Lecture 27

- Covers
  - Programming with arrays
- Reading: Savitch 6.2, 6.3

Array elements as arguments

- When we write methods that take arguments, we may wish to pass in variables from array collections as those arguments
- For example, a maximum method may take two integer arguments

Example

```
public static int maximum (int num1, int num2)
{
    if (num1 > num2)
    {
        return num1;
    }
    else
    {
        return num2;
    }
}
```

Example

- If we have an integer array, we may wish to find the maximum of two of the values in the array
- We pass them into the method as arguments by specifying the elements with the array name and their indexes
- E.g.
  - To find the maximum of the first two elements of the integer array myArray

```
int max = maximum(myArray[0], myArray[1]);
```

Example

- To find the maximum of all the elements in the array, we can use a for loop

```
int max = number[0];
for (int i = 1; i < number.length; ++i)
{
    max = maximum(max, number[i]);
}
```

Example

- In the horizontal bar chart from the previous lecture, we have a nested for loop
- To have a line in the bar chart displayed for each of the numbers, we used a for loop to output the correct number of stars

Example

```
for (int i = 0; i < 5; ++i)
{
    for (int j = 0; j < number[i]; ++j)
    {
        System.out.print("* ");
    }
    System.out.println( );
}
```

Class exercise

- Write a method displayLine to display a single row of stars
- It should take an integer parameter

Class exercise

- Rewrite the for loop to display the bar chart so that it uses the displayLine method

Array parameters

- Sometimes we wish to pass an entire array as an argument to a method
- We have to declare the parameter type as an array

```
public static void doSomething(int[ ] myArray)
{ …
}
```

- And pass in the entire array

```
int[ ] array1 = new int[12];
…
doSomething(array1);
```

- Note that we specify the entire array by name - no subscript operators

Example

- We can convert the horizontal bar chart program into a method that takes an array as an argument

Example

```
public static void displayHorizontalBarChart(int[ ] number)
{
    for (int row = 0; row < number.length; ++row)
    {
        displayLine(number[row]);
    }
}
    …
    System.out.print("Enter the number of values in the bar chart: ");
    int arraySize = keyboard.nextInt( );
    int[ ] number = new int[arraySize];
    System.out.print("Enter " + arraySize + " integers: ");
    for (int row = 0; row < number.length; ++row)
    {
        number[row] = keyboard.nextInt( );

    }
   displayHorizontalBarChart(number);
```

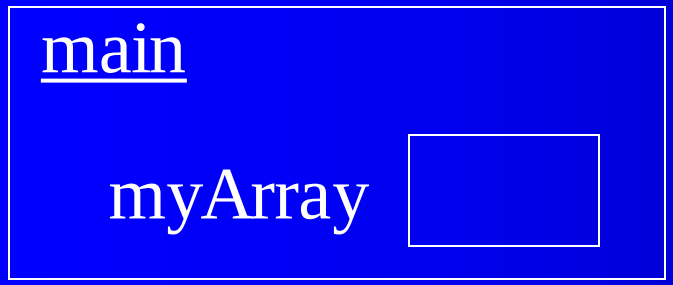
Methods that return an array

- Methods can be made to return an array
- The return type must be indicated as an array type and the return value must be an array of the same base type
- The returned array object can be referred to by an array reference in the calling method

```
public double[ ] createArray( )
{
   …
}

double[ ] myArray = createArray( );
```

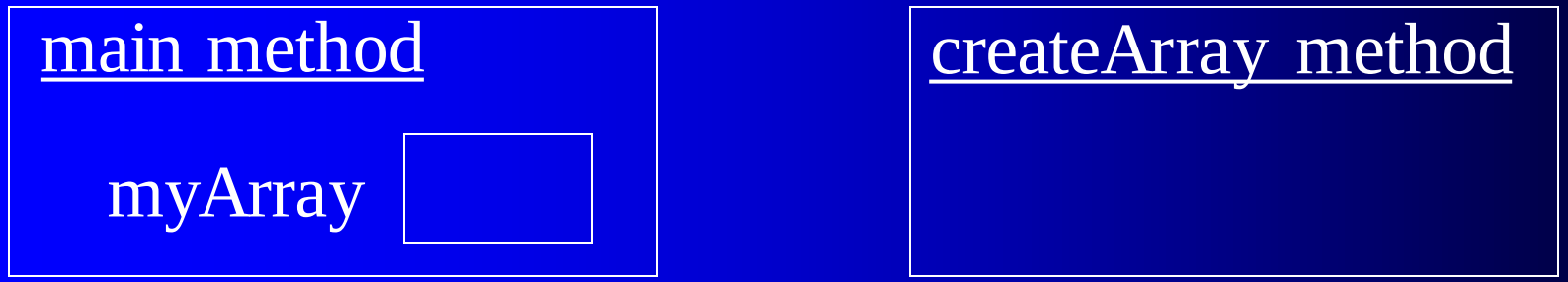
Methods that return an array

```
public double[ ] createArray( )
{
    double[ ] a1 = new double[3];
    a1[0] = 11.1;
    a1[1] = 22.2;
    a1[2] = 33.3;
    return a1;
}

double[ ] myArray = createArray( );
```



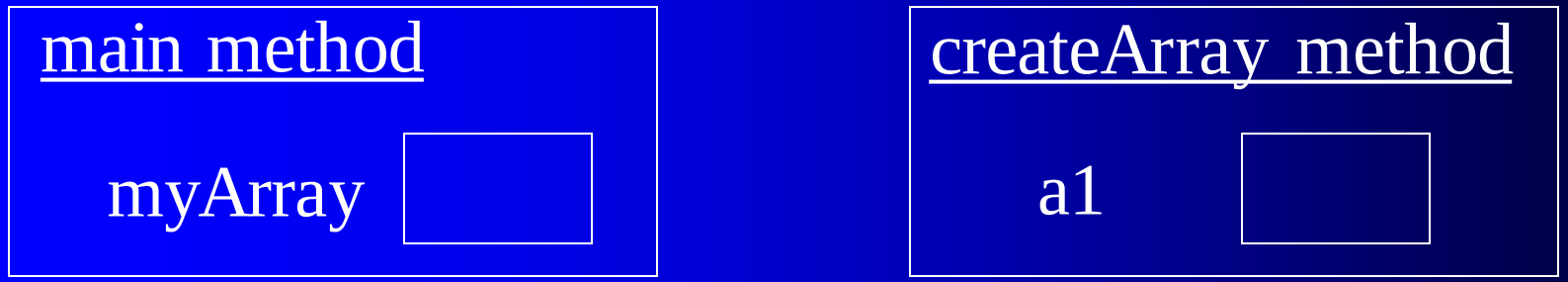
Methods that return an array

```
public double[ ] createArray( )
{
    double[ ] a1 = new double[3];
    a1[0] = 11.1;
    a1[1] = 22.2;
    a1[2] = 33.3;
    return a1;
}

double[ ] myArray = createArray( );
```



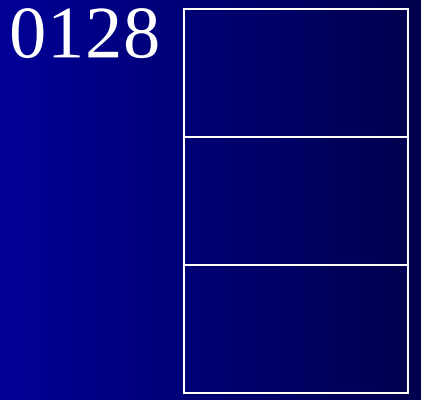
Methods that return an array

```
public double[ ] createArray( )
{
    double[ ] a1 = new double[3];
    a1[0] = 11.1;
    a1[1] = 22.2;
    a1[2] = 33.3;
    return a1;
}

double[ ] myArray = createArray( );
```

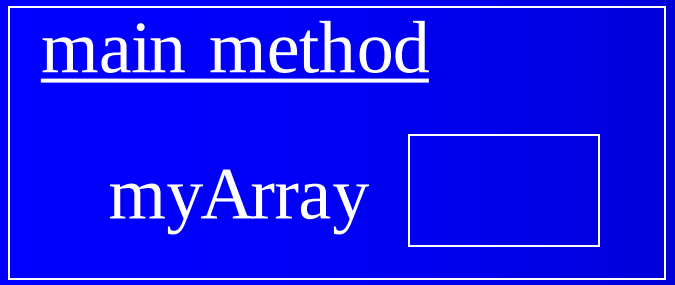


Methods that return an array

```
public double[ ] createArray( )
{
    double[ ] a1 = new double[3];
    a1[0] = 11.1;
    a1[1] = 22.2;
    a1[2] = 33.3;
    return a1;
}
```

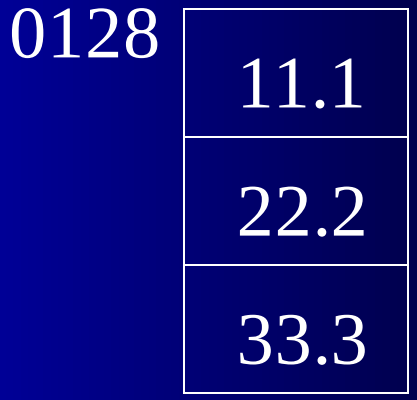



```
double[ ] myArray = createArray( );
```

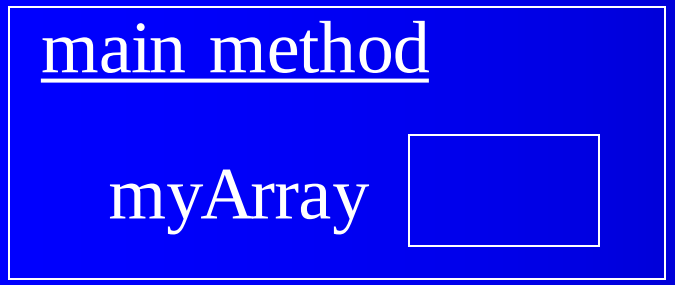


Methods that return an array

```
public double[ ] createArray( )
{
    double[ ] a1 = new double[3];
    a1[0] = 11.1;
    a1[1] = 22.2;
    a1[2] = 33.3;
    return a1;
}
```

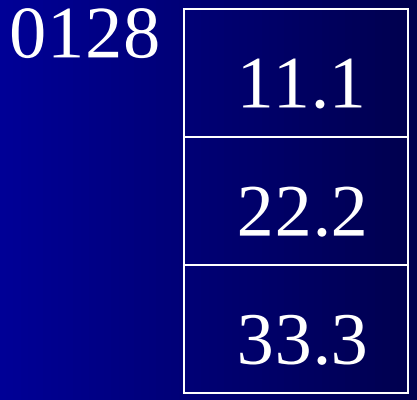



```
double[ ] myArray = createArray( );
```



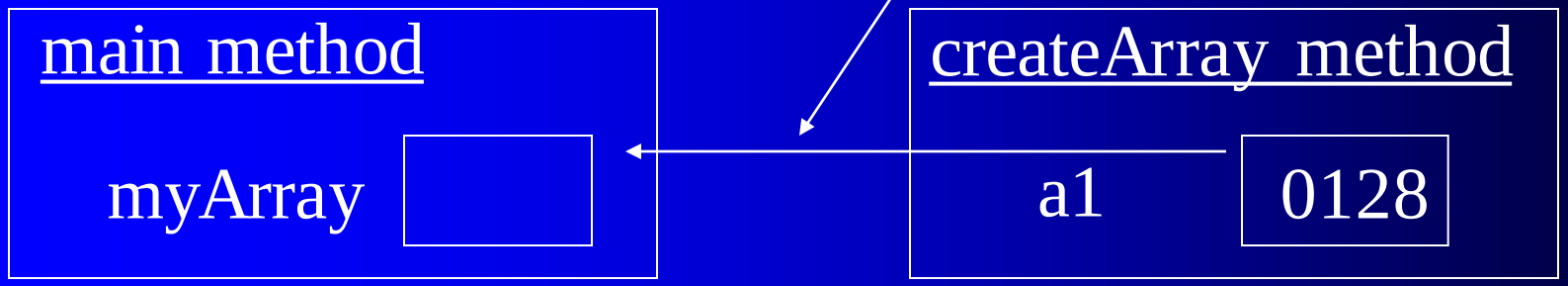
Methods that return an array

```
public double[ ] createArray()
{
    double[ ] a1 = new double[3];
    a1[0] = 11.1;
    a1[1] = 22.2;
    a1[2] = 33.3;
    return a1;
}
```



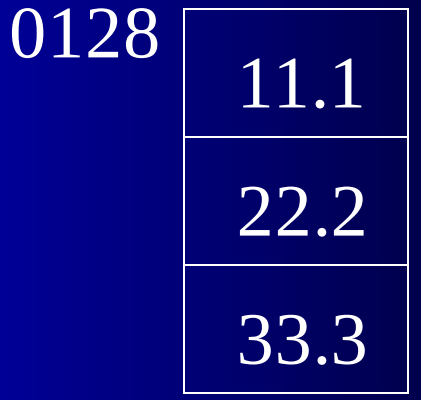

```
double[ ] myArray = createArray( );
```

the address of a1
is returned by createArray

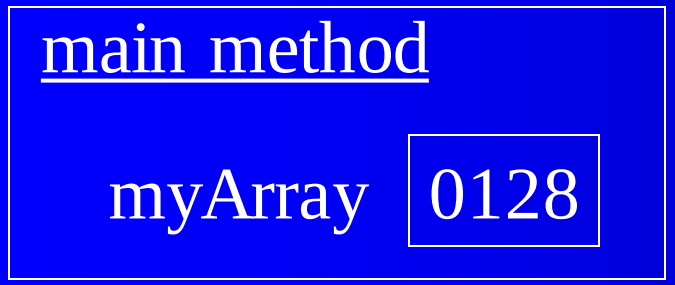


Methods that return an array

```
public double[ ] createArray( )
{
    double[ ] a1 = new double[3];
    a1[0] = 11.1;
    a1[1] = 22.2;
    a1[2] = 33.3;
    return a1;
}
```




```
double[ ] myArray = createArray( );
```



Example

- The array object in the bar chart program could be created and returned by a method that reads in the values

Example

```
private static int[ ] readArray(int n)
{
    int[ ] numArray = new int[n];
    System.out.print("Enter " + n + " integers: ");
    for (int i = 0; i < numArray.length; ++i)
    {
        numArray[i] = keyboard.nextInt( );

    }
    return numArray;
}

public static void main(String[ ] args)
{
    System.out.print("Enter the number of values in the bar chart: ");
    int arraySize = keyboard.nextInt( );
    int [ ] number = readArray(arraySize);
    displayHorizontalBarChart(number);
}
```

Arguments to main

- When a Java program starts, it is possible to pass in values to be used in the main method
- Every time we write a main method, we have to specify that it takes an array of String arguments

```
public static void main(String[ ] args)
```

- args will contain values that may be passed in at the command line

Arguments to main

```
public class CommandLineTester
{
    public static void main(String[ ] args)
    {
        System.out.println("args contains " + args.length + " strings");
        for (int i = 0; i < args.length; ++i)
        {
            System.out.println("Argument " + (i+1) + " is " + args[i]);
        }
    }
}
```

Arguments to main

```
>java CommandLineTester
args contains 0 strings

>java CommandLineTester Fred
args contains 1 strings
Argument 1 is Fred

>java CommandLineTester Fred Nerk
args contains 2 strings
Argument 1 is Fred
Argument 2 is Nerk

>java CommandLineTester Fred Nerk Age 23
args contains 4 strings
Argument 1 is Fred
Argument 2 is Nerk
Argument 3 is Age
Argument 4 is 23
```

= and == with arrays

- Equality and assignment with arrays are the *same as for other objects*
- = copies the reference to the array (memory address)
- == compares memory addresses
- Usually you do *not* want to use = or == with array object references

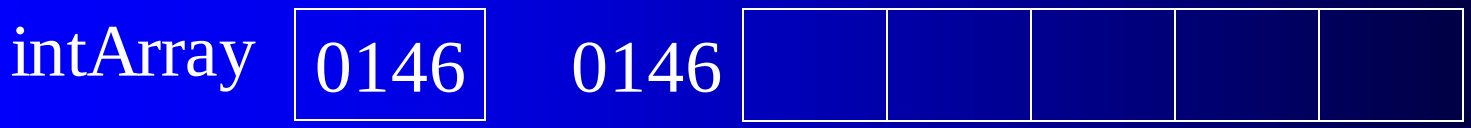
Arrays of objects

- Arrays can contain either primitive types or object references

```
int[ ] intArray = new int[5];

DigitalClock[ ] myTimes = new DigitalClock[10];
```

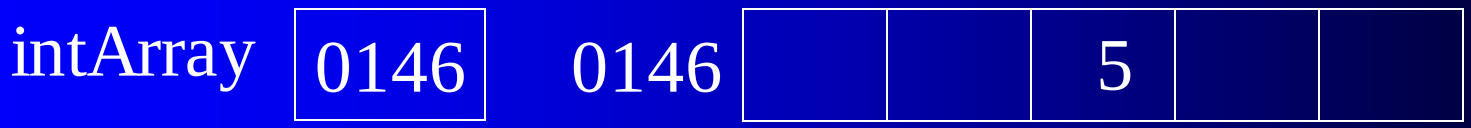
Arrays of objects

```
int[ ] intArray = new int[5];
intArray[2] = 5;
```



Arrays of objects

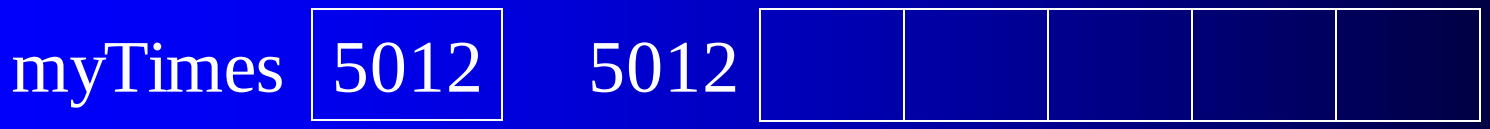
```
int[ ] intArray = new int[5];
intArray[2] = 5;
```



** The variables in the array store the actual values*

Arrays of objects

```
DigitalClock[ ] myTimes = new DigitalClock[4];
myTimes[2] = new DigitalClock(15,30);
```



Arrays of objects

```
DigitalClock[ ] myTimes = new DigitalClock[4];
myTimes[2] = new DigitalClock(15,30);
```

** The variables in the array store references to DigitalClock objects*

Arrays of objects

- When an array of object references is created, memory is allocated to store the references but not the objects themselves
- We need to create the objects separately

```
DigitalClock[ ] myTimes = new DigitalClock[10];
for (int j = 0; j < myTimes.length; ++j)
{
    myTimes[j] = new DigitalClock( );
}
```

Null references

- If we do not allocate memory for the objects before we attempt to use them, we will get a null pointer error

```
DigitalClock[ ] dc  = new DigitalClock[10];
dc[0].setHours(23);
```

- This is because, until we allocate memory, the reference does not refer to an object but to nothing (null)

Changing array elements in method arguments

- Array elements passed into a method behave in the same manner as other arguments of the same type as the element
- If the base type of the array is a primitive type, then the value of the array element is copied into the method and changes to it inside the method do not affect the actual parameter

```
public void changeNumber(int n)
{
   …
}

changeNumber(myArray[2]);
```

Changing array elements in method arguments

- If the type of the array element is a class type, then the value of the array element (which is a reference to an object) is copied into the method and changes to it inside the method affect the actual parameter

```
public void changeTime(DigitalClock dc)
{
   …
}

changeTime(myTime[3]);
```

Changing array arguments

- As an array is treated as an object, and an array variable is a reference to an array object, passing an entire array as the argument to a method copies the reference to the array object to the formal parameter of the method
- Changes to the array object in the method, therefore, affect the array argument passed to the method

Example

```
public static void reverse(int[ ] numbers)
{
    for (int i = 0; i < numbers.length / 2; ++i)
    {
        // swap i and length - 1 - i
        int temp = numbers[numbers.length - 1 - i];
        numbers[numbers.length - 1 - i] = numbers[i];
        numbers[i] = temp;
    }
}

public static void main(String[ ] args)
{
    int[ ] myArray = {1,3,5,7,9,11};
    reverse(myArray);
    for (int i = 0; i < myArray.length; ++i)
    {
        System.out.print(myArray[i] + " ");
    }
    System.out.println( );
}
```

Array attributes

- Objects can have array attributes
- The creation and use of array attributes must be carefully planned so that you do *not* try to use an array before allocating memory for it
- Careful consideration also needs to be given to returning the value of an array attribute so that you do not have privacy leaks

Example

- Rewrite the bar chart class so that it stores the array of values as an attribute
  - The constructor must create the array
  - There should be a method to read in the values
  - There should be methods to view and change each value in the array
  - There should be methods to display a horizontal or a vertical bar chart

Example

```
public class BarChart
{
    int[ ] values;

    public BarChart(int numberOfValues)
    {
        values = new int[numberOfValues];
    }

    public void readValues( )
    {
        System.out.print("Enter " + values.length + " integers: ");
        for (int i = 0; i < values.length; ++i)
        {
            values[i] = keyboard.nextInt( );

        }
    }
```

Example

```
public void setValue(int index, int newValue)
{
    values[index] = newValue;
}

public int getValue(int index)
{
    return values[index];
}
```

Example

```
private void displayLine(int num)
{
    for (int i = 0; i < num; ++i)
    {
        System.out.print("* ");
    }
    System.out.println( );
}

public void displayHorizontalBarChart( )
{
     for (int row = 0; row < values.length; ++row)
     {
          displayLine(values[row]);
     }
}
```

Example

```
private int maximum (int num1, int num2)
{
    if (num1 > num2)
    {
        return num1;
    }
    else
    {
        return num2;
    }
}
```

Example

```
private void displayRowOfVerticalChart(int row)
{
    for (int i = 0; i < values.length; ++i)
    {
        if (values[i] >= row)
        {
            System.out.print("* ");
        }
        else
        {
            System.out.print("  ");
        }
    }
    System.out.println( );
}
```

Example

```
public void displayVerticalBarChart( )
{
    // find maximum
    int max = values[0];
    for (int i = 1; i < values.length; ++i)
    {
        max = maximum(max,  values[i]);
    }
    System.out.println("maximum is " + max);
    for (int row = max; row > 0; --row)
    {
        displayRowOfVerticalChart(row);
    }
}
}
```

Example

```
public class BarChartTester
{
    public static void main(String[ ] args)
    {
        BarChart chart = new BarChart(10);
        chart.readValues( );

        chart.displayVerticalBarChart( );

        System.out.println( );
        System.out.println( );

        chart.displayHorizontalBarChart( );
    }
}
```

Returning an array attribute from a method

- What is the problem with this method in the BarChart class?

```
public int[ ] getValues( )
{
    return values;
}
```

Returning an array attribute from a method

- We could change the BarChartTester program to alter the returned array

```
BarChart chart = new BarChart(10);
chart.readValues( );

chart.displayVerticalBarChart( );

System.out.println( );
System.out.println( );

int[ ] myValues = chart.getValues( );
myValues[3] = 15;
chart.displayHorizontalBarChart( );
```

Returning an array attribute from a method

- getValues should make a copy of the array and return the copy (and not the attribute itself) to avoid a privacy leak

```
public int[ ] getValues( )
{
    int[ ] returnArray = new int[values.length];
    for (int i = 0; i < values.length; ++i)
    {
        returnArray[i] = values[i];
    }
    return returnArray;
}
```

Next lecture

- Partially filled arrays