More Basic Java

- Arrays
  - similar syntax to C/C++
    - 0-based
    - BUT: range checked
      - ArrayIndexOutOfBoundsException if illegal access
  - new syntax
    - aType [ ] aVar
    - String [ ] args
  - manipulated by reference
    - created with new: int [ ] a = new int [10];
  - garbage collected when no longer referenced
  - can't extend

More Basic Java

- can be created statically
  - int [ ] knownThings = {7, 3, 8, 13};
- multiple dimensions
  - String [ ] [ ] [ ] a = new String [10] [ ] [ ];
  - String [ ] [ ] [ ] a = new String [10] [10] [ ];
  - String [ ] [ ] [ ] a = new String [10] [ ] [10];
    - can't figure out how to access elements
  - example:

```
String [] [] param_info =
  {
  {"foreground", "Color", "foreground color"},
  {"background", "Color", "background color"},
  {"message", "String", "the message to display"},
  };
```

More Basic Java

- arrays need not be regular; ragged arrays also possible:
  - static

```
static byte [] [] twodim = {{1, 2}, {3, 4, 5}, {6, 7, 8, 9}};
```

  - dynamic:

```
short [] [] triangle = new short [10] [];
for (int i = 0; i < triangle.length; i ++)
  {
  triangle [i] = new short [i + 1];
  for (int j = 0; j < i + 1; j ++)
    triangle [i][j] = i + j;
  }
```

- note use of a.length
- no sizeof
  - unnecessary, since all primitive sizes defined

More Basic Java

- array == object with special syntax support
- some standard object things:
  - a = a1 copies reference only; still one object
    - System.arraycopy (src, srcPos, dst, dstcPos, num);
  - int [ ] copy = (int [ ]) orig.clone ();
    - can raise an exception if an object isn't cloneable
  - a == a1 compares references
    - a.equals (a1);

More Basic Java

- Strings
  - first-class objects, not just null-terminated arrays plus convention for use
    - many methods
      - '+', charAt, length, substring, indexOf, etc.
  - actually two types of String *(in java.lang)*:
    - String
      - immutable
      - operations by creating new object—"cut & paste"
    - StringBuffer
      - contents alterable & dynamically sized
      - in-situ modification—more efficient
  - 16-bit Unicode v2.0

More Basic Java

- given "…", compiler allocates a *String* object
- no ## stringify operator as in ANSI C
  - 'cos no preprocessor
- no \↵
  - no preprocessor
  - use '+'
- trim () saves space
  - " 1234 ".trim() gives "1234"
- most standard classes have toString method
  - often produces gibberish
  - useful when primitive types involved
    - compiler often knows to call this

More Basic Java

- Flow of control
  - mostly as in C/C++
  - main difference: tests must evaluate to boolean
    - 1 != true; 0 != false
    - use expr != 0, not (boolean) expr
    - while (x -- > 0) not while (x--)
    - if…, while…, do…, for… affected by this
  - switch
    - label types may be: byte, char, short, int or long
      - *not* boolean
    - break needed
    - default entry

More Basic Java

- for loop semantics slightly altered
  - , only in 1st & 3rd sections
  - local variable *strictly* local
    - but mustn't override another name in scope
- no goto provided
  - but is a reserved word
- labelled statements; break & continue

```
TEST:
if (check (i))
  {
  for (int j = 0; j < 10; j ++)
    {
    if (j > i) break;
    if (a [i][j] == null)
     break TEST;
    }
  }
```