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

Administrivia

- Course website
  - BMC-CS-151.github.io
  - Assignments and lab instructions, syllabus

- Piazza:
  - Asynchronous communication
  - Can post anonymously (anonymous just to classmates)
  - Answer your peers questions!
    - Counts for participation grade

- Gradescope:
  - Entry code **GPBX65**
  - 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

- Homeworks <u>due on Thursdays</u> released on Wednesdays
  - 10 points deducted each day. After two days, the submission window will be closed.
  - WHAT YOU SEE IS WHAT YOU GET

- Lab Park 231/W 2:40pm-4:00pm (After class)

- Midterm: Oct 30th

- Final Exam: self scheduled

AI Disclaimer!

Syallbus

- Homeworks: 50%

- Labs: 5%

- Midterm: 15%

- Final: 25%

- Participation: 5%

Average Workload
(reported by last semesters 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

Course Staff

- Khahn Ha Nguyen

- Renata Del Vecchio

- Keziah Keya

- Glory Zhang

- Kripa Lamichhane

Course Staff

- Office hours Park 231:
  - Times TBD

- Poll on Piazza

Dr. Elizabeth Dinella

- 1st year at BMC
- Recent Penn Grad: (PhD thesis neural inference of program specifications)
- Office Hours: TBD
- Research:
  - Program Analysis
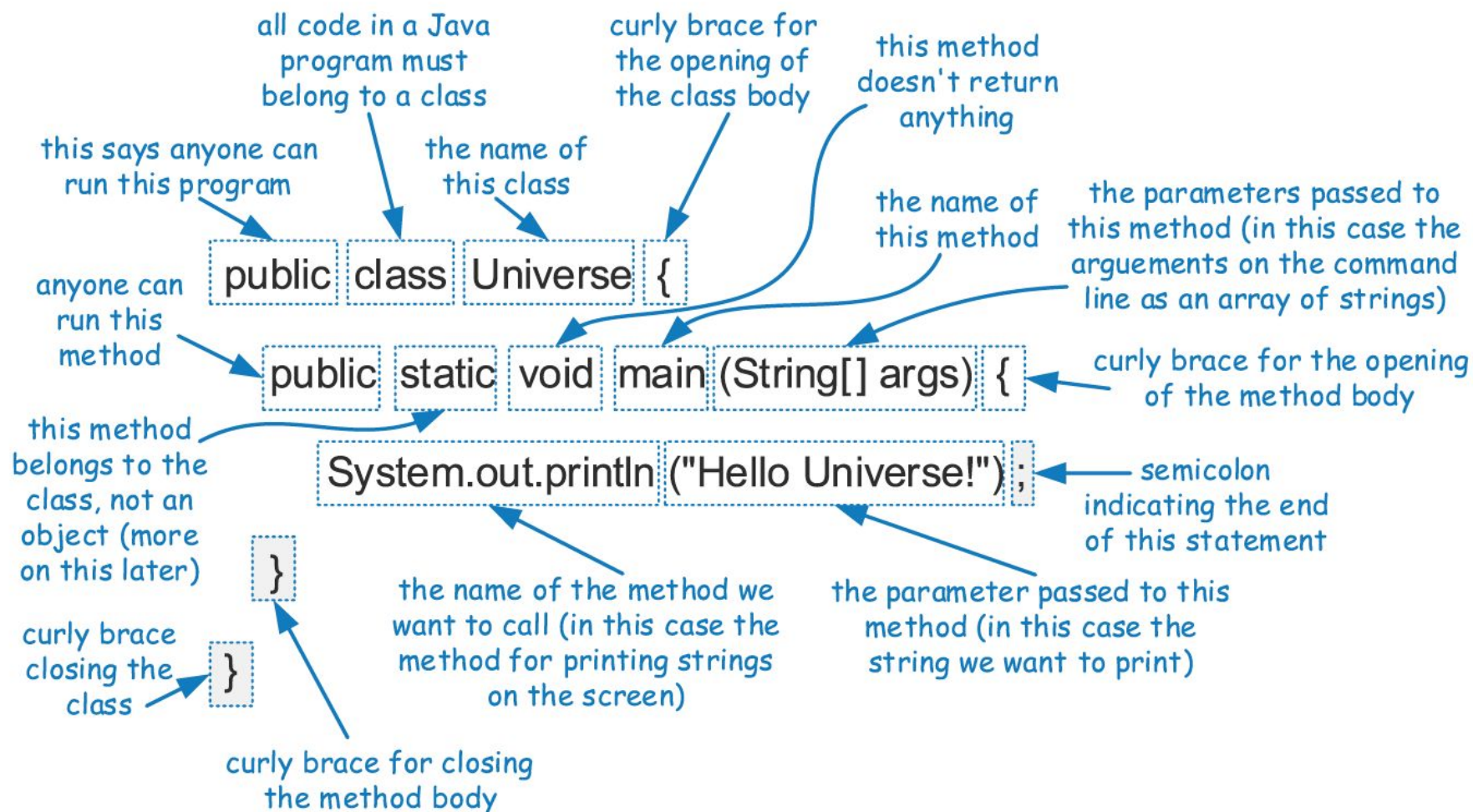 - Machine Learning
  - Web3 Security

First Things

- 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

- Lab attendance is required.

- Software: vim, Java, or just ssh

Outline

- Data Types
- Objects
- String review
- Input (Scanner)
- OOP (Inheritance)
- File I/O, Exceptions
- ***Not reviewing:***
  - Methods
  - Loops

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)

```
public  class  Universe  {
```

anyone can run this method

```
public  static  void  main (String[] args)  {
```

curly brace for the opening of the method body

this method belongs to the class, not an object (more on this later)

```
System.out.println ("Hello Universe!") ;
```

semicolon indicating the end of this statement

}

curly brace closing the class

}

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 for closing the method body

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;
```

Classes and Objects

- *Classes* are blueprints, *objects* are instance of the classes

- A class defines:
  - instance variables – what the object stores
  - Methods – how the object functions

- Every variable is either a primitive or a reference to an object

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 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

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

javadoc comments

- Comments
  - /* */
  - //

- A style/format of commenting for auto-generation of documentation in html
  /**

   */

  - used for method headers and classes

Example

```
/**
 * returns the sum of two integers
 * @param x The first integer
 * @param y The second integer
 * @return int The sum of x+y
 */
int sum(int x, int y)
```

Casting – convert the type

- **More coding :)**

Equality - More coding :)

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 (exceptionType₁  variable₁) {
    remedyBody₁
} catch (exceptionType₂  variable₂) {
    remedyBody₂
} …

    …
```

What you should know/review

- variables
- expressions
- operators
- methods
  - parameters
  - return value
- conditionals
- `for`/`while` loops

- class design and object construction
  - instance variables
  - constructor
  - getters/setters
  - class methods
  - `new`
- arrays
- arrays of objects
- `String`

What you don't know

- Read the manuals/references
  - Unix commands (flags, usage, examples)
  - Java methods (parameters/overloading)
- Google – but with judgement
- **AI Disclaimer**
- Trial-and-Error is a fundamental method of problem-solving
- The ability to tinker is a fundamental engineering/CS skill