```
public void run() {
 * Create tidisplay ( finterpreter getHelloPrompt() ); //declared here only to ensure visibility in finally claus
 * this method should be invoked from the * event-di//pass each line of input to finterpreter, and display Introduction to Java
            //fInterpreter's result
private statingutStreamReader inputStreamReader = new InputStreamReader ( System: Arrays and Strings
    //Make BufferedReader stdin new BufferedReader ( inputStreamReader ) // buffer )
            boolean hasRequestedQuit = false
            String line snull, lang Object
   //Creat List resulte as new ArrayList (); name> //display its data
   frame = try { | Invalid | Example "java lang String"> | Iterator quarksItr = recoveredQuarks.iterator(); frame = try { | Iframe("FocusConceptsDemo"); | Iang String"> while ( quarksItr.hasNext() ) { | frame.set! | String | quarksItr.hasNext() ) {
                line stdin.readline() availang
    //Create and/note that "result" is passed as an "out" parameter
    JComponent hasRequestedQuit = fInterpreter parseInput( line, result );
    newContentPdisplay(Sresult: ) | Sol Maring | les must be | light of lass Console |
    frame set Corresult clear () on tentrane
    //Display the window.
    frame p catch ( IOException ex ) {
    frame.set System, erraprintin(ex); lass name>java.util.Gregor
                                                                                                List [aText o) ments
              The inheritance tree
                                                                                       ispl
            finally {class java.util.GregorianCalendar
                                                                                                = aText.iterator()
                                                                                       extIt
                                                                                             hasNext() a)ur{uments
public static display(fBYE) ava util Calendar
                                                                                             ter. next() preter) the
    //Scheduleshutdown (stdin event dispatching threa
    //creating and showing this application's GU
    javax. swing. SwingUtilities.invokeLater(new_
        public void run() {
///_PRIVATE
                                                                                                           aStdin )
         private static final String fBYE =
         private Interpreter fInterpreter
          /**
 void pri*tDisplay some ttext/stostdoutsn
     final String[] mvStrings = new String[2]
```

Introduction

- Arrays and Strings are special Java classes
 - Special in the sense that the allow for a syntax that is more like regular programming languages
 - For example, we have been treating Strings like primitive variables
 - Java provides that syntax as "syntactic sugar" to make it easier to code in a natural stype



Arrays

- Arrays is Java look just like arrays in other programming languages
 - An array name with an indexed set of entries like a[0] etc.
 - We cannot reference an array entry that is out of range
 - Arrays know how long they are

```
public class Runner {

public static void main(String[] args) {
    // defining an array by providing initial values
    int [] arr = {11,11,12,13};
    System.out.println("arr[] is a" + arr + "with length " + arr.length);
    for (int index = 0; index < arr.length; index++) {
        System.out.println("Entry " + index + " is " + arr[index]);
    }
}

arr[] is a[I@251a69d7 with length 4
Entry 0 is 11
Entry 1 is 11
Entry 2 is 12
Entry 3 is 13</pre>
```



Arrays

What is actually going on is the following code

```
public static void main(String[] args) {
    // define an array object that will reference a block of memory containing
    // 4 ints
    int [] arr = null;
    // allocate the memory
    arr = new int[4];
    // initialize
    arr[0] = 10;
    arr[1] = 11;
    arr[2] = 12;
    arr[3] = 13;
    System.out.println("arr[] is a" + arr + " with length " + arr.length);
    for (int index = 0 ; index < arr.length; index++) {</pre>
        System.out.println("Entry " + index + " is " + arr[index]);
  arr[] is a[I@251a69d7 with length 4
  Entry 0 is 10
  Entry 1 is 11
  Entry 2 is 12
  Entry 3 is 13
```

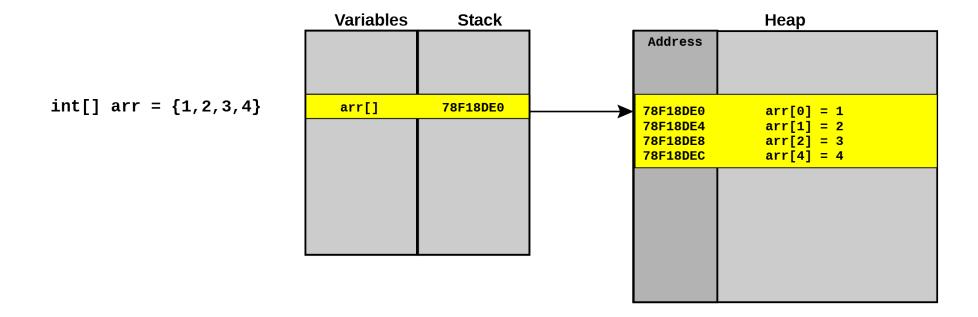


What are Arrays

- An array is a contiguous block of memory on the heap
 - The length of an array is fixed they can't grow or shrink
- All of the elements in an array must be the same type
 - This means they are all the same size
 - This allow for very fast look ups
 - We don't have to loop through an array to find an element
- If we want to find element at index 3 in an array of ints
 - If the array starts at memory location '4577' and int is 4 bytes wide
 - Then the memory location of the element we want is '4577 +(3 + sizeOf(int))'
 - This allows for constant access time to any element no matter how big the array is



Arrays in Memory





Copying Arrays

- An array is an object
- Assigning an array variable to another array variable does not copy the array
 - It copies the address of the array this is called a shallow copy
 - We have to copy the array element by element this is called a deep copy
- Two arrays are "equal" the point to the same memory location
- Two arrays are "equivalent" if corresponding elements are equal
- There are utility methods in the collections class that can make the task easier



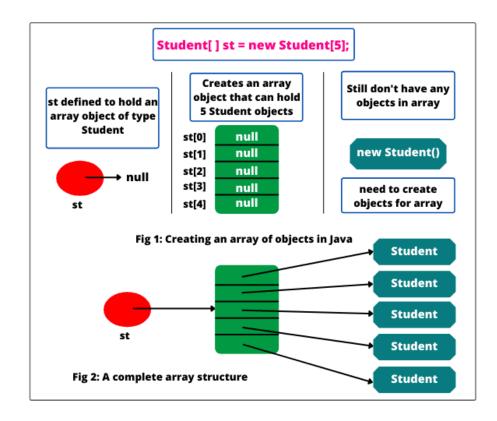
Array Operations

- In loops earlier, we had to provide an index or iterator to keep track of the iteration
 - Arrays have an internal iterator, specifically the index of the elements
 - We can use a simpler version of the for loop
- The java.util.Arrays library has a number of functions that can be used to copy, sort and perform other operations on arrays.



Arrays of Objects

- We can also have arrays of object
 - In this case, the elements of the arrays are the memory locations of the object
 - This meets the constant size requirement since all address are the same width



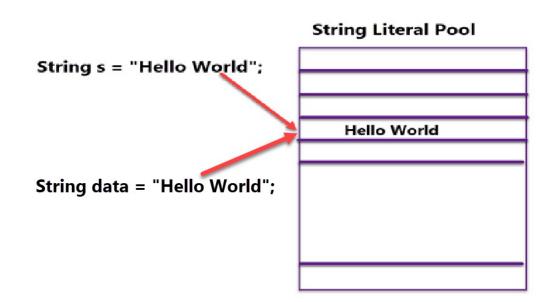






Strings

- Strings are objects just like other objects
- Java allows syntactic sugar so that we can write them as if they were primitive types
- Strings are immutable
 - Once created, the value of a string can not be changed
 - This is because there in only one copy of String literal, like "Hello World" that is shared by all String objects that have that literal as a value
 - These unique literals are said to be interned in a special constant area called the string pool





String Builder

- A far more efficient way to build a string is to convert it to an array of chars
 - A StringBuilder is a buffer that allows us work with a String in this way.







