CS151 Intro to Data Structures

Java Basics

Data Structures

What you'll learn:

1. Data Structures
2. Programming and Debugging Skills
3. Designing Complex Programs

Today

Part 1:

- Administrative info
- Syllabus
- Tips for Success

Part 2:

- Java Basics
- Exercises: in class coding
    - Ex 1: classes, arrays, loops
    - Ex 2: input / output, exceptions

Lab:

- Review Java Basics
- Exceptions
- Input / Output

Administrivia

- Course website
  - BMC-CS-151.github.io
    - Assignments and lab instructions, syllabus
    - Recordings!
    - Code from lecture
- Piazza:
  - Asynchronous communication
  - Can post anonymously (anonymous just to classmates)
  - Answer your peers questions!
    - Counts for participation grade
- Gradescope:
  - Entry code **WNNG5Z**
  - Submit all assignments
  - Can request re-grade requests
  - WHAT YOU SEE IS WHAT YOU GET
- Optional Textbook
- 2.7 GPA requirement for CS Major

Schedule

- Lecture Mon and Wednesday
- Homeworks <u>due on Mondays</u>
  - Two late days allowed
  - After two days, the submission window will be closed.
- Lab Park 230/W 2:40pm-4:00pm
- Midterm: March 18 (Wed after Spring break)
- Final Exam: Date TBD

Grade Breakdown

- Homeworks: 30%
- Midterm: 30%
- Final: 30%
- Labs: 5%
- Participation: 5%

Labs

- Practice what we learned in lecture
- Will sometimes be a start to your HW
- Should be shorter than HWs
- Submission:
  - Get checked off manually
    - this means TA or Prof will have a conversation with you and assign points when you've finished
  - Autograde: some assignments will be graded automatically by pre written test cases on gradescope

Homeworks

- All submitted on Gradescope and autograded
- Longer assignments

Average Workload
(reported by past students)

HW0: 6 hours

HW1: 11 hours

HW2: 19 hours

HW3: 7 hours

HW4: 6 hours

HW5: 13 hours

HW6: 16 hours

HW7: 20 hours

HW8: 15 hours

How to succeed

1. DO YOUR HOMEWORK

2. Start early

3. Ask for help
    a. Piazza
    b. TA and Professor office hours

Tips for Success

- Get things working in the smallest case and continue to
- Iterate on assignment until the deadline
- Compile early and often
- Prioritize homeworks
- START EARLY
- How to use AI effectively
  - high level planning vs low level implementation

Prerequisites

- Comfortable in command line and vim
  - If you’re not - complete Lab0

- Java
  - basic syntax - we’ll review some today

Your Environment

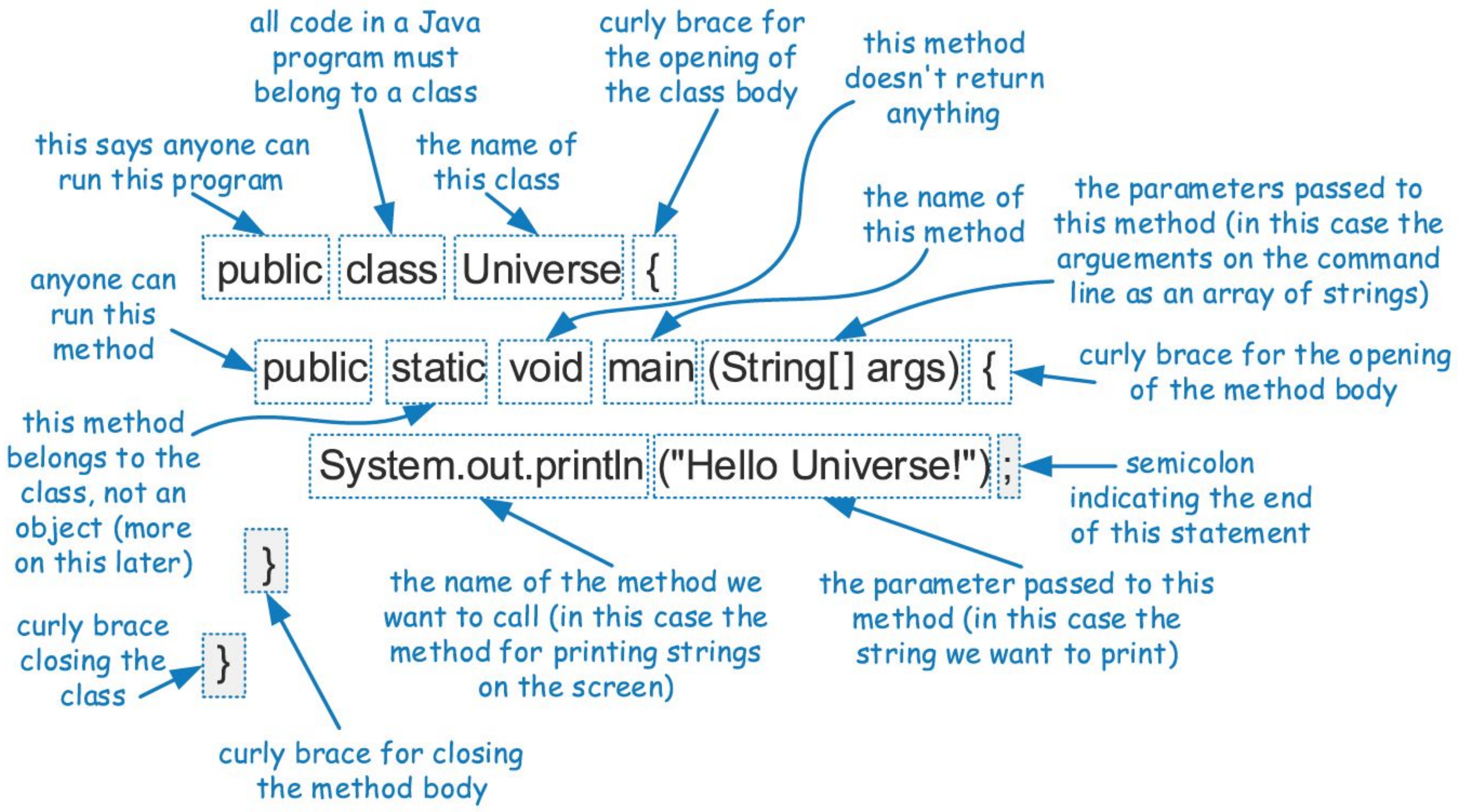
- CS server account
  - Make sure you can log in
  - Email David Diaz if encountering issues (ddiaz1@brynmawr.edu)

- Lab00: ideally completed already, getting up and running with vim and linux

- Software: vim, Java, or just ssh

Part 2: Java Basics and CS1 Review

An Example Program


all code in a Java program must belong to a class
curly brace for the opening of the class body
this method doesn't return anything
this says anyone can run this program
the name of this class
the name of this method
the parameters passed to this method (in this case the arguements on the command line as an array of strings)
anyone can run this method
curly brace for the opening of the method body
this method belongs to the class, not an object (more on this later)
semicolon indicating the end of this statement
the name of the method we want to call (in this case the method for printing strings on the screen)
the parameter passed to this method (in this case the string we want to print)
curly brace closing the class
curly brace for closing the method body

```
public class Universe {
    public static void main(String[] args) {
        System.out.println("Hello Universe!");
    }
}
```

Java: A compiled language

- Java program in .java (source code)
- Compiler create .class file (byte code)
- Java Virtual Machine (JVM) execute the code

Java Basics

- Name of main class and file must agree
  - `class Driver` <--> `Driver.java`

- Compilation
  - `javac Driver.java`

- Execution
  - `java Driver`

Components of a Java Program

- Statements are placed in *methods*, that belong to class definitions.

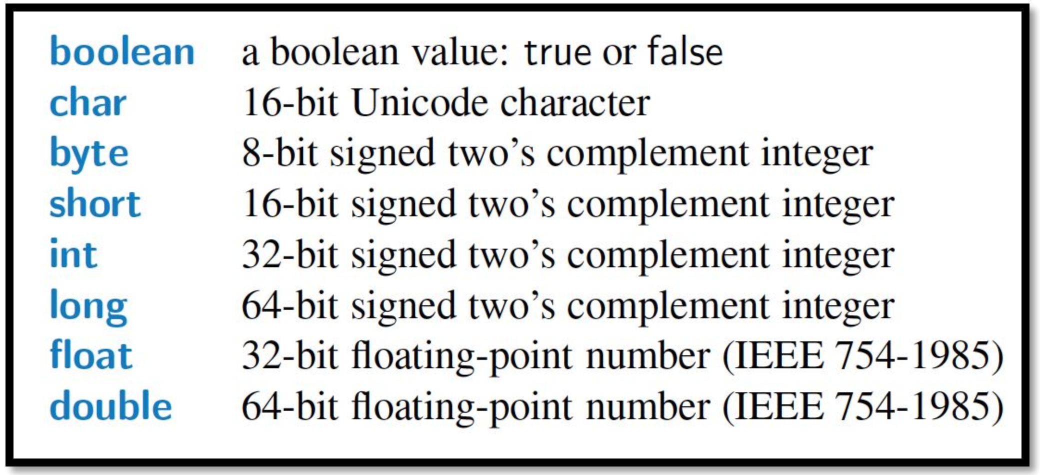
- The static method named `main` is the first method to be executed when running a Java program.

- Any set of statements between the braces `{` and `}` define a program block.

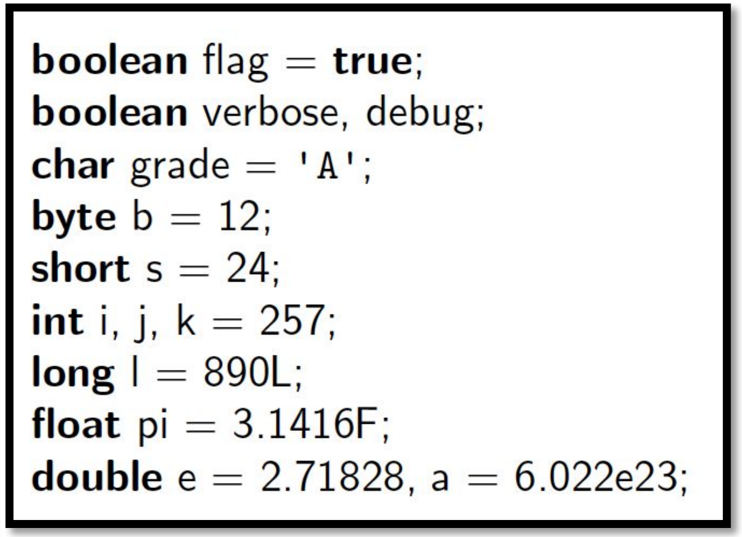
Base/Primitive Types

- Variables must have types
  - base type
- Types define memory used to store the data

- Primitives:



boolean	a boolean value: true or false
char	16-bit Unicode character
byte	8-bit signed two's complement integer
short	16-bit signed two's complement integer
int	32-bit signed two's complement integer
long	64-bit signed two's complement integer
float	32-bit floating-point number (IEEE 754-1985)
double	64-bit floating-point number (IEEE 754-1985)



```
boolean flag = true;
boolean verbose, debug;
char grade = 'A';
byte b = 12;
short s = 24;
int i, j, k = 257;
long l = 890L;
float pi = 3.1416F;
double e = 2.71828, a = 6.022e23;
```

Pop Quiz!

Does not count for your grade

Type Casting

- **Let’s look at some code**

CS1 Review Topics

1. Classes - accessors, constructors, this keyword, new keyword, toString, object equality

2. Arrays - initialization, default values, searching through an array

3. Command Line Arguments

4. Scanner - reading from user input and reading from a file

5. Exceptions

Exercise 1 -

Part a: Create a College class with:

name,

number of students,

year founded

Part b: In the main, create 3 colleges and put them in an array

Part c: Take a college name as input and print the year it was founded

Exercise 1 -

What is a class?

What is an object?

What is a primitive? How is it different from an object?

What are access modifiers?

Access Control Modifiers

- `public:`
  - designates that all classes may access
- `private`:
  - designates that access is granted only to code within that class.
- protected:
  - child classes may access
- `static`
  - associates a variable/method with the class as a whole, rather than with each individual instance of that class

Exercise 2 - count words in a file

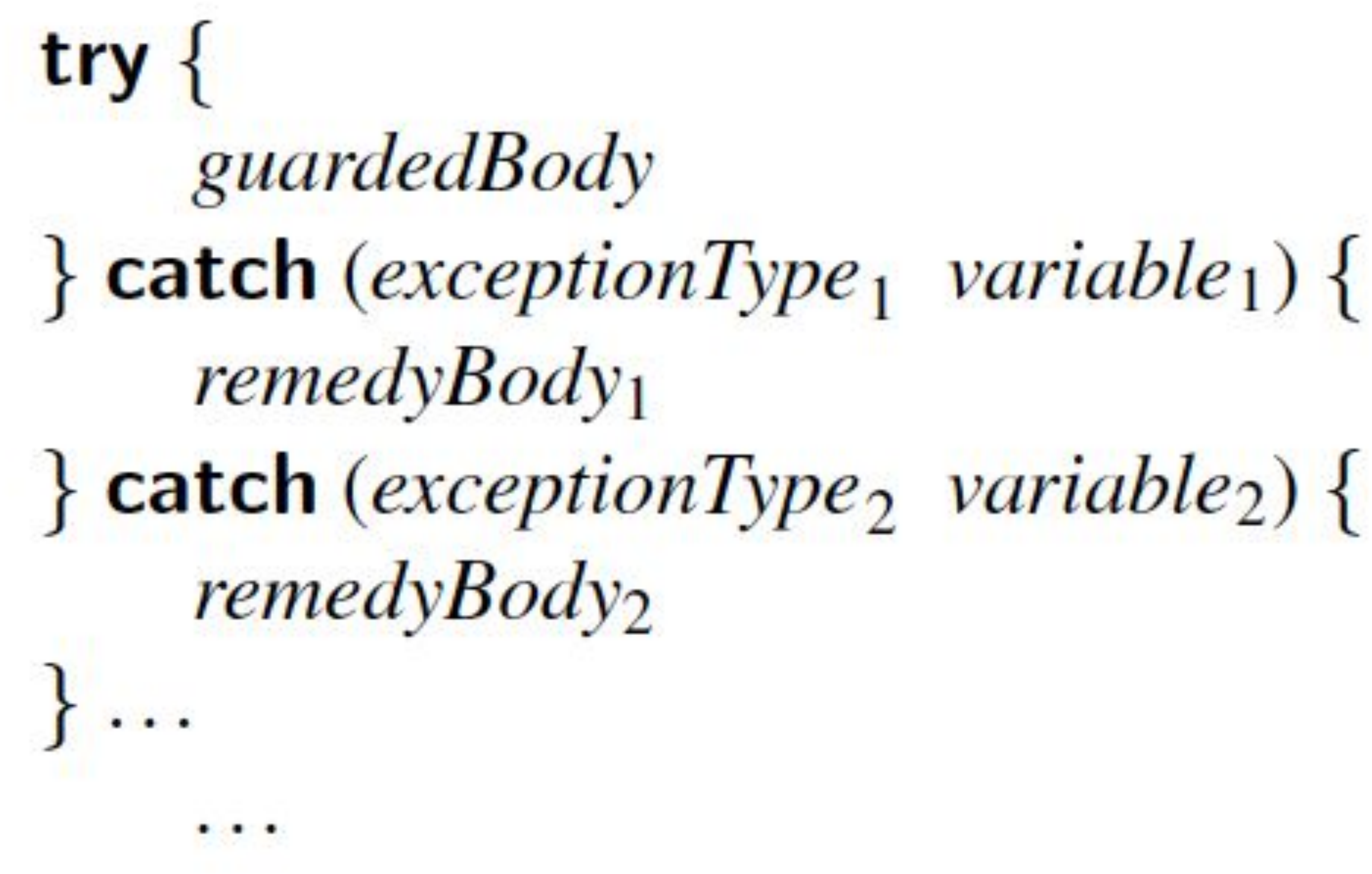
Part a: Read in a filename from command line

Part b: Count the number of words in the file

Exceptions – way to deal with unexpected events during execution

- Unexpected events:
  - unavailable resource
  - unexpected input
  - NPE
  - AOB

How do we deal with exceptions?



```
try {
    guardedBody
} catch (exceptionType$_1$  variable$_1$) {
    remedyBody$_1$
} catch (exceptionType$_2$  variable$_2$) {
    remedyBody$_2$
} ...
    ...
```

Summary

- Lab 1 today
  - Not autograded
  - Due Monday (Jan 26)

- HW0 Released
  - Due Monday (Jan 26)

- Join Piazza and Gradescope