Arrays

- An array is a data structure that stores a homogeneous collection of elements
- Each element of the array has an associated integer index
- An array storing n elements has valid index values of 0 through $n-1$
- An variable referring to an array is defined as

```
TYPE[] A; // array variable of type TYPE
int[] val; // array variable of type int
Complex[] c; // array variable of type
    Complex
```

- An array is created using the `new` operator

```
val = new int[100]; // allocate 100 int
    elements
c = new Complex[5]; // allocate 5 Complex
    elements
```

Using Arrays

- Any integer expression can be used as the index

```
int [] val = new int[5]; // 5 int elements
val[0] = 1; // assign 1 to element at index 0
val[3] = 2; // assign 2 to element at index 3
val[-1] = 0; // error: bad index
val[5] = 100; // error: bad index
int i = 4;
val[i] = 5; // assign 5 to element at index 4
// assign -1 to element at index 2
val[i-2] = -1;
// assign 13 to element at index 1
val[(i*4)%5] = 13;
// What does this do?
val[(val[i])%5] = val[(i+1)%5];
```

Loops and Arrays

- For loops and arrays are BFF's

```
int n = 10;
int [] val = new int[n]; // n int elements

// fill array
for (int i = 0;  i < 5;  i++)
  val[i] = i;

// print array
for (int i = 0;  i < 5;  i++)
  println(i+" "+val[i]);
```

More fun with arrays

- An array can be declared an initialized in one step
- `arrayName.length` returns the number of elements stored in the array

```
int [] primes = {2, 3, 5, 7, 11, 13, 17, 19};

// print array
for (int i = 0;  i < primes.length;  i++)
  println(i+" "+primes[i]);
```

Odds and Ends

- Using an array of objects is typically better than multiple parallel arrays
- Array element selection `[ ]` has precedence over object member selection `.`
  - `array[i].f()` applies `f` to `array[i]`
- Arrays can be used as function parameters, the reference is sent
- Arrays can be used as function return values, the reference is returned