COMP 1010- Summer 2015 (A01)

Jim (James) Young

young@cs.umanitoba.ca

jimyoung.ca

String methods!! (Object)

Your string variable type has several built-in **methods** (commands) that you can use.

variableName.**method(parameters);**

String dogName;

dogName = “sprocket”;

// dogName.**method(parameters)**;

dogName.length(); // takes no parameters

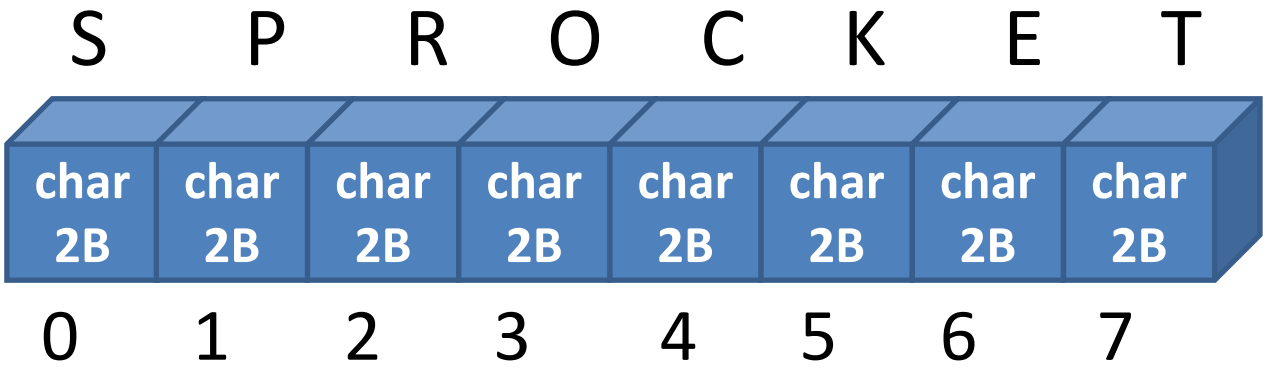
…

String Length:
int variableName.length()

What is the length of this string?

note: the index of the last character is string.length() - 1

off by one error



Get character:

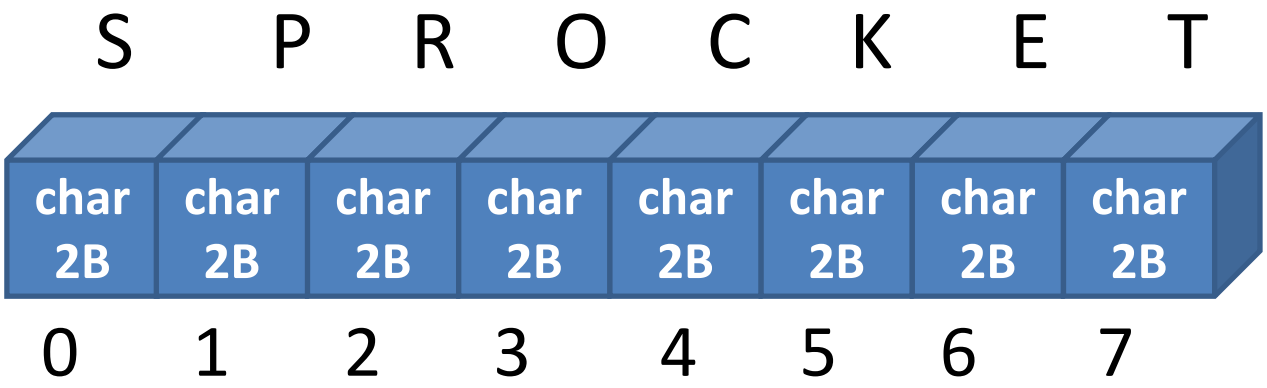
char variableName.charAt(int index)

returns the specific single character at the given index (box #).

e.g.,:

String dogName = “sprocket”;

char secondLetter = dogName.charAt(1);

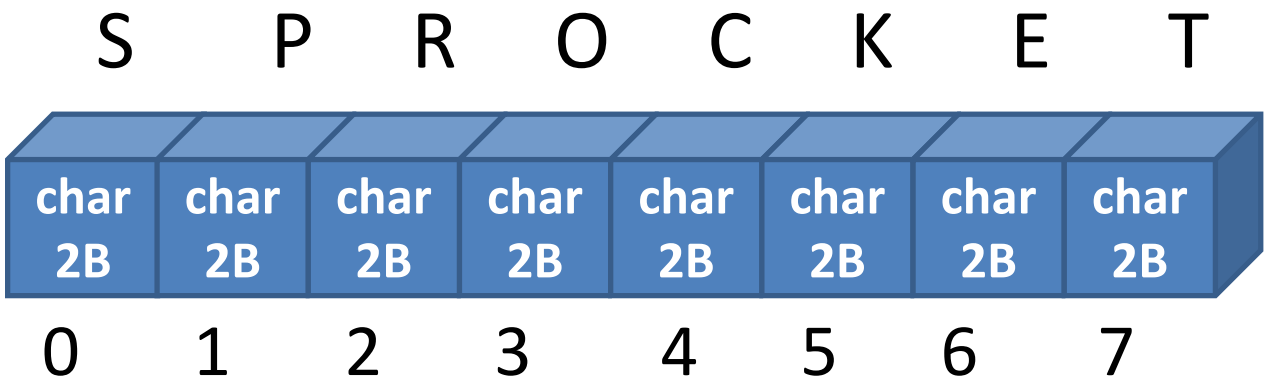


Off-by-one string length error

String s = “SPROCKET”;

```
char lastCharacter = s.charAt(s.length());
```
ERROR

```
char lastCharacter = s.charAt(s.length()-1);
```



Example: put string out one character at a time

Select spacing

Off by one errors!

Increase spacing

Link spacing to mouse

Palindrome tester

Reverse a string

Compare against original

If equal – palindrome!

How to reverse a string? (tricky)

Go through string with a for loop

Get each character

Add to a new string in the opposite order

Let's re-visit the == operator

Compares two values and returns a boolean type

→Cannot be used to compare Strings

it may look like it works sometimes, but

not what you think

Tells you if they are the same object

String is a special case

```
String s1 = “hello.”;
String s2 = “hello.”;
if (s1 == s2) // not what you think!!
{
…
}
```

boolean stringVariable.equals(String)

```
String chantPartA = “hi”;
String chantPartB = “ho”;
boolean areEquals = chantPartA.equals(chantPartB);
// or, = chantPartB.equals(chantPartA);
```

User-defined functions- make your own commands

user-defined functions

user-defined functions: code which the user can write once and then use over and over. E.g., random was written once and you can use it over and over by calling random(number)

create once and use many times!

Example: draw random squares

If the mouse is pressed, clear to white first

If a key is pressed, clear to black first

Random place, random size, random color

Global: max size

Tedius – repetitive

What if I want to make a change, e.g., location range?

Let's make a new command called drawRandomSquare();

user-defined functions

require:

function name: the keyword used to invoke (use) the function. e.g., **ellipse** is a function name

some code: the processing code to run every time the function is invoked (called).

user-defined functions: syntax

```
void functionName ()
{
   …//java code;
}

void sayHello()
{
        text(“Hello”,100,100);
}
```

user-defined functions: syntax

```
void sayHello()
{
        text("Hello",random(width),random(height));
}
```

this is a the name of a new command we just created!

… in your other code

```
sayHello();
```

create outside the draw and start blocks!

note: user-defined functions created outside the other blocks

you can place them before or after them, it is a matter of style

jump around…

when a function is called, Processing remembers where it left off, and jumps to the function. When the function is done, it jumps back.

program flow

```
void sayHello()
{
    text("Hello", random(width),random(height));
}

void draw()
{
    sayHello();
    sayHello();
}
```

run each command in the function (compartment) in order

to run this command, jump up here! To the compartment!

program starts here as usual

run this command first, as usual

when the function is done, come back here and continue where we left off

and run the function again!

Example: draw random squares

Let’s make a new command called drawRandomSquare();

functions and scope

note: scope is the range within which a variable exists. Outside of that scope, you cannot access or work with that variable.

blocks define scope!

Functions each have their own scope – data in one method is not visible to other methods – try it

variable scope is limited to its function


```
void sayHello() {
  text(message,100,100);
}

void draw() {
  String message = “Hey there!”;
  sayHello();
}
```

note: a variable declared in one function is not **accessible** from another function!

you cannot read it, or change it.

No nesting of scopes here

the same variable name?

```
void sayHello()
{
  String message = “say hi”;
  println(message);
}

void draw()
{
  String message = “Hey there!”;
  sayHello();
  println(message);
}
```

variables in different scopes can have the same name, but they do not share data – they are completely separate.