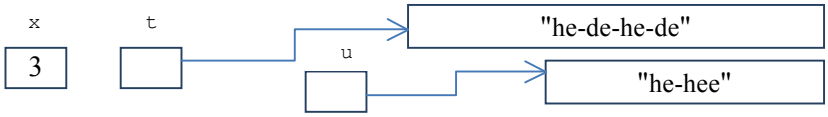
Practice Problems: String Problems

1. Tracing Code with Strings

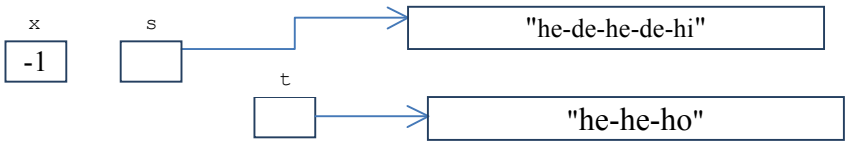
Show what is stored in memory at the end of each of these programs.

```
public class String-CharAt {
  public static void main(String [] args) {
       String s = "hello";
       char c = s.charAt(1);
       c++;
       s = "he" + c + c + "o";
  }
}
```


```
public class String-Substring {
  public static void main(String [] args) {
       String t = "he-de-he-de";
       int x = t.indexOf("de");
       String u = t.substring(0,x);
       u = u + "hee";
  }
}
```

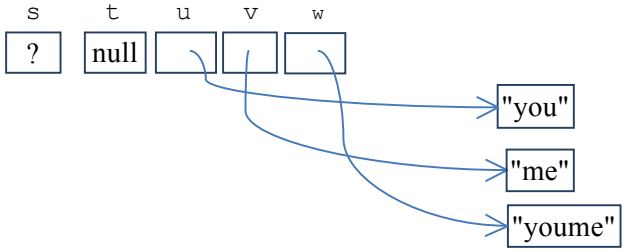



```
public class String-IndexI {
  public static void main(String [] args) {
       String s = "he-de-he-de-hi";
       String t = "";
       int x = s.indexOf("de", 0);
       while(x>=0) {
             t = t + "he-";
             x = s.indexOf("de", x+1);
       }
       t = t + "ho";
  }
}
```

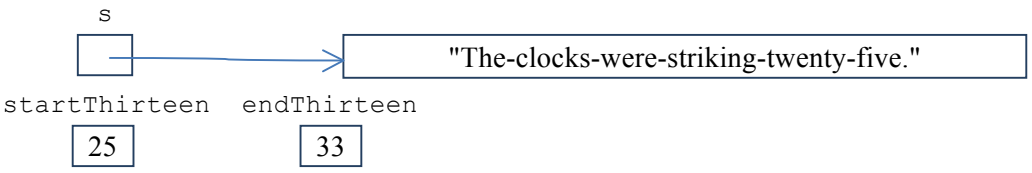


```
public class String-Equals {
  public static void main(String [] args) {
      String s = “hibbity-hibbity”;
      int i=0;
      int count = 0;
      while(i<s.length()-2) {
            if(s.substring(i,i+2).equals(“ty”)) {
                  count++;
            }
            i++;
      }
  }
}
```

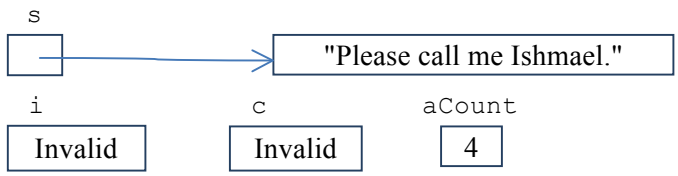

```
public class String-Assignments {
  public static void main(String [] args) {
    String s;
    String t = null;
    String u = "you";
    String v = new String("me");
    String w = u + v;
  }
}
```



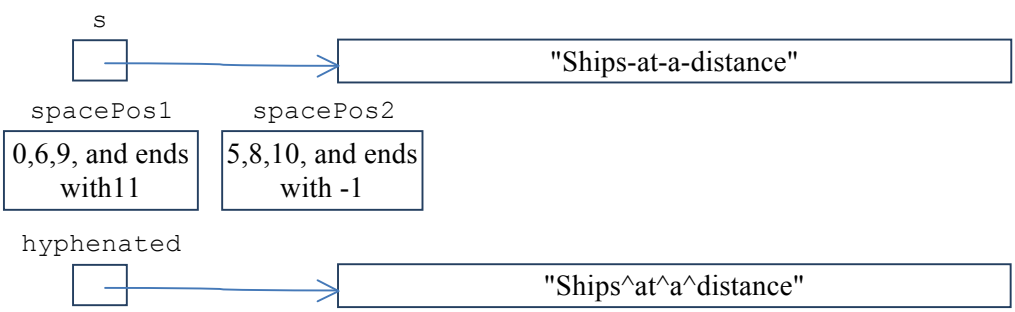

```
public class String-Insert-Delete {
  public static void main(String [] args) {
    String s = "The-clocks-were-striking-thirteen.";
    int startThirteen = s.indexOf("thirteen");
    int endThirteen = startThirteen + "thirteen".length();
    s = s.substring(0, startThirteen)
        + "twenty-five"
        + s.substring(endThirteen);
  }
}
```



```
// Here is a typical example of a loop used to
// process a String.
// In this example, the loop visits each character
// in the String once.
public class String-Processing {
  public static void main(String [] args) {
    String s = "Please call me Ishmael.";
    int aCount = 0;
    for(int i=0; i<s.length(); i++) {
      char c = s.charAt(i);
      if(c == 'e') {
        aCount++;
      }
    }
  }
}
```




```
// Here is an example that repeatedly loops through the String,
// processing one word at a time.
public class String-Processing {
public class String-Processing {
  public static void main(String [] args) {
    String s = "Ships-at-a-distance";
    int spacePos1 = 0;
    int spacePos2 = s.indexOf("-");  // target
    String hyphenated = "";
    while(spacePos2>=0) {
      String word = s.substring(spacePos1,spacePos2);
      hyphenated = hyphenated + word + "^";  // replacement
      spacePos1 = spacePos2 + 1;
      spacePos2 = s.indexOf("-", spacePos1);  // multiple appearance
    }
    if(spacePos1<s.length()) {
      hyphenated = hyphenated + s.substring(spacePos1);
    }
  }
}
```



2. Repeat-X and Sum Algorithms with Strings

Write an event-controlled loop to solve the following problem, based on the fencepost problem discussed in loop chapter.

1. Write a program that asks the user to repeatedly enter a String. The program should concatenate those Strings together, but insert spaces and the word "not" between every pair of words the user enters. Stop when the user enters the String "stop". Display the final String. For instance, the program output might look like:

```
Please enter some Strings:
Such
eyes
you
have
stop

Such not eyes not you not have
```

```
Scanner keyboard = new Scanner(System.in);
String inputStr, sumStr="";

inputStr = keyboard.nextLine();
if (!inputStr.equals("stop")){
      sumStr+= inputStr;
      inputStr = keyboard.nextLine();
      while (!inputStr.equals("stop")) {
          sumStr+=" not ";
          sumStr+=inputStr;
          inputStr = keyboard.nextLine();
      }
}
System.out.println(sumStr);
```

3. File I/O and exception throw (see the attached files)

a. (FileLetter.java) Write a program that asks the user to enter the name of a file, and then asks the user to enter a character. The program should check the validation of such a file (i.e., existing or not), and avoid using any unaccepted file name. Then, the program will count and display (on screen) the number of times that the specified character appears in the file.

b. (FileHead.java) Write a program that asks the user for the name of a file. The program should check the validation of such a file (i.e., existing or not), and avoid using any unaccepted file name. The program should display only the first 5 lines of the file's content. If the file contains fewer than 5 lines, it should display the file's entire contents.