CMPSC 100

Computational Expression

Data Type	Size	Min value	Max Value
byte	1 byte	-128	127
short	2 bytes	-32,768	32,767
int	4 bytes	-2,147,483,648	2,147,483,647
long	8 bytes	- a lot	+ a lot
float	4 bytes	7 decimals	7 decimals
double	8 bytes	15 decimals	15 decimals
char	2 bytes	0	65,536
boolean	(not important)	0 (true)	1 (false)

“primitive” data types

Data Type	Size	Min value	Max Value
String	Various	?	?
Scanner	Various	?	?

“reference” data type

Reference types

- Two ways to tell a “reference” type:
  - It’s not in the “primitives” table
  - The first letter of the type is capitalized

Example:

Isn’t in the “primitives” table

```
String name = “G. Wiz”
```

Is capitalized unlike int or double

```
String name = “G. Wiz”;
```

Here, name “refers to” an object made of a set of characters

```
char ‘G’ → 71
char ‘.’ → 46
char ‘ ’ → 32
char ‘W’ → 87
char ‘i’ → 105
char ‘z’ → 122
```

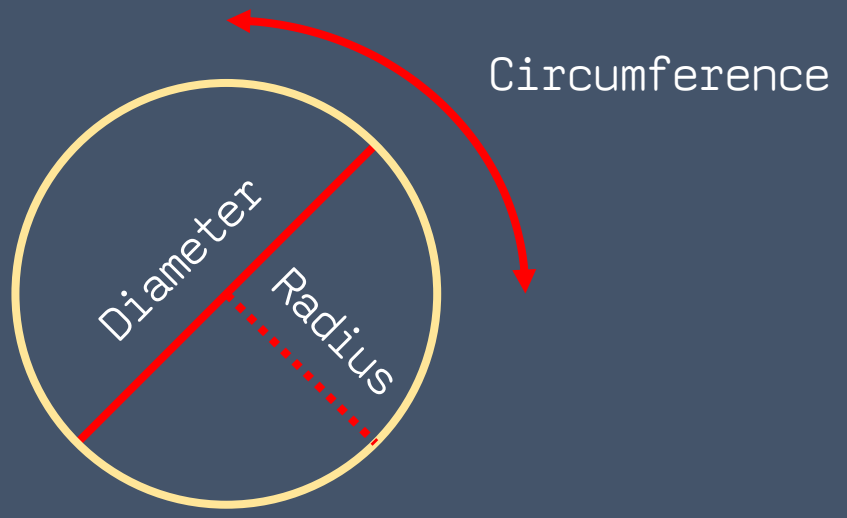
Meanwhile, somewhere in memory…

71	46	32	87	105	122

Reference types: String

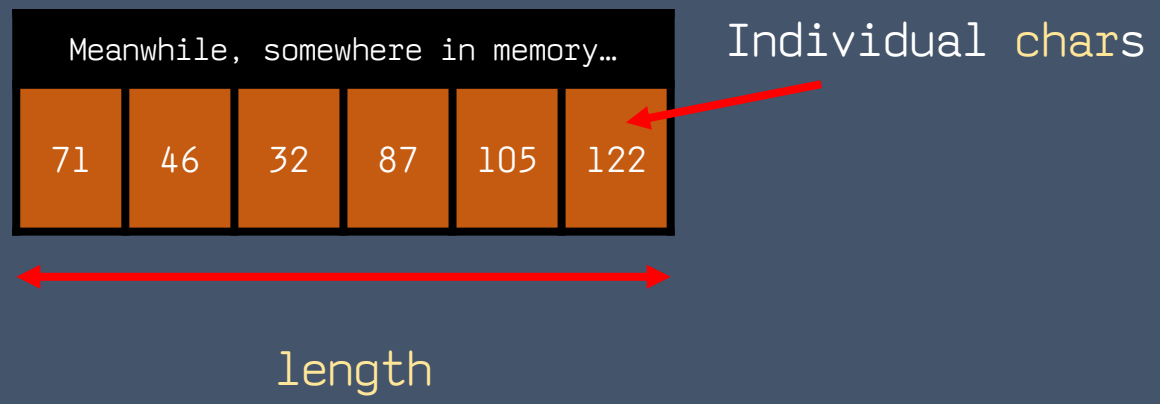
- A String is a reference type that represents an object with many different attributes and features called “methods”

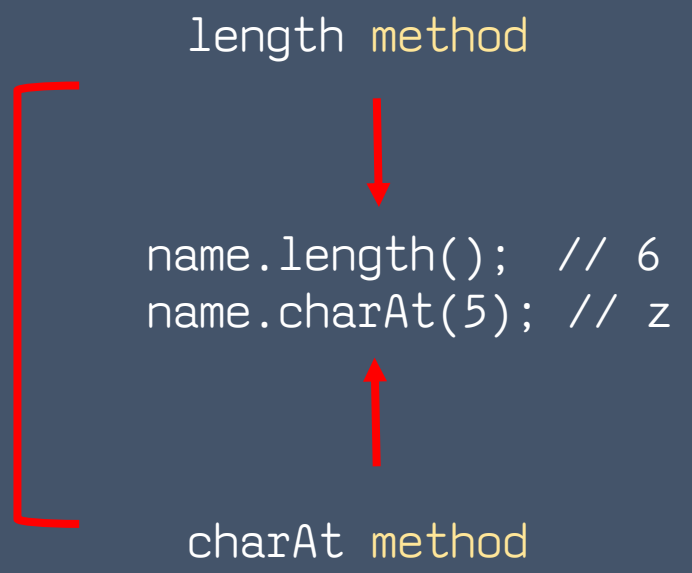
Circumference	$2\pi r$ (πd)
Diameter	$2r$
Area	πr^2



A circle as an object

```
String name = “G. Wiz”;
```





Reference types: String

- Formally, we create strings like this:

```
String name = new String("G. Wiz");
```

initialization/"instantiation"

Reference types: String

- However, Java makes the following equivalent:

```
String name = “G. Wiz”;
```

initialization/“instantiation”

`String (String str)`
Constructor: creates a new string object with the same characters as `str`.

`char charAt (int index)`
Returns the character at the specified `index`.

`int compareTo (String str)`
Returns an integer indicating if this string is lexically before (a negative return value), equal to (a zero return value), or lexically after (a positive return value), the string `str`.

`String concat (String str)`
Returns a new string consisting of this string concatenated with `str`.

`boolean equals (String str)`
Returns true if this string contains the same characters as `str` (including case) and false otherwise.

`boolean equalsIgnoreCase (String str)`
Returns true if this string contains the same characters as `str` (without regard to case) and false otherwise.

`int length ()`
Returns the number of characters in this string.

`String replace (char oldChar, char newChar)`
Returns a new string that is identical with this string except that every occurrence of `oldChar` is replaced by `newChar`.

`String substring (int offset, int endIndex)`
Returns a new string that is a subset of this string starting at index `offset` and extending through `endIndex-1`.

`String toLowerCase ()`
Returns a new string identical to this string except all uppercase letters are converted to their lowercase equivalent.

`String toUpperCase ()`
Returns a new string identical to this string except all lowercase letters are converted to their uppercase equivalent.

Some objects contain "powers" (called methods) that we can use whenever we've made a variable of that type. String is no different.

We summon these using the dot operator:

```
String name = "G. Wiz";
int nameLen = name.length(); // 6
char fifth = name.charAt(4); // 'i'
String lastName = name.substring(2,nameLen);
String lastName = name.substring(2,5);
```

Methods

object method argument(s)

```
int nameLen = name.length();
```

```
int length ()
```
Returns the number of characters in this string.

```
char fifth = name.charAt(4);
```

```
char charAt (int index)
```
Returns the character at the specified `index`.

```
String lastName = name.substring(2,nameLen);
```

```
String lastName = name.substring(2,6);
```

```
String substring (int offset, int endIndex)
```
Returns a new string that is a subset of this string starting at index `offset` and extending through `endIndex-1`.

Try it out!

cd to your class activities folder

perform a git pull download master

This should give you a new folder called sandbox which will allow you to just try out code!

Try it out!

Create a `String` of more than 5 characters called `myString`

Find `myString.charAt(3);`

Find `myString.length();` and store in `int len`

Find `myString.charAt(len);`

Bonus: why did you get the result you did here?

Print `myString.toUpperCase();`

Try another method in the table on page 105!

Activity

`cd` to the `activity-05` folder

Your job today is to transform a fool into someone much wiser

Reference types: Scanner

A Scanner is an object which scans an input source for input.

Today we're using it to read a file from our secondary memory (I've provided the file).

Reference types: Scanner

This is not part of the Java API (java.lang), so we add it to our program using import statements at the top of our Java file.

Another reference type: File

Another reference type: an Exception

```
import java.io.File;
import java.io.FileNotFoundException;
import java.util.Scanner;
```

The Scanner reference type

```
// Create identifiers for input
File file = null;
Scanner input = null;
// Read input from file
try {
  file = new File("input/words.list");
  input = new Scanner(file);
} catch (FileNotFoundException noFile) {
  System.exit(0);
}
```

Here, we have to make a copy of it to “initialize it” and point it at an input source: in this case a File object that I’ve provided

Activity

Using Strings and the substring method, change each word one letter at a time.

When finished, uncomment the commented lines at the end of the main method

We'll do the first one together

Data conversion: casting

```
int cookies = 10;
int students = 14;
double cookieShare = cookies/students;
```

```
> 0
```

Data conversion: casting

```
int cookies = 10;
int students = 14;
double cookieShare = (double)cookies/students;

> 0.71…
```

Data conversion: assignment conversion

```
int cookies = 10;
double crumbs = 10/2;
System.out.println(crumbs);
```

```
> 5.0
```

Data conversion: promotion

```
double cookies = 10;
int students = 14;
System.out.println(cookies/students);
```

```
> 0.71…
```