A large, spreading tree with many trunks and branches, set against a bright sky.

# CSCI 1102 Computer Science 2

Meeting 3: Thursday 2/4/2021  
More on Java

# Python & Dynamic Typing

Me, looking at a list of advantages of dynamic typing.

```
def f(a):  
    return (a[0], a + 6)
```



```
static int fib(int n) {  
    if (n < 2)  
        return 1;  
    else  
        return fib(n - 1) + fib(n - 2);  
}
```

```
def fib(n):  
    if (n < 2):  
        return 1  
    else:  
        return fib(n - 1) + fib(n - 2)
```

Two wildly inefficient Fibonacci functions.

# Run-times on VMs for fib(38)

```
> javac Fib.java  
> time java Fib 38  
fib(38) = 63245986
```

```
real      0m0.327s ←  
user      0m0.322s  
sys       0m0.035s  
> time python fib.py 38  
63245986
```

```
real      0m15.247s ←  
user      0m14.767s  
sys       0m0.125s  
>
```

The Python time includes time for translation.

# But Java is Huge & Wildly Complicated



# Today

- Message-Passing Style
- Frequency Table Example
- Abstract Data Types

# Example: Who is taking CS2?

S	School	Major	Grad Year			
has@bc.edu	Morrissey College, A	computer science b.	2023	MCAS	CSBA	2024
i@bc.edu	Morrissey College, A	computer science b.	2024	MCAS	PHIL	2022
ne@bc.edu	Morrissey College, A	computer science b.	2024	MCAS	CSBA	2024
c.edu	Morrissey College, A	computer science b.	2022	MCAS	CSBA	2022
@bc.edu	Morrissey College, A	philosophy	2022	MCAS	CSBS	2024
@bc.edu	Morrissey College, A	computer science b.	2024	MCAS	CSBA	2023
y@bc.edu	Morrissey College, A	computer science b.	2023	CSOM	MATH	2022
dia@bc.edu	Morrissey College, A	mathematics bs	2022	MCAS	ISYS	2023
an@bc.edu	Carroll School of Ma	information systems	2023	CSOM	BIOL	2021
is.5@bc.edu	Morrissey College, A	biology	2021	MCAS	UNDC	2023
c.edu	Carroll School of Ma	undecided	2023	MCAS	CSBA	2024
@bc.edu	Morrissey College, A	computer science b.	2024	MCAS	CSBA	2024
bc.edu	Morrissey College, A	computer science b.	2024	MCAS	CSBS	2024
za@bc.edu	Morrissey College, A	computer science b.	2024	MCAS	CSBS	2024
.edu	Morrissey College, A	computer science b.	2024	MCAS	CSBA	2024
.2@bc.edu	Morrissey College, A	computer science b.	2024	MCAS	MATH	2024
@bc.edu	Morrissey College, A	computer science b.	2024	MCAS	CSBS	2024
es@bc.edu	Morrissey College, A	computer science b.	2024	MCAG	BIOL	2026
				MCAS	ECON	2023

# Count Enrolled Students

The screenshot shows a Java application running in an IDE. The code in `Count.java` reads a CSV file named `05.csv` and counts the number of students. The output window shows the student names and the final count.

```
public class Count {
    public static void main(String[] args) {
        int count = 0;
        String student;
        In in = new In(args[0]);

        student = in.readLine();
        while (student != null) {
            count++;
            System.out.format("%s\n", student);
            student = in.readLine();
        }
        in.close();
        System.out.format("There are %d students.\n", count);
    }
}
```

Run: Count

MCAS	MATH	2021
MCAS	MATH	2021
There are 98 students.		
Process finished with exit code 0		

Git: Build completed successfully in 1 s 116 ms (moments ago)

Event Log: 12:20 LF UTF-8 2 spaces master

# Message Passing Style

- In the Count example, `in` is an *input channel* and `readLine` is a function.
- In standard mathematical style, we read a line from the input channel using standard function call notation: `readLine(in)`
- `static` functions are called using the standard notation.

# Message Passing Style

- Non-static (or “dynamic”) functions are called in message-passing style: `in.readLine()`

```
11
12     public static void main(String[] args) {
13         int count = 0;
14         String student;
15         In in = new In(args[0]);
16
17         student = in.readLine(); ←
18         while (student != null) {
19             count++;
20             System.out.format("%s\n", student);
21             student = in.readLine();
22         }
23         in.close(); ←
```

# Message Passing Style

- The input channel `in` is understood to be a structure (*aka*, an “object”) containing functions

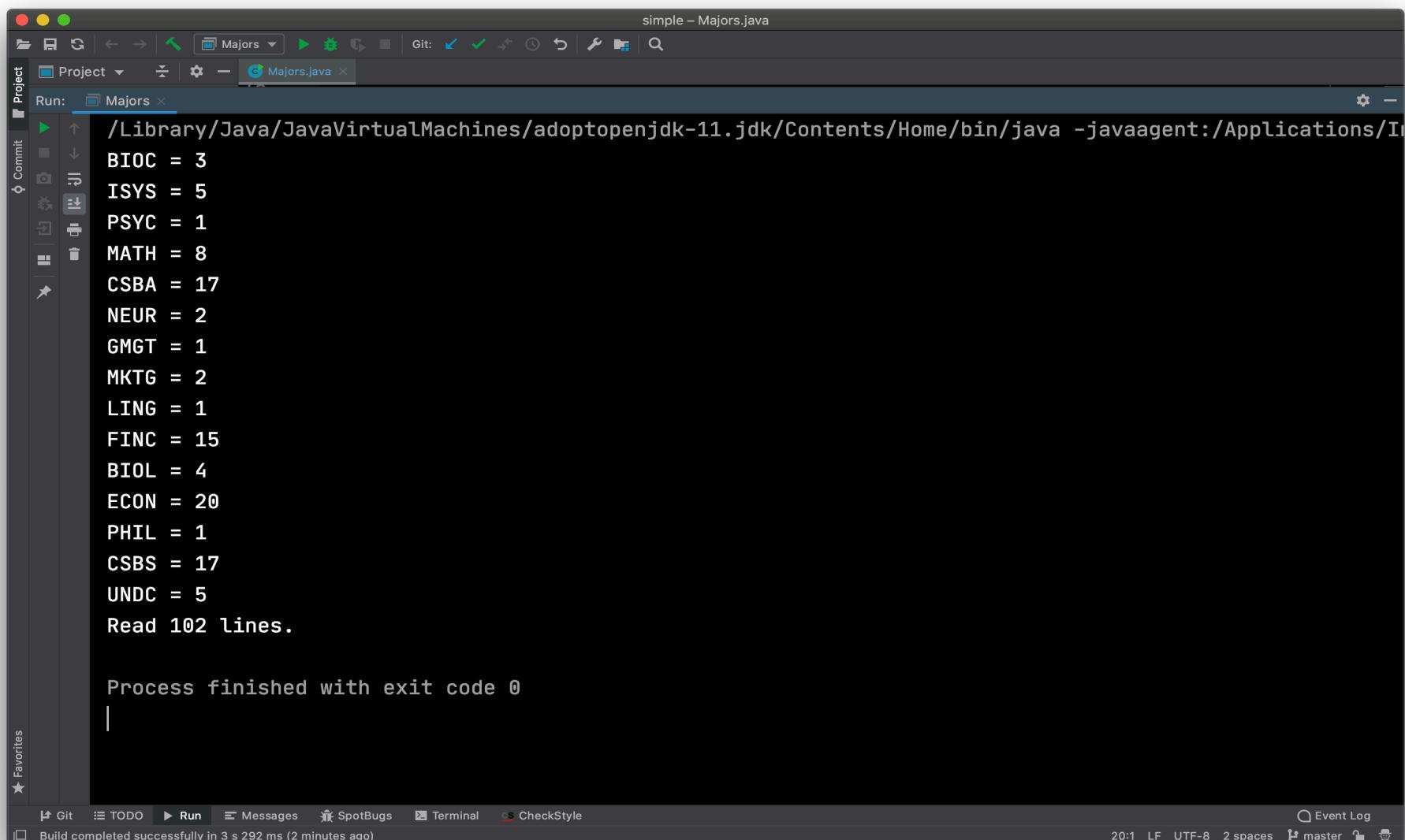
```
11
12     public static void main(String[] args) {
13         int count = 0;
14         String student;
15         In in = new In(args[0]);
16
17         student = in.readLine(); ←
18         while (student != null) {
19             count++;
20             System.out.format("%s\n", student);
21             student = in.readLine();
22         }
23         in.close(); ←
```

# Message Passing Style

- `in.readLine` is sometimes interpreted as “*send the `readLine` message to channel `in`*”

```
11
12     public static void main(String[] args) {
13         int count = 0;
14         String student;
15         In in = new In(args[0]);
16
17         student = in.readLine(); ←
18         while (student != null) {
19             count++;
20             System.out.format("%s\n", student);
21             student = in.readLine();
22         }
23         in.close(); ←
```

# Compute a Frequency Table of Majors



The screenshot shows a Java application running in an IDE. The title bar indicates the project is named "simple" and the file is "Majors.java". The run configuration is set to "Majors". The terminal output window displays the following text:

```
simple - Majors.java
Run: Majors ×
/Library/Java/JavaVirtualMachines/adoptopenjdk-11.jdk/Contents/Home/bin/java -javaagent:/Applications/I
BIOC = 3
ISYS = 5
PSYC = 1
MATH = 8
CSBA = 17
NEUR = 2
GMGT = 1
MKTG = 2
LING = 1
FINC = 15
BIOL = 4
ECON = 20
PHIL = 1
CSBS = 17
UNDC = 5
Read 102 lines.

Process finished with exit code 0
|
```

The IDE interface includes a Project tree on the left, a Run configuration dropdown, and various toolbars and status bars at the bottom.

# Map/Dictionary -- Python

```
2      # PYTHON  
3  
4      frequency = { "BIOC" : 3, "ISYS" : 5} # etcetera  
5  
6      biochemistry = frequency["BIOC"]  
7
```

# Map/Dictionary -- OCaml

```
2 (* OCaml *)
3
4 module Map = Map.Make(String)
5
6 let frequency = Map.add "BIOC" 3 (Map.add "ISYS" 5 Map.empty)
7
8 let biochemistry = Map.find "BIOC" frequency
9
```

# Map/Dictionary -- Java

java.time.format  
java.time.temporal  
java.time.zone  
**java.util** ←  
java.util.concurrent  
java.util.concurrent.atomic  
java.util.concurrent.locks  
java.util.function  
java.util.jar

## java.util

### Interfaces ←

Collection  
Comparator  
Deque  
Enumeration  
EventListener  
Formattable  
Iterator  
List  
ListIterator  
Map  
Map.Entry  
NavigableMap  
NavigableSet  
Observer  
PrimitiveIterator  
PrimitiveIterator.OfDouble  
PrimitiveIterator.OfInt  
PrimitiveIterator.OfLong  
Queue

## Java™ Platform, Standard Edition 8 API Specification

This document is the API specification for the Java™ Platform, Standard Edition.

See: [Description](#)

### Profiles

- compact1
- compact2
- compact3

### Packages

Package	Description
<a href="#">java.applet</a>	Provides the classes necessary to create to communicate with its applet context.
<a href="#">java.awt</a>	Contains all of the classes for creating user interface components and images.
<a href="#">java.awt.color</a>	Provides classes for color spaces.
<a href="#">java.awt.datatransfer</a>	Provides interfaces and classes for transferring data between applications.

# Map/Dictionary -- Java

java.time.format  
java.time.temporal  
java.time.zone  
**java.util** ←  
java.util.concurrent  
java.util.concurrent.atomic  
java.util.concurrent.locks  
java.util.function  
java.util.jar

Dictionary  
DoubleSummaryStatistics  
EnumMap  
EnumSet  
EventListenerProxy  
EventObject  
FormattableFlags  
Formatter  
GregorianCalendar  
HashMap ←  
HashSet  
Hashtable  
IdentityHashMap  
IntSummaryStatistics  
LinkedHashMap  
LinkedHashSet  
LinkedList  
ListResourceBundle  
Locale  
Locale.Builder  
Locale.LanguageRange  
LongSummaryStatistics  
Objects  
Observable

## Java™ Platform, Standard Edition 8 API Specification

This document is the API specification for the Java™ Platform, Standard Edition.

See: [Description](#)

### Profiles

- compact1
- compact2
- compact3

### Packages

Package	Description
<b>java.applet</b>	Provides the classes necessary to create to communicate with its applet context.
<b>java.awt</b>	Contains all of the classes for creating and images.
<b>java.awt.color</b>	Provides classes for color spaces.
<b>java.awt.datatransfer</b>	Provides interfaces and classes for trans applications

# Map/Dictionary -- Java

```
6      // Java  
7  
8      Map<String, Integer> frequency = new HashMap();  
9  
10     frequency.put("BIOC", 3);  
11     frequency.put("ISYS", 5);  
12  
13     Integer biochemistry = frequency.get("BIOC");  
14
```

# Map/Dictionary -- Java

```
6 // Java interface  
7 // An interface  
8 Map<String, Integer> frequency = new HashMap();  
9  
10 frequency.put("BIOC", 3);  
11 frequency.put("ISYS", 5);  
12  
13 Integer biochemistry = frequency.get("BIOC");  
14
```

A class



# NB: This Works but Avoid if Possible

```
6 // JavaA class  
7  
8 HashMap<String, Integer> frequency = new HashMap(); A class  
9  
10 frequency.put("BIOC", 3);  
11 frequency.put("ISYS", 5);  
12  
13 Integer biochemistry = frequency.get("BIOC");  
14
```

# Map/Dictionary -- Java

```
6 // Java      Key      Value
7
8 Map<String, Integer> frequency = new HashMap();
9
10 frequency.put("BIOC", 3);
11 frequency.put("ISYS", 5);
12
13 Integer biochemistry = frequency.get("BIOC");
14
```



# Digression 1: – Base Types & Wrapper Classes

```
6 // Java      A reference type
7
8 Map<String, Integer> frequency = new HashMap();
9
10 frequency.put("BIOC", 3);
11 frequency.put("ISYS", 5); >>> Values of base type int
12
13 Integer biochemistry = frequency.get("BIOC");
14
```

# Base Types & Wrapper Classes

Base Type	Wrapper Class/Type
int (32-bit integers)	Integer
long (64-bit integers)	Long
short (16-bit integers)	Short
byte (8-bit integers)	Byte
float (32-bit floats)	Float
double (64-bit floats)	Double
char	Character
boolean	Boolean

java.io
java.lang
java.lang.annotation
java.lang.instrument
java.lang.invoke
java.lang.management
java.lang.ref
java.lang.reflect
java.math
java.net
java.nio
<b>Classes</b>
Boolean
Byte
Character
Character.Subset
Character.UnicodeBlock
Class
ClassLoader
ClassValue
Compiler
Double
Enum
Float
InheritableThreadLocal
Integer
Long
Math
Number
Object

boolean	Returns the hash code value for this map.
<b>Set&lt;K&gt;</b>	
default <b>V</b>	
<b>V</b>	
void	
default <b>V</b>	
<b>V</b>	
default boolean	

# Base Types & Wrapper Classes

```
16  
17     Integer wrapped = new Integer(343);  
18     String succeed = wrapped.toString(); // Integer has a toString fun Scree  
19  
20     String fail = 343.toString();  
21  
22     Integer wrapped = 343;           // Autoboxing  
23     String succeed = wrapped.toString();  
24  
25     int succeed = 12 * wrapped;      // Auto-unboxing  
26
```

# Digression 2: Exceptions

- Each line of the input file has two strings and one integer;
- The algs4 `In` class has `readString` and `readInt` functions;
- What happens if there is no string or int to read?

### **readString**

```
public String readString()
```

Reads the next token from this input stream and returns it as a `String`.

**Returns:**

the next `String` in this input stream

**Throws:**

`NoSuchElementException` – if the input stream is empty

If an exception  
is thrown we  
need to catch it.

### **readInt**

```
public int readInt()
```

Reads the next token from this input stream, parses it as a `int`, and returns the `int`.

**Returns:**

the next `int` in this input stream

**Throws:**

`NoSuchElementException` – if the input stream is empty

`InputMismatchException` – if the next token cannot be parsed as an `int`

java.time.format  
java.time.temporal  
java.time.zone  
**java.util**  
java.util.concurrent  
java.util.concurrent.atomic  
java.util.concurrent.locks  
java.util.function  
java.util.jar  
java.util.logging

Formatter.BigDecimalLayoutForm  
Locale.Category  
Locale.FilteringMode

## Exceptions

ConcurrentModificationException  
DuplicateFormatFlagsException  
EmptyStackException  
FormatFlagsConversionMismatchException  
FormatterClosedException  
IllegalFormatCodePointException  
IllegalFormatConversionException  
IllegalFormatException  
IllegalFormatFlagsException  
IllegalFormatPrecisionException  
IllegalFormatWidthException  
IllformedLocaleException  
InputMismatchException  
InvalidPropertiesFormatException  
MissingFormatArgumentException  
MissingFormatWidthException  
MissingResourceException  
NoSuchElementException  
TooManyListenersException  
UnknownFormatConversionException  
UnknownFormatFlagsException

default **V**

**putIfAbsent(K key, V value)**

If the specified key is not already associated with the given value and returns null, else returns the current value.

**V**

**remove(Object key)**

Removes the mapping for a key from this map if present.

default boolean

**remove(Object key, Object value)**

Removes the entry for the specified key or returns false if absent.

default **V**

**replace(K key, V value)**

Replaces the entry for the specified key or inserts a new entry if absent.

default boolean

**replace(K key, V oldValue, V newValue)**

Replaces the entry for the specified key or inserts a new entry if absent.

default void

**replaceAll(BiFunction<? super K, ? super V, V> function)**

Replaces each entry's value with the result of applying the provided function to its current key and value. If the function returns null for a key, then the entry is removed or not processed if it was added by another thread during iteration.

int

**size()**

Returns the number of key-value mappings.

**Collection<V>**

**values()**

Returns a **Collection** view of the values contained in this map.

## Method Detail

**size**

# Handling Exceptions

```
27  
28     try {  
29         ... code that may throw an exception ...  
30     }  
31     catch (Exception e) {  
32         ... recovery code ...  
33     }  
34
```

# Code

# Compute a Frequency Table of Majors

The screenshot shows a Java application window titled "simple – Majors.java". The code in the editor is as follows:

```
public static void main(String[] args) {
    int count = 0;
    Map<String, Integer> frequency = new HashMap<String, Integer>();
    In in = new In(args[0]);
    try {
        while (true) {
            String school = in.readString();
            String major = in.readString();
            if (frequency.containsKey(major))
                frequency.put(major, frequency.get(major) + 1);
            else
                frequency.put(major, 1);
            int year = in.readInt();
            count++;
        }
    }
    catch (NoSuchElementException e) {
        in.close();
    }
    frequency.forEach((key, value) ->
        System.out.format("%s = %d\n", key, value));
    System.out.format("Read %d lines.\n", count);
}
```

The IDE interface includes a Project sidebar, a Git status bar at the top, and various toolbars and status bars at the bottom.