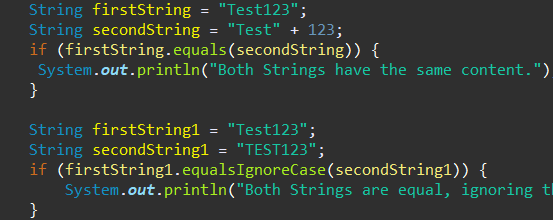
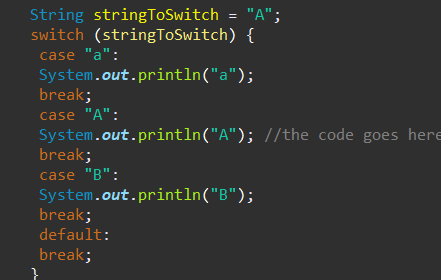
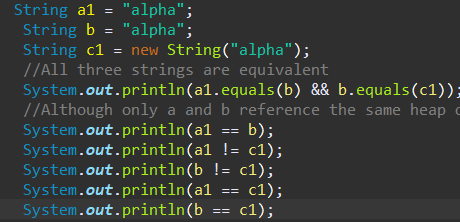
**Strings**Strings (java.lang.String) are pieces of text stored in your program. Strings are not a primitive data type in Java; however, they are very common in Java programs.  
In Java, Strings are immutable, meaning that they cannot be changed.

**Comparing Strings**  
In order to compare Strings for equality, you should use the String object's equals or equalsIgnoreCase methods.  


**Do not use the == operator to compare Strings**  
operators to compare Strings. These operators actually test references, and since multiple String objects can represent the same String, this is liable to give the wrong answer.   
Instead, use the String.equals(Object) method, which will compare the String objects based on their values. For a detailed explanation, please refer to Pitfall: using == to compare strings

**Comparing Strings in a switch statement**  
As of Java 1.7, it is possible to compare a String variable to literals in a switch statement. Make sure that the String is not null, otherwise it will always throw a NullPointerException. Values are compared using String.equals, i.e. case sensitive. 

**Comparing Strings with constant values**  
When comparing a String to a constant value, you can put the constant value on the left side of equals to ensure that you won't get a NullPointerException if the other String is null.  
"baz".equals(foo);  
While foo.equals("baz") will throw a NullPointerException if foo is null, "baz".equals(foo) will evaluate to false.

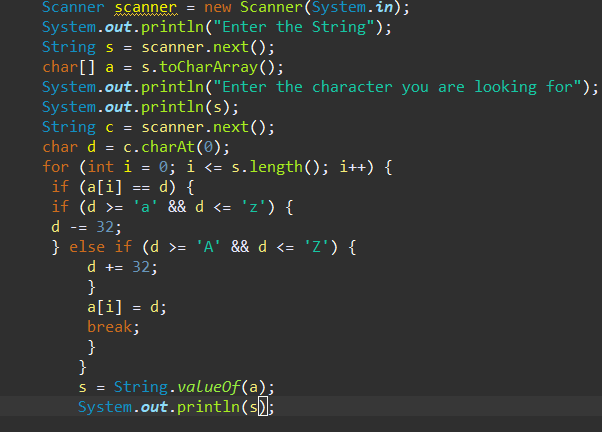
**String pool and heap storage**  
Like many Java objects, all String instances are created on the heap, even literals. When the JVM finds a String literal that has no equivalent reference in the heap, the JVM creates a corresponding String instance on the heap and it also stores a reference to the newly created String instance in the String pool. Any other references to the same String literal are replaced with the previously created String instance in the heap

Interview Questions

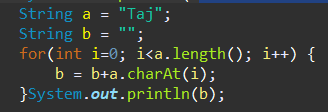
**Level 1: Beginner**

**1. Changing case of a specific character within an ASCII string?**

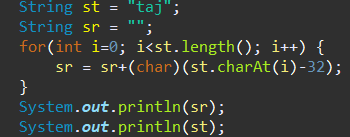
* Steps: 1. Declare a string.   
  2. Input the string.   
  3. Convert the string into a character array.   
  4. Input the character that is to be searched.   
  5. Search for the character into the character array.   
  6. If found,check if the character is lowercase or uppercase. If Uppercase, add 32 to the ASCII code of the character. If Lowercase, subtract 32 from the ASCII code of the character.   
  7. Change the original character from the Character array.   
  8. Convert the character array back into the string.   
  Voila, the Case of the character is changed.   
  An example of the code for the algorithm is



**2. TO COPY ONE STRING TO ANOTHER**

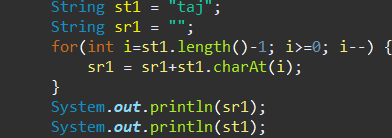
* .

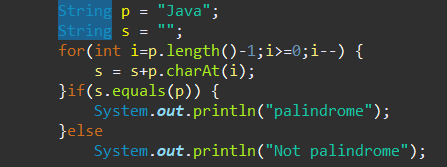
**3. LOWE TO UPPER WITHOUT INBUILT"?**



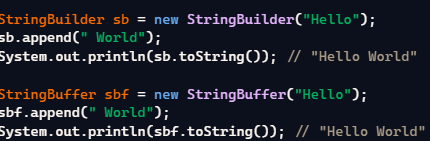
**Level 2: Intermediate**

**1. Reverse String?**

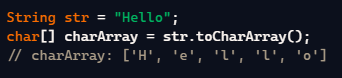


**2. PALINDROME  
**

**3. Explain the difference between StringBuilder and StringBuffer.?**

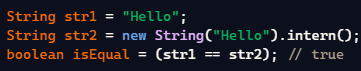
Both StringBuilder and StringBuffer are used to create mutable strings. The key difference is that StringBuffer is synchronized (threadsafe) and StringBuilder is not, which makes StringBuilder faster and preferable in a single-threaded environment.|.  


**4. How do you convert a String to an array of characters.?**



**Level 3: Advanced**

**1. Explain the concept of String internment in Java.?**

String internment refers to a method of storing only one copy of each distinct String value, which must be immutable. This helps save memory. The intern() method returns a canonical representation for the string object.  
  
  
**2. How do you split a String into an array using a delimiter?**

