many applications involve collections of objects:
  - Personal organizers.
  - Library catalogs.
  - Student-record system.

- The number of items to be stored varies:
  - Items added.
  - Items deleted.

Primitive Arrays

- Primitive arrays can store groups of objects.
- However, the size of the array is fixed when it is created.
- This means that:
  - We have to maintain a variable that tells us how many objects have actually been added to the primitive array e.g. int total;
  - If our array is full, we cannot add any more additional items.
- We need a way of grouping objects that can grow/shrink according to our needs.

Example: A personal notebook

- Notes may be stored.
- Individual notes can be viewed.
- There is no limit to the number of notes.
- It will tell how many notes are stored.

Java class libraries

- Many useful classes.
- We don't have to write everything from scratch.
- Java calls its libraries, *packages*.

Back to the notebook:

- Grouping objects is a recurring requirement.
  - The `java.util` package contains classes for doing this.

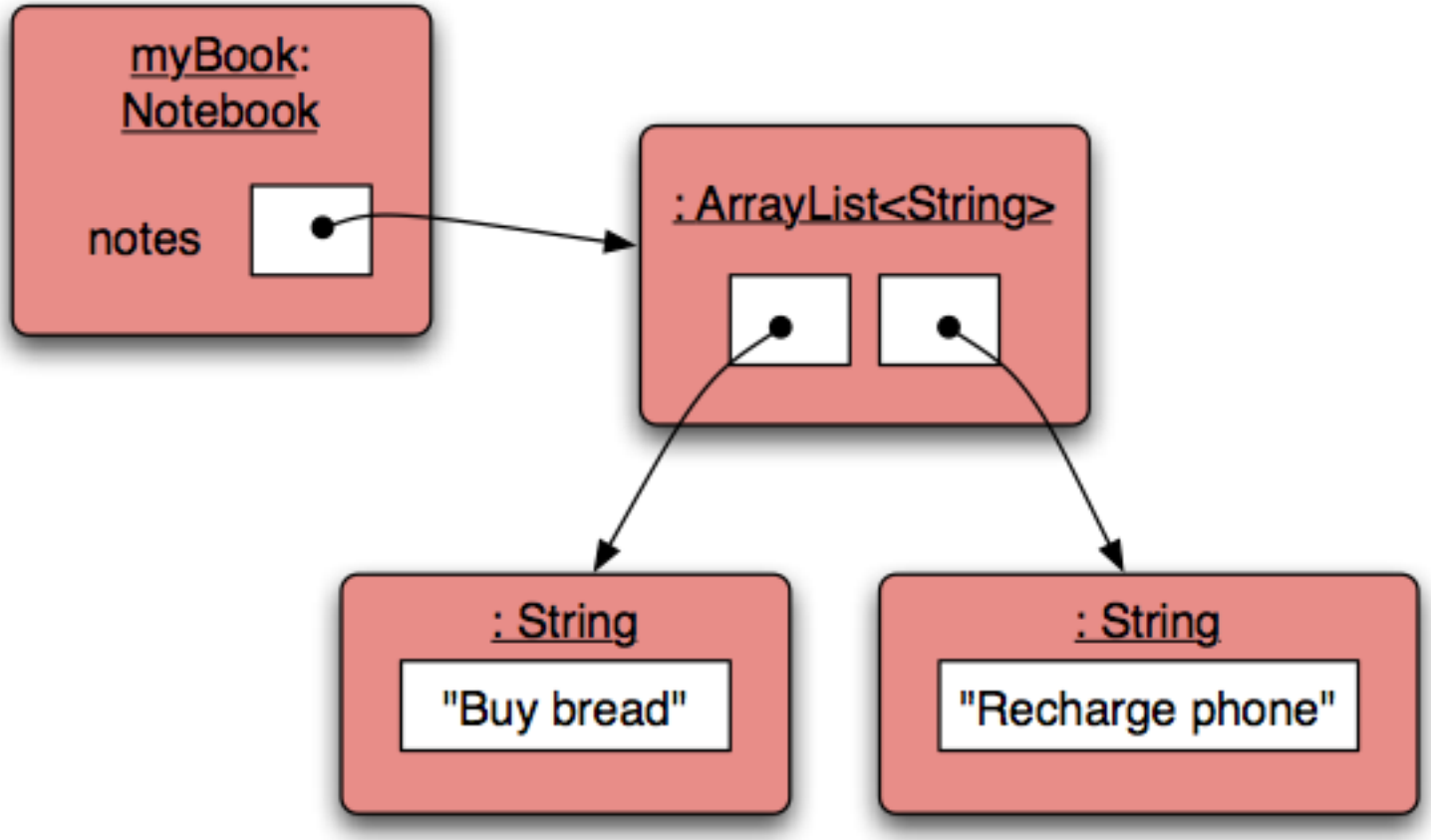
```
import java.util.ArrayList;

public class Notebook
{

        // Storage for an arbitrary number of notes.
        private ArrayList<String> notes;

        // Perform any initialization required for the notebook.
        public Notebook()
        {
                notes = new ArrayList<String>();
        }

}
```

Collections

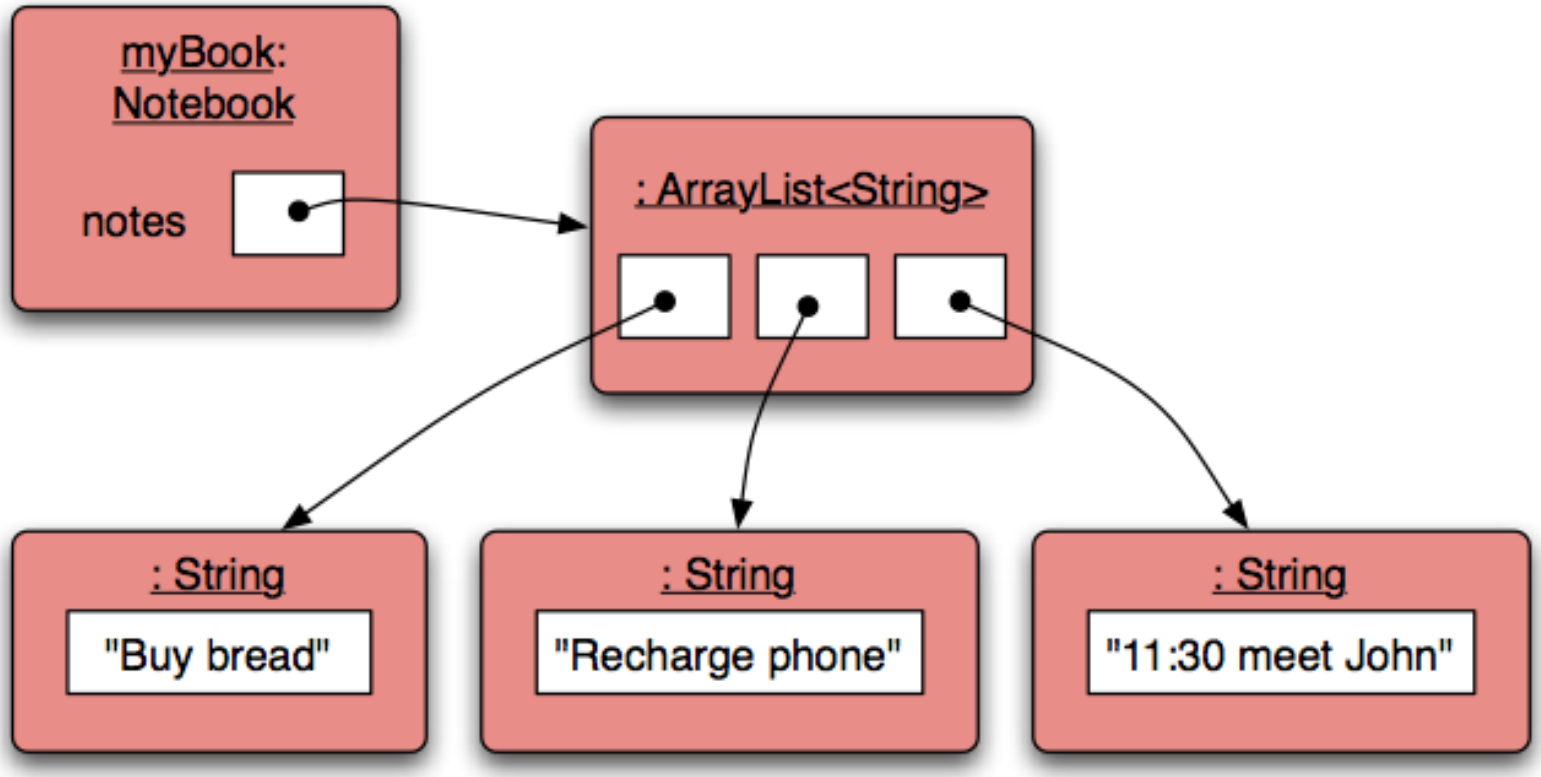
- We specify:
  - the type of collection: `ArrayList`
  - the type of objects it will contain: `<String>`

- We say, “ArrayList of String”.

Object structures with collections



Adding a third note



Features of the collection

- It increases its capacity as necessary.
- It keeps a private count (`size()` accessor).
- It keeps the objects in order.

- Details of how all this is done are hidden.
  - Does that matter? Does not knowing how prevent us from using it?

```
import java.util.ArrayList;

public class Notebook
{
    private ArrayList<String> notes;

    public Notebook(){
            notes = new ArrayList<String>();
    }

    public void storeNote(String note){
        notes.add(note);
    }

    public int numberOfNotes(){
        return notes.size();
    }
}
```

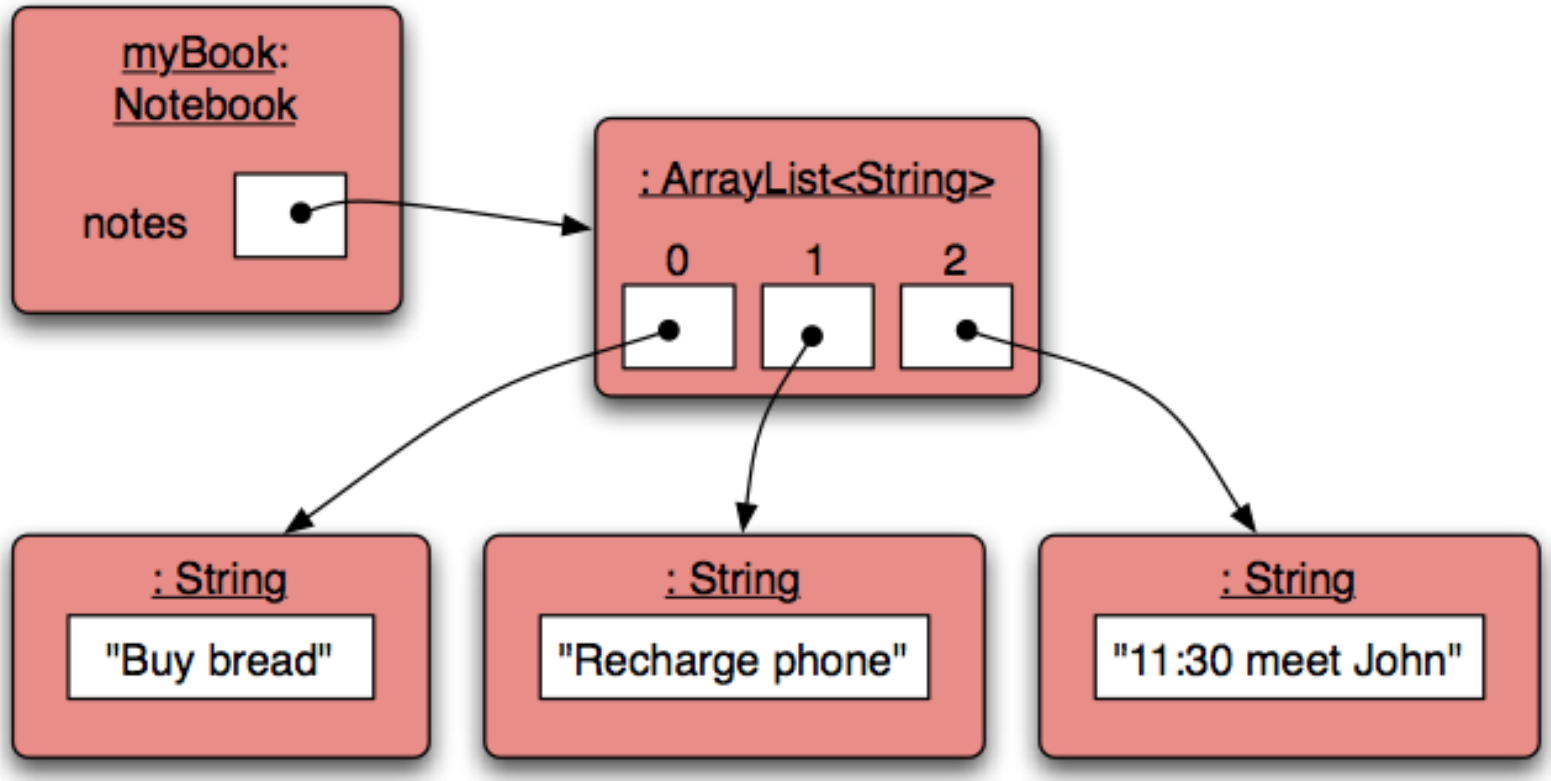
Adding a new note

Returning the number of notes

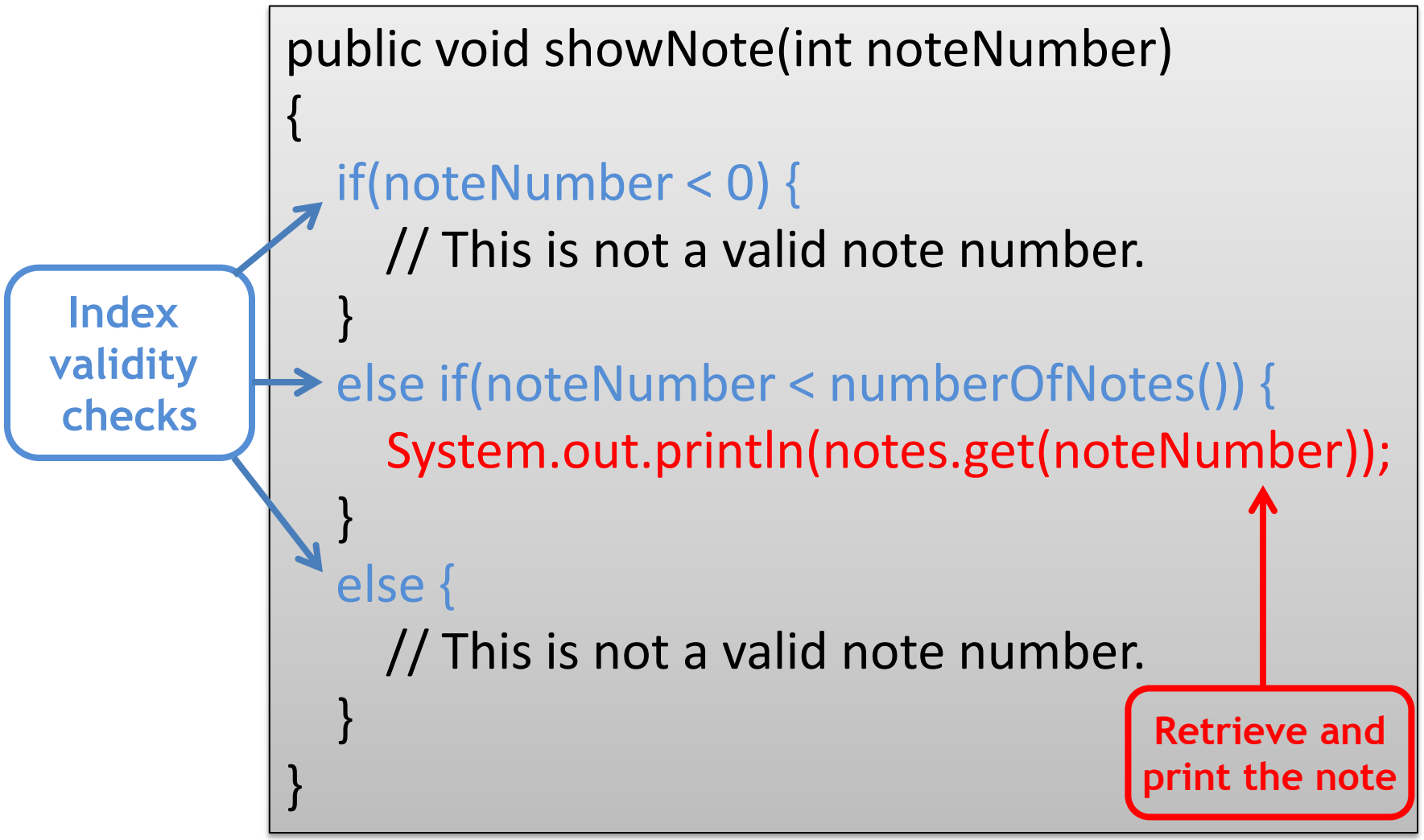
Topic list

- Grouping Objects
  - Developing a basic personal notebook project using Collections e.g. ArrayList
- Indexing within Collections
  - Retrieval and removal of objects
- Generic classes e.g. ArrayList
- Iteration
  - Using the for each loop
  - Using the while loop
- Coding ShopV3.0 that stores an ArrayList of Products.

Index numbering



Retrieving an object



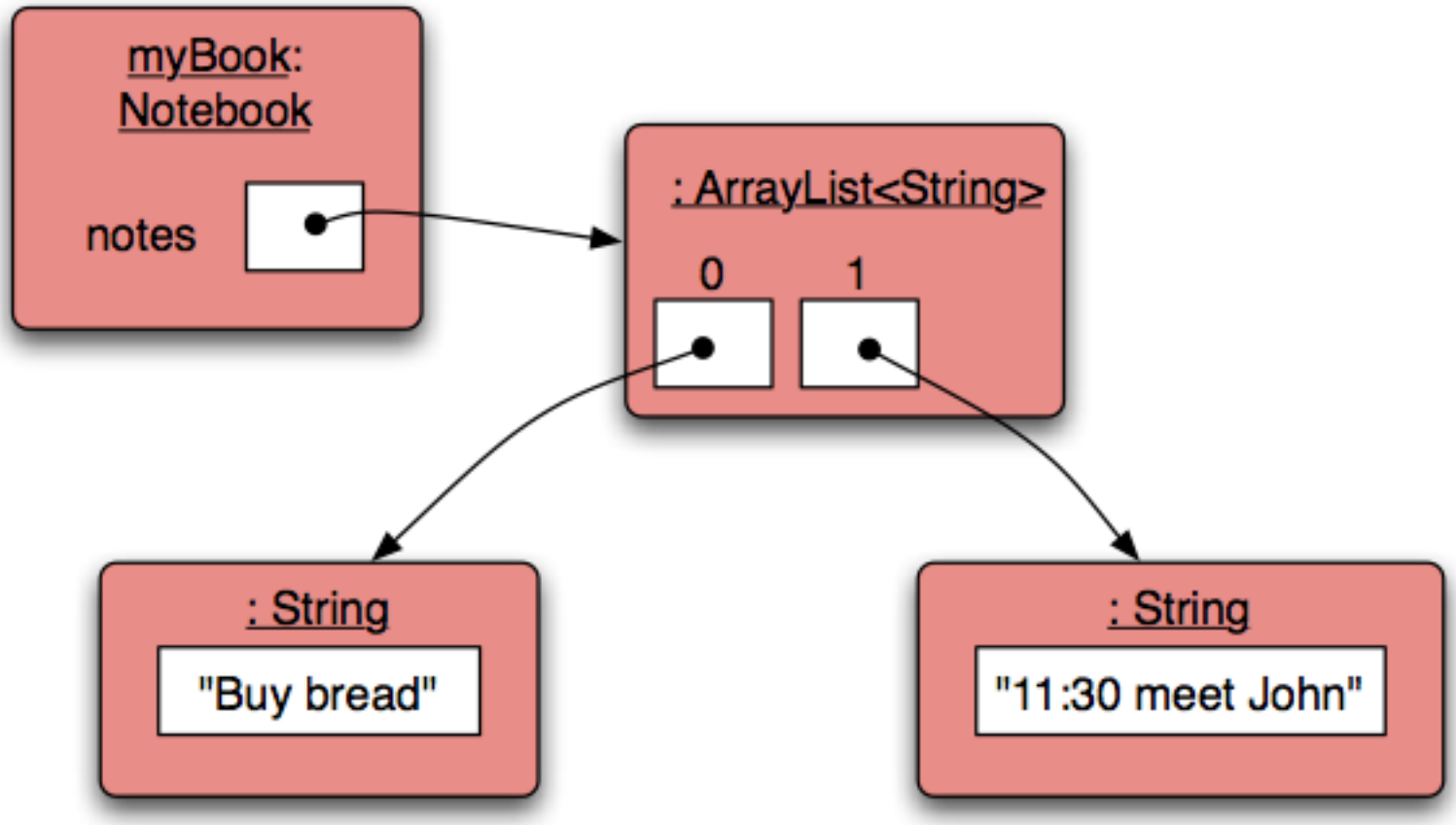
Removing an object

```
public void removeNote(int noteNumber)
  {
    if(noteNumber < 0) {
       // This is not a valid note number, so do nothing.
    }
    else if(noteNumber < numberOfNotes()) {
       // This is a valid note number.
       notes.remove(noteNumber);
    }
    else {
       // This is not a valid note number, so do nothing.
    }
  }
```

Index validity checks

Delete the note at the specific index

Removal may affect numbering



Topic list

- Grouping Objects
  - Developing a basic personal notebook project using Collections e.g. ArrayList
- Indexing within Collections
  - Retrieval and removal of objects
- Generic classes e.g. ArrayList
- Iteration
  - Using the for each loop
  - Using the while loop
- Coding ShopV3.0 that stores an ArrayList of Products.

Generic classes

- Collections are known as *parameterized* or *generic* types.
- `ArrayList` implements list functionality:
  - **`add`**, **`get`**, **`size`**, etc.
- The type parameter says what we want a list of:
  - **`ArrayList<Person>`**
  - **`ArrayList<TicketMachine>`**
  - etc.

Topic list

- Grouping Objects
  - Developing a basic personal notebook project using Collections e.g. ArrayList
- Indexing within Collections
  - Retrieval and removal of objects
- Generic classes e.g. ArrayList
- Iteration
  - Using the for each loop
  - Using the while loop
- Coding ShopV3.0 that stores an ArrayList of Products.

Processing a whole collection (iteration)

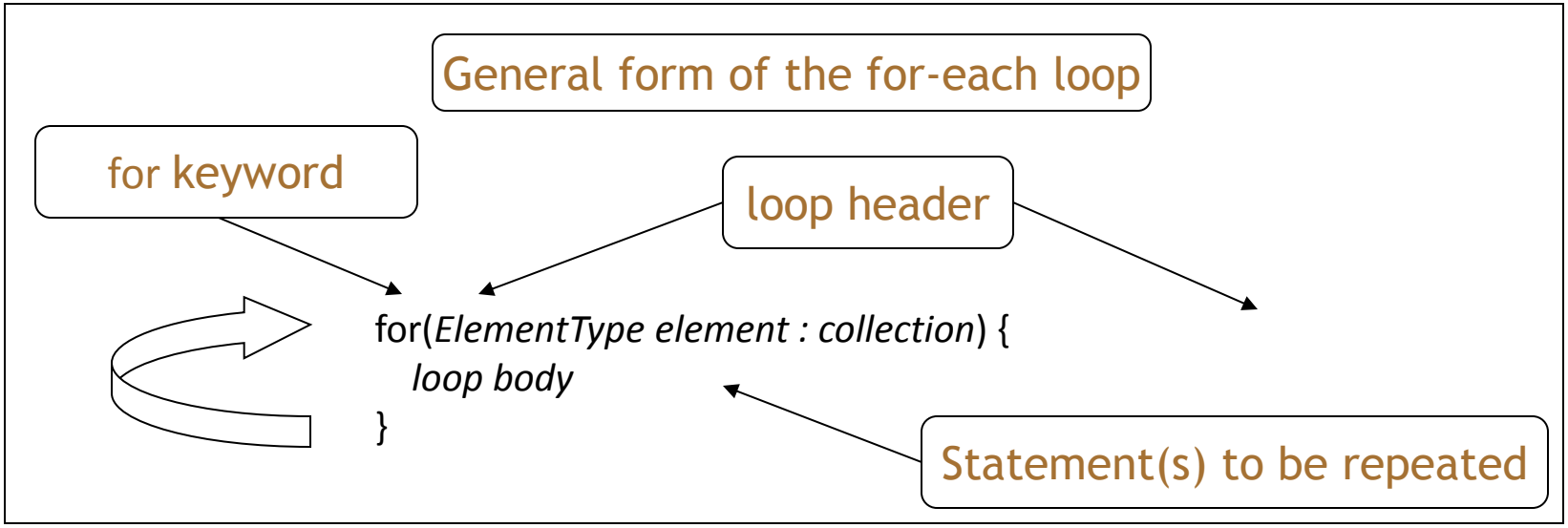
- We often want to perform some actions an arbitrary number of times.
  - E.g., print all the notes in the notebook. How many are there?
- Most programming languages include *loop statements* to make this possible.
- Java has several sorts of loop statement.
  - We will start with its *for-each loop*.

Iteration fundamentals

- We often want to repeat some actions over and over
  - e.g., print all the notes in the notebook.

- Loops provide us with a way to control how many times we repeat those actions.
- With collections, we often want to repeat things once for every object in a particular collection.

For-each loop pseudo code



Pseudo-code expression of the actions of a for-each loop

For each *element* in *collection*, do the things in the *loop body*.

A Java example

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

for each *note* in *notes*, print out *note*

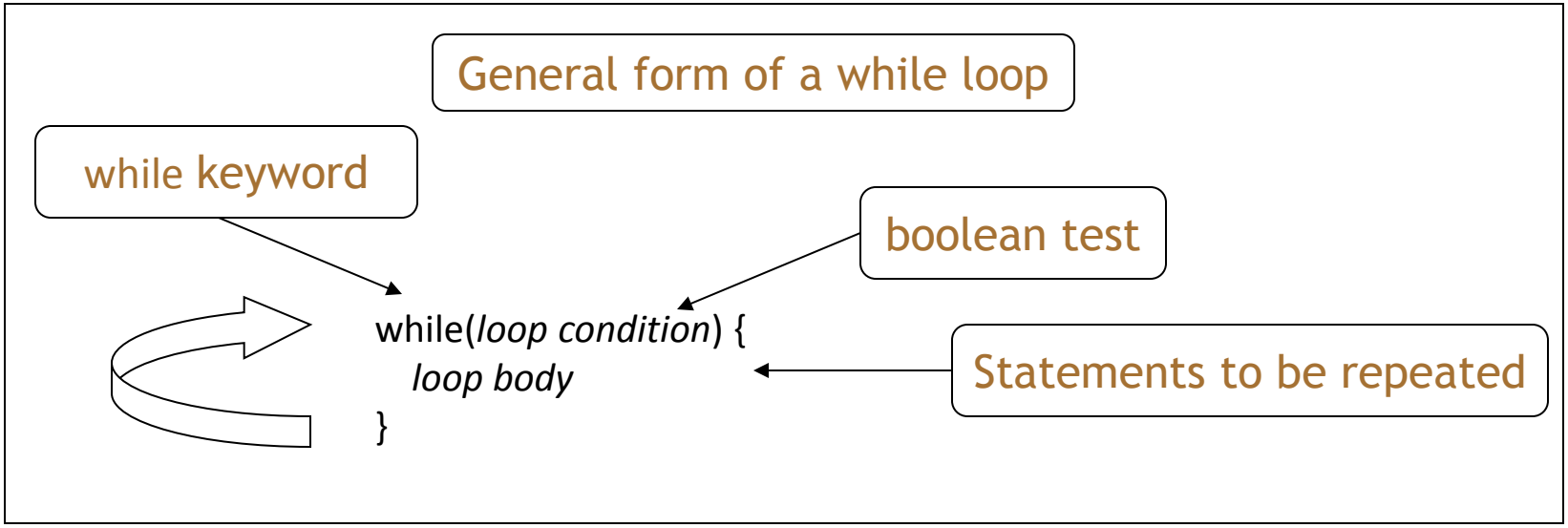
Topic list

- Grouping Objects
  - Developing a basic personal notebook project using Collections e.g. ArrayList
- Indexing within Collections
  - Retrieval and removal of objects
- Generic classes e.g. ArrayList
- Iteration
  - Using the for each loop
  - Using the while loop
- Coding ShopV3.0 that stores an ArrayList of Products.

The while loop

- A for-each loop repeats the loop body for each object in a collection.
- Sometimes we require more variation than this.
- We can use a boolean condition to decide whether or not to keep going.
- A while loop provides this control.

While loop pseudo code



Pseudo-code expression of the actions of a while loop

while we wish to continue, do the things in the loop body

while loop for iterating over a collection

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

Increment *index* by 1

while the value of *index* is less than the size of the collection, print the next note, and then increment *index*

for-each versus 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*.

Waterford Institute *of* Technology
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE

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