CO1005: Strings

Dr Tom Ridge

2018 Q1

This version compiled 2018/01/16 at 16:07:41

Why strings?

What are strings?

- See Java API resources listed on the module webpage

Constructing strings, examining strings

Obvious question- which of the library String functions are truly essential?
Essential String functions - every function can be constructed from these

Building up	Breaking down
`""`	`"".equals(...)`
`c+s`	`s.charAt(0), s.substring(1)`

(notice how the two entries in each row correspond to each other)
To make things easier, we also allow joining a character onto the end of a string: `s+c`

Mantra

The mantra for this week is:

> *A string is either the empty string or it is not, in which case there must be a first character, and the rest of the string.*

Please repeat this 100 times before you go to bed at night. Also when you wake up in the morning.

Some subtleties

- What is the difference between `s.equals("")` and `"".equals(s)` ?
- What is the domain of these functions (i.e. what are the allowable arguments to these functions?)? What happens if you call a function outside the domain e.g. `"".substring(1)`? `"".charAt(0)`?
- What rules should you obey when working with strings?

The substring rule; the charAt rule

The substring rule (not standard terminology) is the following:

> *Do not use `s.substring(1)` unless you know that `s` is not the empty string!*

This is a very important rule!!!
A similar rule (the charAt rule) is:

> *Do not use `s.charAt(0)` unless you know that `s` is not the empty string.*

The reason for these rules is, as in the last slide, things go wrong if you use these functions on the empty string.

You need to put code fragments in a class

In the following I give code fragments. However, Java requires that all code is put in a class. A simple class is:

```
class Simple {
  // so we can use print rather than System.out.println
  void print(Object s) { System.out.println(s.toString());

  // a method with no arguments, and which doesn't return
  void f() {
    // you can put some code here...
  }

  public static void main(String[] args) {
    // when you run the program, you need a main method
    // here we just create a new object, and call the meth
    Simple s = new Simple();  s.f();
  }
}
```

Some code

- The following code is Java syntax, but without class declarations
- You typically have to put this code in methods inside a class (see previous slide).
- Sometimes I will write `print` when I mean `System.out.println`.

VERY IMPORTANT NOTE ABOUT CODE EXCERPTS

Repeating the last few slides:

- The following code excerpts will not compile on their own!!!
- You have to put them inside a class
- On the webpage, there are `.java` files you can download, compile, and run. These files contain all the excerpts from these slides.
- The excerpts in the slides are manually copied over from the `.java` files, potentially introducing bugs. If in doubt, use the `.java` files rather than copying from the slides.

Some code

To understand this code, put some print statements in, and run the code with various arguments... yes, it does take a long time!

```
String sfoo = "Tom";
sfoo;

sfoo.charAt(1);
sfoo.charAt(2);
sfoo.charAt(3);

sfoo.substring(1);
sfoo.substring(2);

sfoo.equals("Cat");
sfoo.equals("Tom");

"".equals(sfoo);
```

The essential algorithm

The essential algorithm: walk through a string character by character (that is all!)

While loops

```
while(true) {
  print("I love programming!");
}
```

What does this do?

While loops

```
int i=0;
while(i<10) {
  print(i);
  i=i+1;
}
```

What does this do?

Template code

```
1  // some template code
2  int somefun(String s) {
3    int to_return = 0;
4    while(true) {
5      if(s.equals("")) return to_return;
6      to_return++;
7      s=s.substring(1);
8    }
9  }
10
11 somefun("hello");
```

What does this do?
How should you understand this code?
What rules does this code follow?

occurs

```
1  boolean occurs(char c, String s) {
2    boolean to_return = false;
3    while(true) {
4      if(s.equals("")) return to_return;
5      // s is not the empty string
6      char c2 = s.charAt(0);
7      if(c==c2) { to_return = true; }
8      s=s.substring(1);
9    }
10   return to_return; // never reached
11 }
12
13 print(occurs('a',"hello"));
```

removeAll

```
1  String removeAll(char c, String s) {
2    String to_return = "";
3    while(true) {
4      if(s.equals("")) return to_return;
5      // at this point s is not ""
6      char c2 = s.charAt(0);
7      if(c2!=c) { to_return += c2; }
8      s=s.substring(1);
9    }
10   return to_return; // won't be reached
11 }
12
13 print(removeAll('o',"hello"));
```

remove1, reverse, reverse_words

Try implementing the following 3 methods:

```
String remove1(char c, String s) {
  ...
}

remove1('a',"bac");
remove1('a',"baac");
remove1('a',"bc");

String reverse(String s) {
  ...
}

reverse("hello");

String reverseWords(String s) {
  ...
}
```

remove1

```
String remove1(char c, String s) {
  String to_return="";
  while(true) {
    if(s.equals("")) return to_return;
    // s is not the empty string
    char c2= s.charAt(0);
    if(c==c2) return(to_return+s.substring(1));
    to_return=to_return+c2;
    s=s.substring(1);
  }
}

remove1('e',"");
remove1('e',"ello");
remove1('e',"elloee");
remove1('e',"hello");
remove1('e',"helleo");
```

Recap

- Strings, and basic string functions
  - What is the mantra?
- While loops - for doing something over and over again
- Simple programs - occurs, remove1
- How to understand programs
  - walk through them line by line, keeping track of the values of the relevant variables

reverse and reverseWords

- have a look at the other two exercises
- reverseWords can be done using reverse

Introduction to recursion

What is recursion? a function calling itself
Recursion is like a while loop- you go back to the beginning of the function
But - and this is crucial to understand - the arguments (almost always) change

Example of recursion

First, we give an example **without** recursion, so we can compare it to the version with recursion:

```
1
2  String rev2(String s, String to_return) {
3    while(true) { // X1
4      if(s.equals(""))
5        return to_return;
6      char c = s.charAt(0);
7      to_return = c + to_return;
8      s=s.substring(1);
9    } // go to X1
10 }
11
12 print(rev2("abc","123"));
```

What does this function do? What does `print(rev2("abc",""));` give?

What are the values of `s` and `to_return` at the beginning of each iteration of the while block?

Example with recursion

Example with recursion:

```
1  String rev2_rec(String s, String to_return) { // X2
2
3    if(s.equals(""))
4      return to_return;
5    char c = s.charAt(0);
6    to_return = c + to_return;
7    s=s.substring(1);
8    return rev2_rec(s,to_return);  // go to X2
9
10 }
11
12 print(rev2_rec("abc","123"));
```

What does this function do?

Note lots of "instances" of the function being evaluated at the same time (although only one is "active")

Comparison of these two functions

Let's put the code for these functions side by side:

```
String rev2(String s, String to_return) { // X2
  /* WITHOUT RECURSION */          |// WITH RECURSION
  while(true) { /* X1 */           |
    if(s.equals(""))               |if(s.equals(""))
      return to_return;            |  return to_return;
    char c = s.charAt(0);          |char c = s.charAt(0);
    to_return = c + to_return;     |to_return = c + to_return;
    s=s.substring(1);              |s=s.substring(1);
    } /* go to X1 */               |return rev2(s,to_return);
}
```

THESE FUNCTIONS DO THE SAME THING!!!

To get the recursive one: remove the while loop, and at the end of the function, call the function again (i.e. keep doing the body of the function)

Let's give it a try

This is the length function that we saw previously. Note that we have to include an extra parameter `to_return` for the function `somefun`, so that we can remember where we have got to.

```
int somefun(String s, int to_return) {
  while(true) {
    if(s.equals("")) return to_return;
    to_return++;
    s=s.substring(1);
  }
}

somefun("hello",0);
```

Challenge: convert this function into a recursive version

Let's give it a try

```
int somefun(String s, int to_return) {
  // while(true) {
  if(s.equals("")) return to_return;
  to_return++;
  s=s.substring(1);
  //}
  return somefun(s,to_return);
}

somefun("hello",0);
```

The while loop has been commented out, and replaced with a call to the same function.

Discussion: which is better?

- While version feels as though it is simpler somehow
- Recursive version has slightly less lines
- While version has the "right" definition of the method; the recursive version needs another parameter
- So it looks like the while loop is better **BUT** later in the course we will meet forms of recursion that are not so easily simulated using while loops.
- For more advanced examples, recursion is more natural, and recursion becomes a very powerful programming tool.

Recursion, another challenge

Convert the following to a recursive version. What do you have to remember?

```
1  String removeAll(char c, String s) {
2    String to_return = "";
3    while(true) {
4      if(s.equals("")) return to_return;
5      // at this point s is not ""
6      char c2 = s.charAt(0);
7      if(c2!=c) { to_return += c2; }
8      s=s.substring(1);
9    }
10   return to_return; // won't be reached
11 }
12
13 print(removeAll('o',"hello"));
```

Another version of length using recursion

```
int somefun_2(String s) {
  if(s.equals("")) return 0;
  else return 1+somefun_2(s.substring(1));
}

somefun("hello",0);
```

See how this executes using pythontutor (linked from Week 1 on module webpage).
With a while loop, we just keep executing the loop; with this form of recursion, we can go away, do something (the recursive call) and come back.

Where are we? Where are we going?

Extra slides

Sorting a string of characters

- Following on from the surgery, an exercise is: write a function `sort` that sorts a string of characters into alphabetical order
- For example, `sort("cxbyaz") ~> "abcxyz"`