CMPSC 100

Computational Expression

Reminders and announcements

Due 8 April 2020:

- Practical 08
- Lab 09

Due 13 April 2020:

- Practical 09
- Project Proposal

This is a change to the schedule!

Upcoming:

- Lab 10 (today)
- Quiz 2 (13 April 2020; due 17 April 2020)

Recap: `while` statements (loops)

- `while` statements alter the flow of control

Loop until condition is no longer true

FLOW OF CONTROL

```
while (count < 3) {
  System.out.println("In a loop!");
  count++;
}
```

Each "run" of the loop is called an "iteration"

Recap: `while` statements (loops)

- Various objects have the ability to be iterators

an object that has methods that allow you to process a collection of items one at a time.

(fancy book definition)

Recap: `while` statements (loops)

- These iterators simply give us the ability to ask a question such as “does this file have another line?”
- And, we can do this until the answer is no longer true

Examples:

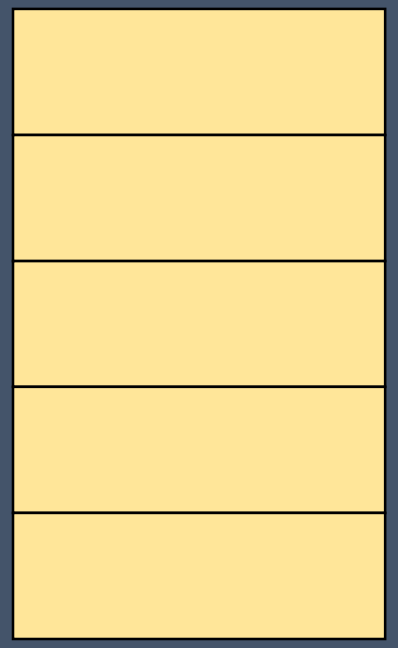
```
hasNextLine()
hasNext()
```

Both are methods of Scanner

Data Structures

- A data structure is an object that allows us to store multiple pieces of data in one place so that we can interact with them later
- First, we'll learn about the ArrayList and (next week) the "dimensional array"

Data Structures: `ArrayList`



Initialization keyword

Identifier

Reference type/object

Reference type/object

Arguments (none)

```
ArrayList<String> catNames = new ArrayList<String>();
```

Object type to store

Data Structures: ArrayList

The Boss	0
Snooze Magoo	1
Ulysses	2
Mr. U	3
The Mane Man	4

```
ArrayList<String> catNames = new ArrayList<String>();
catNames.add("The Boss");
catNames.add("Snooze Magoo");
catNames.add("Ulysses");
catNames.add("Mr. U");
catNames.add("The Mane Man");
```

I can't believe you told them my secret names

Data Structures: ArrayList

The Boss	0
Snooze Magoo	1
Ulysses	2
Mr. U	3
The Mane Man	4

- Each item is assigned its own space in the ArrayList
- Each space has an index (aka a “position”) which corresponds to its spot in the list
- Indexes start counting, like Java’s rule, from 0
- ArrayLists are, essentially, unlimited in size

By the time you’ve made it upstairs, I’ll have torn up the couch.

Data Structures: `ArrayList`

The Boss	0
Snooze Magoo	1
Ulysses	2
Mr. U	3
The Mane Man	4

```
catNames.size();
> 5
catNames.get(2)
> Ulysses
catNames.indexOf("The Mane Man");
> 4
```

Data Structures: `ArrayList`

The Boss	0
Snooze Magoo	1
Ulysses	2
Mr. U	3
The Mane Man	4

```
int index = 0;
String name;

while (index < catNames.size()){
  name = catNames.get(index);
  System.out.println(name);
  index++; // Don't forget to increment your index!
}
```

```
> The Boss
  Snooze Magoo
  Ulysses
  Mr. U
  The Mane Man
```

Data Structures: `ArrayList`

If ArrayLists can hold objects, what does that mean?

Let's imagine a Book. It has:

- Title
- Author
- Page count

Data Structures: `ArrayList`

```
ArrayList<Book> library = new ArrayList<Book>();
Book book = new Book();
book.setTitle(“Old Possum's Book of Practical Cats”);
book.setAuthor(“T.S. Eliot”);
book.setPageCount(56);
library.add(book);
```

Data Structures: ArrayList

```
Old Possum's Book of Practical Cats
T.S Eliot
56 pages
```

```
Book checkedOut = library.get(0);
```

If the library object holds objects of Book type

"Getting" an entry from it should return a Book type

Data Structures: `ArrayList`

```
ArrayList<E>()
```
Constructor: creates an initially empty list.

```
boolean add(E obj)
```
Inserts the specified object to the end of this list.

```
void add(int index, E obj)
```
Inserts the specified object into this list at the specified index.

```
void clear()
```
Removes all elements from this list.

```
E remove(int index)
```
Removes the element at the specified index in this list and returns it.

```
E get(int index)
```
Returns the object at the specified index in this list without removing it.

```
int indexOf(Object obj)
```
Returns the index of the first occurrence of the specified object.

```
boolean contains(Object obj)
```
Returns true if this list contains the specified object.

```
boolean isEmpty()
```
Returns true if this list contains no elements.

```
int size()
```
Returns the number of elements in this list.

Data Structures: ArrayList

- The ArrayList cannot hold primitives
- However, it can hold objects
  - Even reference types/objects we create from scratch!
- We can use it to store many different copies of reference objects
- Last, but not least: we have to import it!

```
import java.util.ArrayList;
```

Part of the util package (like Scanner)

Data Structures: `ArrayList`

We're going to use this data structure in an activity

cd to the activity-12 folder

If you're having issues getting the new content try a git stash or commit your current repository and then pull the new content!

Activity

To revisit things we know about M & Ms

- Come in 6 colors:
  - Brown
  - Yellow
  - Red
  - Green
  - Orange
  - Blue
- Blue is still, by far, the best
- Each bag has somewhere around 20 candies.

Activity

```
Bag()
        Constructor: creates an empty Bag object

void setColorCount(String color, int count)
        Sets the count of candies of specified color

int getColorCount(String color)
        Gets the count of candies of specified color

void setTotalPieces(int total)
        Sets the reported number of total candies per bag from file

int getTotalPieces()
        Gets total pieces in bag as reported

boolean getVerifiedTotal()
        Returns result of comparison between reported and actual total of candies

String toString()
        Returns a report of the contents of the bag
```

Activity

Our input file has a given format:

Each line represents one bag
Each line's format is #B, #Y, #R, #G, #O, #B, #Total