- Previous Lecture:
  - 1-d array of primitive-typed things (e.g., array of numbers)
- Today's Lecture:
  - Linear search
  - Selection sort
  - Binary search (to be discussed in section)
  - 1-d array of objects
- Reading:
  - Sec 6.2

Pattern for processing an array

```
// assume an array has been
// created and is referred to by
// variable A

for (int i=0; i<A.length; i++) {

  // perform some process
  // (on A[i])

}
```


Example

```
// Given int array v and int z,
// how many times does z appear
// in v?

int count= 0;  //Count of z so far
```



```
// Linear Search:
// f is index of first occurrence of z in array a
int k= 0;
while ( a[k]!=z  &&  k<a.length )
   k++;
if (k==a.length)  f= -1; //signal for z not found
else  f= k;
```

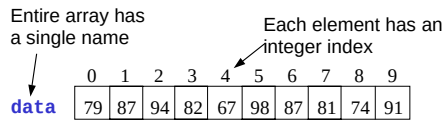
a. Correct
b. Incorrect: f is off by one
c. Incorrect: while condition is wrong
d. Incorrect: if conditional is wrong


```
public static void selectSort(double[] a){


  // Loop from first to second last element
  // Index i: 1st cell in unsorted segment
  for (int i=0; i<a.length-1; i++){
      // Find index of min in unsorted segment




      // Swap i-th element with min


  }
}
```

```
public static void selectSort(double[] a){
  int min_loc;  // index of min in unsorted segment
  double temp;
  // Loop from first to second last element
  // Index i: 1st cell in unsorted segment
  for (int i=0; i<a.length-1; i++){
      // Find index of min in unsorted segment
         min_loc = i;
         // Compare each element j in unsorted
         // segment with min found so far
            for (int j=i+1; j<a.length; j++)
                if (a[j]<a[min_loc])
                    min_loc= j;
      // Swap i-th element with min
         temp= a[min_loc];
         a[min_loc]= a[i];
         a[i]= temp;
  }
}
```

Array of primitive-typed values

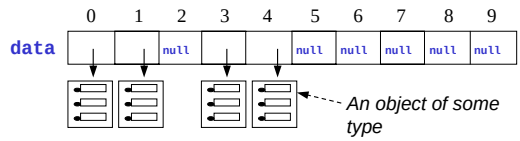
- An array is an object
- An array is an ordered list of values (or objects)
- Each element is of the same type



An array of size N is indexed from 0 to N-1

Array of objects

- An array is an object
- Elements of an array can be object references
- Each element is of the same type



An array of size N is indexed from 0 to N-1

Creating an array of objects

Three steps:

1. Declare array reference variable
   ```
   Interval[] series;
   ```
2. Instantiate array of object references
   ```
   series= new Interval[4];
   ```
3. Instantiate individual objects
   ```
   series[0]= new Interval(0,5);
   series[1]= new Interval(1,7);
   ```


Many `Interval`s

- Class `ManyIntervals` is a client of class `Interval`.
- Create an array of `Interval` objects with random `base` and `width` values. Use integer values.
- Find the `Interval` with the highest endpoint.
- Search for the first `Interval` that has a specific endpoint value


```
class Interval {

  private double base;  // low end
  private double width; // interval width

  public Interval(double base, double w){
    this.base = base;
    width = w;
  }

  public double getEnd() { return base+width; }

  //other methods
}
```

```
public class ManyIntervals{

  public static void main(String[] args) {

    int n= 4; //number of Intervals to create
    int H= 5; //highest value for base, range
    int L= 1; //lowest value for base, range

    //Set of Intervals
    Interval[] set=









    //Find Interval with highest endpoint









    System.out.println("Interval with highest endpoint: " +         );


    //Find 1st Interval with endpoint 6
    int target= 6;









  } //method main

} //class ManyIntervals
```