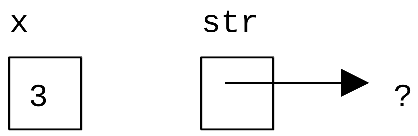
A String is a list of characters. A String variable only contains a number, which is an address for the location in memory of the characters. The String variable is then said to **refer** (- point to an address in memory) to the actual object containing the list of characters. Declaring a String variable does not allocate any memory for characters, it only allocates memory to store an address, which takes the same amount of memory as an int. All variables that have object data types contain addresses that refer to the location in memory where the object is stored. These variables are also called **references**.

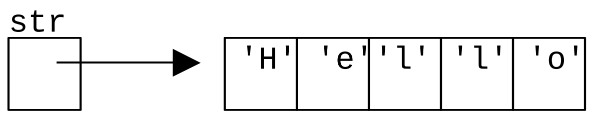
When drawing by hand the memory contents of a program, the memory address stored in a String variable is usually drawn as an arrow pointing to the list of characters associated with that String. The picture shown below is the result of the following code:

```
String str;
int x;
x = 3;
```



After declaring the String variable str, we don't know what memory address is stored in that variable, and no list of characters exists in memory yet, so we say we don't know what str refers to, and show it pointing to a question mark. One thing we can do to create a list of characters in memory is to put a String literal in the code. If we assign a String literal to a variable, it will make that String variable refer to that list of characters. For example, the picture shown below is the result of the following code:

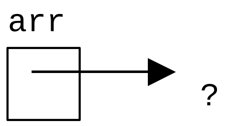
```
String str;
str = "Hello";
```



The characters are stored in a set of **contiguous** (- side-by-side) memory locations in an object called an **array** (- non-resizable list of data, all of the same data type, stored in contiguous memory locations). The arrays used by Strings contain characters. Each character is an **element** (- member) of the list, and each element has a numeric index. The first element in the list has an index of zero, and each element after that has an index that is one greater than the one before it. The **length** is the number of elements.

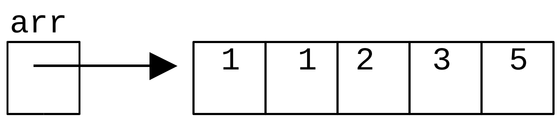
Arrays can hold data of any type, and can have any length, but neither the data type nor the length of an array can be changed once the array has been created. Also, every element in an array must have the same data type. Just as with Strings, you can have array variables. Array variables also contain a numeric address that refers to the memory location of the actual list of data. An array can be declared as follows:

```
int arr[];
```



This says that arr is an array variable that will refer to an array of ints. There is no array object for arr to refer to. To change that, you can create an array object and assign it to arr. There are two ways to create an array object. The simplest way is an **array initializer** (- an expression that creates an array and gives each element a value), which is similar to a literal. An example:

```
int arr[] = {1, 1, 2, 3, 5};
```

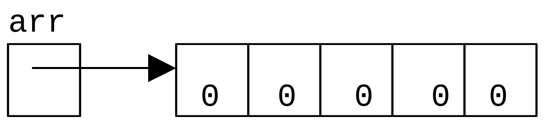


To use an array initializer anywhere other than an initialization statement, a little extra code is required:

```
int arr[];
arr = new int[] {1, 1, 2, 3, 5};
```

When using an array initializer, the length of the array that is created is equal to the number of elements in the initializer list. Arrays can also be created without specifying the values for the elements, but then the length needs to be specified. The values of all of the elements in the array that is created will be the default value for the data type stored by the array. An array can be created this way using an array constructor:

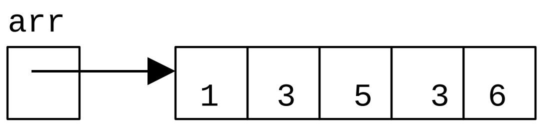
```
int arr[];
arr = new int[5];
```



The array constructor consists of the word new, the data type, and the length in brackets. The default value for ints is zero.

In a String object, the values of the characters cannot be changed. In an array object, however, elements can be changed. Array access expressions, which can be used to get the value of an element of an array, can also be used on the left side of the equals sign in assignment statements to change the value of an element. An array access expression consists of the name of a reference to the array, followed by the index in brackets. The following example continues from the previous one:

```
arr[1] = 3;
arr[arr[4]] = 1;
arr[4] = arr[1] * 2;
arr[arr[1] – arr[0]] = 5;
arr[3] = arr[3-2];
```



Finally, the length of an existing array can be determined by accessing its length attribute. The length attribute is an instance constant. The following line of code will print the length of the array arr from the previous example (The length of arr is 5):

```
System.out.println("The length of arr is " + arr.length);
```