Grouping Objects

1 Class Libraries

- These are collections of useful classes
- We don't have to write everything from scratch
- Java calls its libraries packages
- Grouping objects is a recurring requirement

2 Collections

We specify

- The type of collection: ArrayList
- The type of objects it will contain <String>

2.1 Features

- Increases capacity as necessary
- Keeps a private count (size() accessor)
- Keeps the objects in order
- Details of how all this is done are hidden

2.2 Using a collection

```
public class Notebook
{
        private ArrayList<String> notes;
        ...

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

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

2.3 Retrieving an object

```
public void showNote(int noteNumber)
{
        if(noteNumber < 0) {
                // This is not a valid note number.
        }
        else if(noteNumber < numberOfNotes()) {
                System.out.println(notes.get(noteNumber));
        }
        else {
                // This is not a valid note number.
        }
}
```

2.4 Generic Classes

- Collections are known as parameterized or generic types
- ArrayList implements list functionality
- The type of parameter says what we want a list of

3 For-each loop

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

4 While loop

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

5 Equality vs Identity

```
// tests identity
if(input == "bye") {
        ...
}
// tests equality
if(input.equals("bye")) {
        ...
}
```

Identity tests if a variable holds the same instance as another variable
Equality tests if two distinct objects can be used interchangeably

6 Iterators

```
Iterator<ElementType> it = myCollection.iterator();
while(it.hasNext()) {
        call it.next() //to get the next object
        //do something with that object
}
```

6.1 Index vs Iterator

For each loop

- Use if we want to process every element

While loop

- Use if we might want to stop part way through
- Use for repetition that doesn't involve a collection

Iterator object

- Use if we might want to stop part way through
- Often used with collections where indexed access if not very efficient, or impossible

7 Arrays

```
public class LogAnalyzer
{
        private int[] hourCounts; // array variable declaration
        private LogfileReader reader;

        public LogAnalyzer()
        {
                hourCounts = new int[24]; // array object creation
                reader = new LogfileReader();
        }

}
```

7.1 Array length

To get the length of an array use

```
private int[] numbers = { 3, 15, 4, 5 };
int n = numbers.length;
```

Note here that length is not a method

8 For loop

- The for loop is often used to iterate a fixed number of times
- Often used with a variable that changes a fixed amount on each iteration

```
for(int hour = 0; hour < hourCounts.length; hour++) {
        System.out.println(hour + ": " + hourCounts[hour]);
}
```