Arrays

- An array is an indexed collection of variables of the same type

- Declaration
  ```java
  int[] a;
  String[] messages;
  ```

- Initialization
  ```java
  a = new int[25];
  messages = new String[10];
  ```
Arrays - usage

- Creates variables indexed from 0 that can be used individually
- Usage

  ```java
  messages[3] = "Meet you at 5pm";
  ```

- Arrays know their own length

  ```java
  int aLen = a.length;  // not method call
  ```
Control Flow I

- Looping constructs
  - For-loop
    
    ```
    for (int i=0; i<10; i++) {
        ... }
    ```
  - While loop
    
    ```
    while (!isDone) {
        ... }
    ```
  - Do-While loop
    
    ```
    do {
        ... } while (...
    ```
Control Flow II

- Branching constructs
  - If-statement
    ```java
    if (boolean-condition) {
      ...
    } else {
      ...
    }
    ```
  - Switch statement (see Java Basics lecture)
Collection classes

- Arrays have a fixed length whereas often we want a “growable array”
- Java library classes provide large range of “collection classes”
- For a growable array there is `java.util.ArrayList`
Using `ArrayList`

- To avoid using the fully qualified name of the class we can put
  
  ```java
  import java.util.ArrayList;
  ```

- This then allows us to use the “short name” `ArrayList` anywhere in the code

  ```java
  private ArrayList messages;
  ```
Creating an ArrayList

- Constructors
  
  ```java
  ArrayList()
  ArrayList(int initialCapacity)
  ```

- Where do you find these details?
  
  From the online Java API !!
  See CITS1220 or CITS1200 web for links and use Eclipse context help
Methods in `ArrayList`

- Basic methods to *add* objects to the list, *get* (i.e. view) objects from the list and *remove* objects from the list.

- `ArrayList` will *grow* automatically if you add lots of objects (obviously subject to total memory limitations).
What type of things?

- What types of thing can be added?
- If we look at the Java API then it looks very odd
  
  ```java
  boolean add (E o)
  E get (int index)
  ```

- What on earth is this strange type E?
Objects in ArrayLists

- Earlier versions of Java had ArrayLists that permitted any object to be added - in other words the type was Object

```java
ArrayList al = new ArrayList();
al.add("Hello");
al.add(new java.awt.Color(255,10,100));
```
Casting

- When extracting something from an ArrayList, the program needed to know what type it was so that the returned value could be cast

  String s = (String) al.get(0);
Generics

- Untyped ArrayLists are too general, but restricting to a single type is too narrow.

- Java now uses a construct called generics - an ArrayList is defined using a “generic type” E that the programmer can specify.

- Can specify that the list contains “all Strings” or “all Colors” but prevent a mixture.
A list of Strings

private ArrayList<String> ml;
ml = new ArrayList<String>();
ml.add("Dinner at 5pm");

- Now the compiler will only permit objects of the type String to be added to this particular list
HashMap

- Java class HashMap (java.util.HashMap) is a fast and easy to use class representing hash table: a data structure that associates keys with values.
Most important HashMap methods

get(Object key)
returns the value associated with specified key in this hash map, or null if there is no value for this key

put(K key, V value)
associates the specified value with the specified key in this map
Other useful HashMap methods

containsKey(Object key)
  (boolean) returns true if map contains a value for the specified key
values()
  returns a collection of the values contained in this map
keySet()
  returns a set view of the keys contained in this map
remove(Object key)
  removes the mapping for key from this map if present
isEmpty()
  (boolean) returns true if this map contains no key-value mappings
// Creating new HashMap objects
// keys are String, values are Integer HashMap<String, Integer>

wordcount = new HashMap<String, Integer>();
...

// Check if word is in HashMap
if (wordcount.containsKey(word)) {
    // get number of occurrences for this word increment it and put back again
    wordcount.put(word, wordcount.get(word) + 1);
} else {
    // this is first time we see this word, set value '1'
    wordcount.put(word, 1);
}
}