Types in Java

- 8 **Primitive** Types
  - byte, short, int, long
  - float, double
  - boolean
  - Char
- **Object** types:
  - predefined e.g. “String” ;
  - User-defined – via “class” and constructors

- What if we want to store lots of items – e.g. midsem marks?

Primitive Type Syntax

```
double number = 3;
```

Array Syntax

```
double[ ] number;
```

Array Syntax

```
double[ ] number = {1,2,3} ;
double[] numArray = new double[3];
String sArray[ ] = {"This", "is", "an", "array"};
```

Arrays

- Arrays are predefined types of objects that can hold a series of values
  - An array uses an index to access a particular value
  - No explicit constructor needed
  - Some pre-defined variables e.g. “.length”

Array Implementation

Every time an array is initialized, using new, or by {… },

- a contiguous block of memory is assigned for it
- the array name (e.g numArray) is assigned the start address (In Java, this address is the *reference* for this variable).
- numArray[0] is the first element

Printing an Array

```
class ArrayPrint
{
   public static void main (String arg[])
   {
      double numbers[] = {1,2,3};
      double numbers2[] = {1,2,3,4};
      String s[] = {"This", "is", "an", "array"};
      printArray (s,4);
   }

   public static void printArray (String[] arr, int length)
   {     for(int j=0;j<length;j++)
          System.out.println("Element "+j+"= "+arr[j]);
   }
}
```

Array Length

Array objects have some predefined variables e.g. length

```
String s[ ] = {"This", "is", "an", "array"};
System.out.println("Length of s: "+
  s.length);
```

Rainfall data - average

```
class Rainfall {
    public static void main (String arg[])
    {
        double dailyRainFall[] = {12.3, 13.5, 4.2, 2.4, 1.1, 0, 10.8};
        int i;

        double average = 0;
        for (i=0; i<7; i++)
        {    average += dailyRainFall[i]; }
        average /= 7;

        System.out.println ("Average rain fall:"+ average + " mm");
//      System.out.printf ("Average rain fall: %2.2f mm %n", average);
    }
}
```

Finding the average

```
for (i=0; i<7; i++)
   {  average += dailyRainFall[i]; }
average /= 7;
```

Everytime I have another day of rain, I need to change the number “7” in my program

Instead, I can use

```
for (i=0; i<dailyRainFall.length; i++)
   {  average += dailyRainFall[i]; }
average /= dailyRainFall.length;
```

class ArrayPrint

```
class ArrayPrint
{
   public static void main (String arg[])
   {
      String s[ ] = {"This", "is", "a", "test", "array"};
      printArray(s);
      System.out.println( "s[0] length is: "+s[0].length() );
   }

   public static void printArray (String[] arr)
   {     for(int j=0;j<arr.length;j++)
          System.out.println("Element "+j+"= "+arr[j]);
   }

    // exercise: define lengthOfStringArray (String[] s)
}
```

Parallel Arrays

- Say wish to record who got how many marks in the recent midsem.
- One solution: use several arrays in parallel:

```
int[ ] rollNum= {6016, 6024, 6078};
double[ ] midsemMarks
        = {61.7,  54 , 74.2 };
String[ ] name = {"AbhishekA", "AbhishekS",
 "Ankit"};
```

- Finding maximum marks?

Maximum Marks

```
public static double findMax (double numbers[])
{
    double max = numbers[0];
    for (int i=1; i<numbers.length; i++)
        {   if (numbers[i] > max)
                max = numbers[i];
        }
    return max;
 }
```

But what if we want to know WHO got this maximum mark?

Can return the index of the maximum mark person instead.

Maximum Marks index

```
/** finds the index of the person with maximum marks */
public static int findMaxIndex (double numbers[])
{
   double max = numbers[0];
   int index = 0;
   for (int i=1; i<numbers.length; i++)
    {
       if (numbers[i] > max)
       {
         max = numbers[i];
         index = i;
        }
   }
   return index;
}
```

See file: ArrayMidsems.java

Parallel Arrays

Now we can use:

```
System.out.println ("Highest marks, by
  "+name[maxIndex]+ " (Y"+rollNum[maxIndex]+") =
  "+midsemMarks[maxIndex]);
```

(See file: ArrayMidsems.java)

BUT: What if we wish to sort the marks? How can we maintain the roll numbers etc?

Student records

```
rollNum=          {6016, 6024, 6078};
midsemMarks =   {61.7,  54 , 74.2 };
name =       {"Abhishek Murthy",
               "AbhishekS",
               "Ankit"};
```

In practice, parallel arrays are hard to maintain.

Better is to define an object STUDENT with data: rollNum, marks, and name, and define arrays of the STUDENT object: Student[ ]

Array Variables

- What happens if we do this?

```
int[] rollNum= {6016, 6024, 6078};
int[] a = {1,2,3,4};
a = rollNum;
```

- Now the part of memory previously accessed through “a” is lost. “a” and “rollNum” now point to the same place in memory.

Passing Arrays

- The method findAverage() is defined thus:

```
public static double findAverage (double[] arr)
  {   for (int i = 1; i<arr.length; i++)
      {  arr[0] += arr[i]; }
  return (arr[0] / arr.length) ;   // will this work?
  }
```

- Q. I invoke this method on the array numbers[ ]. findAverage(numbers). Does numbers[0] remain what it was or does it change?

- Since method arguments in Java are passed by reference – so numbers[0] has the same address as arr[0] inside the method; and modifications to arrays inside the method are reflected outside the method