Topics: Selection sort, array of objects

Reading: review Sec 8.5, 8.6

Selection Sort

```
/** Sort array in non-descending order */
public static void selectSort(double[] array)
```

Array of objects

- Elements of an array can be object references
- Three steps: (1) declaration of the array reference variable, (2) creation (instantiation) of the array of object references, and (3) instantiation of individual objects
- E.g., the statement below gets space to store 10 **`Interval`** references (assuming a **`Interval`** class is defined):

  ```
  Interval[] series = new Interval[10];
  ```

  The individual **`Interval`** objects need to be created separately:

  ```
  series[0] = new Interval();
  series[1] = new Interval();
  . . .
  ```

Example 1

```
/** Organize data for any Person:  name, age */
public class Person {
       private String name;
       private int age;
       public final static int LEGALage=18;

       /** Constructor */
       public Person(String name, int age) {
              this.name = name;
              this.age = age;
       }

       /** = this Person is an adult
       public boolean isAdult() { return age >= LEGALage; }

       /** = String description of this Person */
       public String toString() { return name + " is " + age; }
} // class Person

/* Client class that uses Person class: create a collection of Person data */
public class Record {
       public static void main(String[] args) {
              int size = 100;     //max length of record

              //declare reference variable for array (of Person objects)



              //instantiate array of Person references



              //create Person objects
                 record[0] = new Person("Daisy", 19);
                 record[1] = new Person("Rob", 18);
                 record[2] = new Person("Mary", 16);

              //report only the adults
                 for (int i=0; i<3; i++)
                    if ( record[i].isAdult() )
                       System.out.println(record[i]);
       } // method main
} // class Record
```

Beware of `null` *references*

```
// Suppose we loop through entire
// array.  Then we must first check
// for existence of object BEFORE
// accessing an object's instance
// method
for (int i=0; i<size; i++)
  if (

                                    )
     System.out.println(record[i]);
```

Example 2, recall `Interval` class

Write a class **`ManyIntervals`** that is a client of class **`Interval`**. In class **`ManyIntervals`**, create an array of **`Interval`** objects with random *intger* **`base`** and **`width`** values, find the **`Interval`** with the highest endpoint, and search for the first **`Interval`** that has a specific endpoint value. Some additional parameters are given below.

```
public class ManyIntervals {

  public static void main(String[] args) {

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

    //Set of Intervals









    //Find Interval with highest endpoint









    //Find 1st Interval with endpoint 6









  }//method main
}//class ManyIntervals
```