Static Methods in Driver

Recall Lab #1

- Read user phrase, character
- Replace character with x.
- Now: Also count the number of x's in the new phrase.

New code:

```
int count = 0;
for (int i = 0; i < newStr.length; i++)
    if (newStr.charAt(i) == 'x')
        count++;
```

Now main() method contains all the code. Better to place some code in other methods.

method - function: set of code that performs an operation

method header - defines properties of the method

```
public static return-type method-name (type1 var1,
    method is what method returns to
    in same file as main() calling method
    can be void)

parameter list

what the method takes as input
(can be empty)
```

Method replaceAndCount:
In-class code

public class ModifyString {

    public static void main(String[] args) {
        //will fill in later

        public static int replaceAndCount(String s, String c) {

            String newstr = s.replace(c, "x"); //notice String "x"
            int count = 0;
            for (int i = 0; i < newstr.length(); i++) {
                if (newstr.charAt(i) == 'x') //char "x"
                    count++;
            }

            return count; //return must be last executed
            //statement
        }

        //convention: top-most method
        public static void main(String[] args) {

            Scanner scan =
            System.out.println("Enter string: ");
            String str = scan.nextLine();
            System.out.println("Enter char to replace:");
            String chr = scan.nextLine();
            int numXs = replaceAndCount(str, chr);
            return;
        }
    }
}
Sys::println("num Xs:" + numXs);

// or replace last 2 lines with:
Sys::println("num Xs:" + replaceAndCount(str, chr));

??

Back to scope:

scan have s have scope
str { Scope local to c
chr { to main() newstr { replaceAndCount() count
numXs

? - scope local to for loop.

If a variable is going to be used by many methods, make that variable accessible to all methods

public class ModifyString {

    public static String str; // global scope
    // accessible by all methods

    <Sample Code>
}
Recap: Data types in Java:
- primitive data: int, double, char, boolean
- built-in objects: String, Scanner, arrays, Random
- programmer-defined objects

Methods/Operations:
- primitive: +, -, *, =, <
- built-in: equals(), nextInt(), [], nextInt()
- programmer-defined: replaceAndCount()

Object: entity with attributes (variables) and operations (methods).
Class: defines variables + methods of an object.

4 terms also apply to built-in objects.

Example:
```java
String str = "hello";
// ↑  ↑  ↑
// class  object/  value
//     variable

attribute: length
method: replace(), toUpperCase(), equals()

Classic example: programmer-defined object: Bank Account

First step: Think about which attributes + operations would be useful.
attributes

- balance (double)
- account number (String)
- name (String)

operations/methods (return type, parameters)

- withdraw()
- deposit()
- checkBalance()

Simplest approach first:

```java
public class BankAccount {  // save as BankAccount.java

    String accNum, name;  // global scope
    double balance;

    // constructor - for creating a BankAccount object
    public BankAccount(String num)
    {
        accNum = num;  // typically an attribute
typically an attribute
    that is unique to
each object:

    this.accNum = num;  // we will use this
}

```

Diagram:

```
<table>
<thead>
<tr>
<th>Main()</th>
<th>BankAccount()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Driver()</td>
</tr>
</tbody>
</table>

```