Previous Lecture:
- 1-dimensional array (numeric)
- Selection Sort

Today’s Lecture:
- Array of objects

Reading (JV):
- Sec 7.3, 7.4

Announcement:
- Prelim 3 tonight, 7:30-9pm, Statler Aud.

---

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

Entire array has a single name
Each element has an integer index

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

---

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

Entire array has a single name
Each element has an integer index

---

Creating an array of objects

Three steps:
1. Declare array reference variable
   `Interval[] series;`
2. Instantiate array of object references
   `series= new Interval[10];`
3. Instantiate individual objects
   `series[0]= new Interval(0,0);`
   `series[1]= new Interval(0,0);`
Example

- We have a `Person` class to organize data about a `Person`:
  - Name
  - Age
  - ...

- Create a `Record` class that is the client of class `Person`:
  - Create an array of `Persons`
  - Enter data

```java
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
```

```java
public class Record {
    public static void main(String[] args) {
        int size = 100;  //max record length
        //array of Person references:
        Person[] record= new Person[size];
        //Person objects
        record[0]= new Person("Dana", 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].toString());
    } //method main
} //class Record
```
Beware of null references

Sometimes, you need to first check that an object exists before you try to access instance methods (or variables). E.g., suppose the for loop iterates to size, not count:

```
// report only the adults
for (int i=0; i<size; i++ )
    if ( record[i].isAdult() )
        System.out.println(record[i]);
```

```
public class Record2 {
    public static void main(String[] args) {
        int size= 100; // max record length
        int count = 0; // # entries so far
        String name; // a person's name
        int age; // a person's age

        // array of Person references:
        Person[] record= new Person[size];

        // read data and create Person objects
        System.out.println("Enter data");
        name= Keyboard.readString();
        age= Keyboard.readInt();
        while (age >= 0) {
            record[count] = new Person(name, age);
            count++;
            name= Keyboard.readString();
            age= Keyboard.readInt();
        }

        // report only the adults
        for (int i=0; i<count; i++)
            if ( record[i].isAdult() )
                System.out.println(record[i]);

    } // method main
} // class Record2
```

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

```
class Interval {
    private double base; // low end
    private double width; // interval width

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

    public double getEnd() { ... }
    public boolean isIn(Interval o) { ... }
    public static Interval overlap(
        Interval a, Interval b) { ... }
    public String toString() { ... }
}
```