CSCI 1103: Strings

Chris Kauffman

Last Updated:
Fri Oct 13 10:53:02 CDT 2017

Logistics

Reading from Eck

- Ch 2.3.2-3 Classes, Objects, Strings
- Next week: Ch 4

Project 3

- Will go up over the weekend
- Discuss in Monday Lecture
- Due the following week

Goals

- Finish arrays
- Strings

Exam 1

- Graded, Hand back at Monday labs
- Discuss answer in lab

Lab06: Strings, Methods

- Strings today
- Methods next week
- Monday lab covers both

Strings, char, and Arrays

- Java's `String` type is essentially an array of characters, `char` data type
- `char` represents a single character as in

```
char letter = 'C';
char punct = '?';
char newline = '\n';
```

- Note the single quotes for `char` and the backslash for the special newline character
- Most techniques that apply for arrays apply equally well to `Strings`: operate on them as arrays of characters
- Minor adjustments are required to access features

Feature	Arrays	Strings
Declaration	`int arr[];`	`String str;`
Assign / initialize	`arr = new int[3]{2,4,6};`	`str = "hello";`
Length	`int len = arr.length;`	`int len = str.length();`
Access elements	`int x = arr[1];`	`char c = str.charAt(1);`
Alter elements	`arr[2] = 8;`	*Not possible*

Note last: `Strings` are immutable, once created cannot be changed

Exercise: Find New / Interesting Things in these Samples

```
 1  // Count characters in a word using an array
 2  public class CountCharArray{
 3    public static void main(String args[]){
 4      System.out.println("Enter word length:");
 5      int len = TextIO.getInt();
 6      char word[] = new char[len];
 7
 8      System.out.println("Enter a whole word:");
 9      for(int i=0; i<word.length; i++){
10        word[i] = TextIO.getChar();
11      }
12
13      System.out.println("Enter a character to count:");
14      char letter = TextIO.getChar();
15
16      int count = 0;
17      for(int i=0; i<word.length; i++){
18        if(letter == word[i]){
19          count++;
20        }
21      }
22
23      System.out.printf("'%c' appears %d times in '%s'\n",
24                        letter, count,word);
25    }
26  }
```

```
> javac CountCharArray.java
> java CountCharArray
Enter word length:
15
Enter a whole word:
hellooooo-world
Enter a character to count:
o
'o' appears 6 times in '[C@2a139a55'
```

```
 1  // Count characters in a word using Strings only
 2  public class CountCharString{
 3    public static void main(String args[]){
 4      // No need to get length ahead of time if using strings
 5
 6
 7
 8      System.out.println("Enter a whole word:");
 9      String word = TextIO.getWord();
10
11
12
13      System.out.println("Enter a character to count:");
14      char letter = TextIO.getChar();
15
16      int count = 0;
17      for(int i=0; i<word.length(); i++){
18        if(letter == word.charAt(i)){
19          count++;
20        }
21      }
22
23      System.out.printf("'%c' appears %d times in '%s'\n",
24                        letter, count,word);
25    }
26  }
```

```
> javac CountCharString.java
> java CountCharString
Enter a whole word:
hellooooo-world
Enter a character to count:
o
'o' appears 6 times in 'hellooooo-world'
```

Notes on CountCharArray CountCharString

- Can make a standard array of characters

  ```
  char myChars[] = new char[10];
  ```

  Has less functionality than a `String`
- `TextIO.getChar()` reads a single `char`
- `TextIO.getWord()` reads a whole word / `String`
- Both of these stop at and skip white space
- Printing a `String` puts expected stuff on the screen
- Printing an array puts weird stuff on the screen

  ```
  char carr[] = new char[5];
  System.out.println(arr);
  // [C@2a139a55

  int iarr[] = {1, 2, 3};
  System.out.println(iarr);
  // [I@23d2a7e8
  ```
- Guesses on what these weird things are? *Hint: arrays are a reference type. What's in a reference types memory cell?*

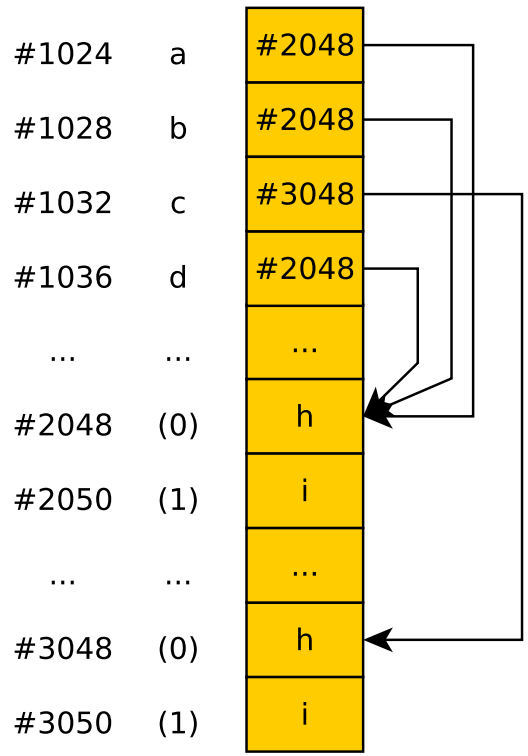
Exercise: String Equality

- Recall array equality
  - How to tell if two arrays are "shallowly" equal?
  - How to tell if two arrays are "deeply" equal?
- Transfer your knowledge: predict the following results
- Support your predictions with a memory diagram

```
 1  public class StringsEqual{
 2    public static void main(String args[]){
 3      String a = "hi";
 4      String b = a;
 5      String c = new String("hi");
 6      String d = "hi";
 7
 8      System.out.printf("a == b : %b\n", a==b);
 9      System.out.printf("a == c : %b\n", a==c);
10      System.out.printf("a == d : %b\n", a==d);
11
12      System.out.printf("a.equals( b ) : %b\n", a.equals(b));
13      System.out.printf("a.equals( c ) : %b\n", a.equals(c));
14      System.out.printf("a.equals( d ) : %b\n", a.equals(d));
15    }
16  }
```

Answers: String Equality

- Shallow equality always checks whether references point to the same place; use `a == b`
- Deep equality checks whether what is pointed at contains the same stuff; for arrays and strings requires a loop over all elements
- Shallow checks with == below are
  - `true` : both point to same place
  - `false`: point to different locations
  - `true` : compiler identifies identical strings, uses same area
- Deep equality is checked with the `equals()` method and gives all `true` in `StringsEqual.java`



Concatenation: String "Addition"

- Concatenation combines two strings to produce a new string
- Concatenation is accomplished in with either `concat()` or `+`

```
String a = "hi";
String b = "gh";
String c = a+b;                 // high
String d = a.concat(b);         // high
```

- Most variables automatically "stringify" if used in concatenation

```
int count = 5;
String req = "I want "+count+" apples";
```

- Note that concatenation always creates new `Strings` in memory. A loop like

```
String ans = "";
for(int i=0; i<10; i++){
  ans = ans + i + " ";
}
```

  creates many intermediate strings

- `String` is a special class in that it has compiler support for some *immediate* operations such as using `+` instead of `concat()`
- Most other classes require long names to invoke methods

String Method Examples

```
String name       = "Chris";
//                   01234
String occupation = "csci prof";
//                   012345678
// Example Methods
int nameLength = name.length();      // ask for the length of name
int occLength = occupation.length(); // length of occupation
char third = name.charAt(3);         // third character of "Chris"
char fifth = occupation.charAt(5);   // third character of "csci prof"
String subString = name.substring(1,4); // "hri" chars 1 to 3
String changed = occupation.replace("prof","badass"); // smirk
```

- Strings have many methods
- Complete list is in the Java documentation:

https://docs.oracle.com/javase/8/docs/api/java/lang/String.html