

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

November 10, 2005

Lecture 22

2

```
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++){

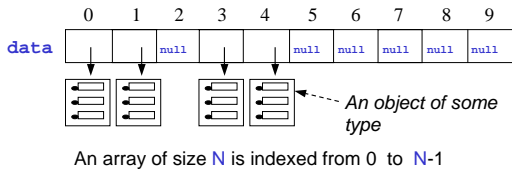
        // Find index of min in unsorted segment

        // Swap i-th element with min

    }
}
```

Array of objects

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



November 10, 2005

Lecture 22

8

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);
```

November 10, 2005

Lecture 22

9

Many Intervals

- 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

November 10, 2005

Lecture 22

12

```
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
}
```

November 10, 2005

Lecture 22

13

```
/* Sort the values in array a in non-descending order using the
 * SELECTION SORT algorithm
 */
public static void selectSort(double[] a){

    // loop from first to second last element
    // i is the start of the unsorted segment
    for (int i=0; i<a.length-1; i++){

        // find index of min in unsorted segment

        // swap ith element with min

    }
} //method selectSort
```

```
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;
```