1. McGraw
2. Review
3. Debugging
4. Assignments
Questions

raise your hand and ask

ask on Ed
(use ❤ to upvote)
The McGraw Center

Your One-Stop-Shop for Finding Academic Support, Building Study Skills, and Learning to THRIVE at Princeton
What We Do

• Princeton is HARD – and we’re here to help!
• McGraw provides resources separate from your courses to help you reach your own goals, meet the expectations of each course, and conquer each challenge that may come your way.
How We Help...

Academic Life and Learning Strategies Consultations

Tutoring Program

Academic Strategies Workshops

Digital Learning Lab - DLL
How To Find Us

• Go to the McGraw Website (https://mcgraw.princeton.edu/undergraduates)
• Or Come Find Us in Frist 328

Note: Sessions (specifically for individual consultations) fill up quickly, especially right before a p-set is due or an exam is scheduled. Don’t wait to participate - use all that Princeton has to offer to make your academic experience that much more successful!
Review: Concepts To Date

✴ IntelliJ:
  ✓ Project zip (download / open folder)
  ✓ Edit
  ✓ compile (javac HelloWorld.java)
  ✓ execute (java HelloWorld)

✴ TigerFile
  ✓ Test cases
  ✓ Checkstyle, etc.

✴ Basic structure of Java program

```java
public class HelloWorld {
    // First program - prints Hello, World
    public static void main(String[] args) {
        System.out.println("Hello, World");
    }
}
```

✴ Data types
  ✓ int, double, boolean, char, String
  ✓ Literals
  ✓ Operators
  ✓ Comparisons

✴ Type conversions
  ✓ Integer.parseInt(), Double.parseDouble()
  ✓ Promotion or automatic
  ✓ Cast

✴ jshell

✴ Conditionals (if / else if / else)

✴ Loops (for / while)

✴ Nesting
  ✓ Nested conditionals
  ✓ Nested loops
  ✓ Nested conditionals and loops, etc.

✴ Command-line arguments, System.out.print,
  System.out.println, + (Strings vs integers)
Review: COS126 Concept Understanding Tips

- Unsure about a concept: redo precept exercises. See Ed Lessons with solutions. Also, additional exercises.

- Write "throwaway programs" to understand a concept:

```java
public class Scratch {
    public static void main(String[] args) {
        int n = Integer.parseInt(args[0]);
        int d = Integer.parseInt(args[1]);
        int r = n / d;
        System.out.println(n + " / " + d + " = " + r);
    }
}
```
• Reuse and adapt code from the course materials: lectures, precepts, book, booksite, etc. (Cite your source.)

• Use pencil and paper

• Work “incrementally” - for example, HiFour.java
  1. Print a command line argument
  2. Print two command line arguments
  3. Print four command line arguments
  4. Now reverse them

• Use print statements!

• Ask for help!
Debugging is like being the detective in a crime movie where you’re also the murderer.

- Filipe Fortes -
Debugging. Identifying and fixing errors in computer code.

- A compile-time error is an error that prevents the program from compiling.
- A run-time exception is an error that causes the program to abort prematurely.
- A logic error is an error that causes the program to produce an incorrect result.

<table>
<thead>
<tr>
<th></th>
<th>valid program?</th>
<th>completes execution?</th>
<th>correct output?</th>
<th>example</th>
</tr>
</thead>
<tbody>
<tr>
<td>compile-time error</td>
<td>×</td>
<td>—</td>
<td>—</td>
<td>variable used before it is declared</td>
</tr>
<tr>
<td>run-time exception</td>
<td>✓</td>
<td>×</td>
<td>—</td>
<td>integer division by zero</td>
</tr>
<tr>
<td>logic error</td>
<td>✓</td>
<td>✓</td>
<td>×</td>
<td>integer division (when floating-point division needed)</td>
</tr>
<tr>
<td>goal</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>Programming Exam 1 (we hope!)</td>
</tr>
</tbody>
</table>
DEBUGGING

• "live" coding
• let's go on a bug hunt!
Debugging a complete program

Goal. Write a program that reads a command-line argument $n$ and prints $n! = 1 \times 2 \times 3 \times \ldots \times n$. 

<table>
<thead>
<tr>
<th>$n$</th>
<th>$n!$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>24</td>
</tr>
<tr>
<td>5</td>
<td>120</td>
</tr>
<tr>
<td>6</td>
<td>720</td>
</tr>
</tbody>
</table>

factorial function
**Goal.** Develop a program, fixing compile-time, run-time, and logic errors.
Goal. Write a program that reads a command-line argument \( n \) and prints \( n! = 1 \times 2 \times 3 \times \ldots \times n \).

```java
public class Factorial {
    public static void main(String[] args) {
        int n = Integer.parseInt(args[0]);
        for (int i = 0; i < n; i++)
            product = product * i;
    }
    System.out.println(product);
}
```

```bash
~/Desktop/debugging> javac Factorial.java
error: <identifier> expected
...
Bug 1

Goal. Write a program that reads a command-line argument \( n \) and prints \( n! = 1 \times 2 \times 3 \times \ldots \times n \).

```java
public class Factorial {
    public static void main(String[] args) {
        int n = Integer.parseInt(args[0]);
        for (int i = 0; i < n; i++)
            product = product * i;
        System.out.println(product);
    }
}
```

~/Desktop/debugging> javac Factorial.java
error: cannot find symbol
    product = product * i;
         ^
symbol:   variable product
...
Bug 2

Goal. Write a program that reads a command-line argument \( n \) and prints \( n! = 1 \times 2 \times 3 \times \ldots \times n. \)

```java
public class Factorial {
    public static void main(String[] args) {
        int n = Integer.parseInt(args[0]);
        int product;
        for (int i = 0; i < n; i++)
            product = product * i;
        System.out.println(product);
    }
}
```

~/Desktop/debugging> javac Factorial.java
error: variable product might not have been initialized
    product = product * i;
   ^

...
Bug 3

**Goal.** Write a program that reads a command-line argument $n$ and prints $n! = 1 \times 2 \times 3 \times \ldots \times n$.

```java
public class Factorial {
    public static void main(String[] args) {
        int n = Integer.parseInt(args[0]);
        int product = 0;
        for (int i = 0; i < n; i++)
            product = product * i;
        System.out.println(product);
    }
}
```

~/Desktop/debugging> javac Factorial.java
error: cannot find symbol
           product = product * i;
              ^
symbol:   variable i
1 error
Goal. Write a program that reads a command-line argument $n$ and prints $n! = 1 \times 2 \times 3 \times \ldots \times n$.

```java
public class Factorial {
    public static void main(String[] args) {
        int n = Integer.parseInt(args[0]);
        int product;
        for (int i = 0; i < n; i++) {
            product = 0;
            product = product * i;
        }
        System.out.println(product);
    }
}
```

~/Desktop/debugging> javac Factorial.java
error: variable product might not have been initialized
    System.out.println(product);
    ^
Bug 5

Goal. Write a program that reads a command-line argument \( n \) and prints \( n! = 1 \times 2 \times 3 \times \ldots \times n \).

```java
public class Factorial {
    public static void main(String[] args) {
        int n = Integer.parseInt(args[0]);
        int product = 0;
        for (int i = 0; i < n; i++) {
            product = product * i;
        }
        System.out.println(product);
    }
}
```

`~/Desktop/debugging> javac Factorial.java
~/Desktop/debugging> java Factorial
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 0
    at Factorial.main(Factorial.java:3)
Bug 6

Goal. Write a program that reads a command-line argument \( n \) and prints \( n! = 1 \times 2 \times 3 \times \ldots \times n \).

```java
public class Factorial {
    public static void main(String[] args) {
        int n = Integer.parseInt(args[0]);
        int product = 0;
        for (int i = 0; i < n; i++) {
            product = product * i;
        }
        System.out.println(product);
    }
}
```

~$/Desktop/debugging$> javac Factorial.java
~$/Desktop/debugging$> java Factorial 3
0
~$/Desktop/debugging$> java Factorial 5
0
**Goal.** Write a program that reads a command-line argument \( n \) and prints \( n! = 1 \times 2 \times 3 \times \ldots \times n \).

```java
public class Factorial {
    public static void main(String[] args) {
        int n = Integer.parseInt(args[0]);
        int product = 1;
        for (int i = 0; i < n; i++) {
            product = product * i;
        }
        System.out.println(product);
    }
}
```

```bash
~/Desktop/debugging> javac Factorial.java
~/Desktop/debugging> java Factorial 3
0
~/Desktop/debugging> java Factorial 5
0
```
**Goal.** Write a program that reads a command-line argument $n$ and prints $n! = 1 \times 2 \times 3 \times \ldots \times n$.

```java
public class Factorial {
    public static void main(String[] args) {
        int n = Integer.parseInt(args[0]);
        int product = 1;
        for (int i = 1; i < n; i++) {
            product = product * i;
        }
        System.out.println(product);
    }
}
```

```bash
~/Desktop/debugging> javac Factorial.java
~/Desktop/debugging> java Factorial 3
2
~/Desktop/debugging> java Factorial 5
24
```
Goal. Write a program that reads a command-line argument $n$ and prints $n! = 1 \times 2 \times 3 \times \ldots \times n$. 

```java
public class Factorial {
    public static void main(String[] args) {
        int n = Integer.parseInt(args[0]);
        int product = 1;
        for (int i = 1; i < n; i++) {
            product = product * i;
            System.out.println("i = "+i+", product = "+product);
        }
        System.out.println(product);
    }
}
```

~/Desktop/debugging> javac Factorial.java
~/Desktop/debugging> java Factorial 4
  i = 1, product = 1
  i = 2, product = 2
  i = 3, product = 6
System.out.println() debugging statement
System.out.println(...);

* Remove before submitting to TigerFile
Goal. Write a program that reads a command-line argument \( n \) and prints \( n! = 1 \times 2 \times 3 \times \ldots \times n \).

```java
public class Factorial {
    public static void main(String[] args) {
        int n = Integer.parseInt(args[0]);
        int product = 1;
        for (int i = 1; i <= n; i++) {
            product = product * i;
        }
        System.out.println(product);
    }
}
```

```bash
~/Desktop/debugging> javac Factorial.java
~/Desktop/debugging> java Factorial 3
6
~/Desktop/debugging> java Factorial 5
120
```
DEBUGGING

- tracing
- live coding
- let’s go on a bug hunt!
What is the value of the variable \( x \) at the end of the computation?

A. 0.0
B. 0.59999999999999997779553950749686919152736663818359375
C. 0.6
D. compile-time error
E. run-time exception

```java
int numerator = 3;
int denominator = 5;
double x = (double) (numerator / denominator);
```
What is the value of `max` at the end of the computation?

```java
int a = 2;
int b = 4;
int c = -4;
int max = Math.max(a, b, c);
```

A. -4
B. 2
C. 4
D. `compile-time error`
E. `run-time exception`
What does the following code fragment print?

```java
double humidity = 40.0;
if (40 <= humidity <= 60.0)
    System.out.println("comfortable");
```

A. "comfortable"
B. 40.0
C. nothing
D. compile-time error
E. run-time exception
What does the following code fragment print?

```java
int x = 0;
if (x == 0) System.out.println("zero");
```

A. "zero"
B. "not zero"
C. nothing
D. compile-time error
E. run-time exception
What does the following code fragment print?

```java
int x = -123;
boolean isPositive = (x > 0);
if (isPositive = true) System.out.println("positive");
else System.out.println("not positive");
```

A. "positive"
B. "not positive"
C. nothing
D. compile-time error
E. run-time exception
What is the value of `result` at the end of the computation?

```java
int l = 0;
int O = 1;
int l00 = 2;
int result = l00 + 100 + 0*O + 1*1 + 1001;
```

A. 1024  
B. 1104  
C. 1201  
D. 1202  
E. compile-time error
What does the following code fragment print?

```java
int default = 123;
int x = 456;
boolean resetToDefault = true;
if (resetToDefault)
    x = default;
```

A. 123
B. 456
C. nothing
D. compile-time error
E. run-time exception
What does the following code fragment print?

```java
int x = -123;
if (x > 0) {
    System.out.println("positive");
}
```

A. "positive"
B. "not positive"
C. nothing
D. compile-time error
E. run-time exception
What does the following code fragment print?

```java
int x = 99;
while (x > 0) {
    x = x / 2;
}
```

A. 0
B. 1
C. 99
D. *nothing*
E. *compile-time error*
What is the value of the variable force at the end of the computation?

\[ F = \frac{G m_1 m_2}{r^2} \]

A. 0.0
B. 0.04
C. 0.08
D. 2.0
E. NaN
Which is the value of the variable $x$ at the end of the computation?

```java
double x = 180.0;
Math.toRadians(x);
```

A. 0.0
B. 1.5707963267948966
C. 3.141592653589793
D. 180.0
E. 10313.2
A legendary bug

Admiral Grace Hopper's operational logbook for the Harvard Mark II computer
“Everyone knows that debugging is twice as hard as writing a program in the first place. So if you’re as clever as you can be when you write it, how will you ever debug it?” — Brian Kernighan
“If debugging is the process of removing bugs, then programming must be the process of putting them in.” — Edsger Dijkstra
Assignments: Assignment 1 | Loops

- Review your Assignment 0 Hello, World feedback

- Reuse and adapt code from the course materials: lectures, precepts, book, booksite, etc. (Cite your source.)

- Work “incrementally” - For example - Checkerboard:

  1. Print the command-line argument and the String "* 

  2. Instead of a space, perhaps use another character, such as ~

  3. Print one row of N "*~"

  4. Print N rows of #3

  5. Change #4 to start even lines with "*~" / odd lines with "~*"

  6. Replace the ~ with a space
• We plan to post your graded assignment Friday morning

• We use a web-based platform called CodePost

• You will receive a notification (check email and Ed!)

• **Review your feedback for next assignment**

• Note that items that are a -0 warning this week, may be an actual deduction on the next assignments.

• And items that are a small -1 or -2 penalty this week, may be a more substantive deduction on the next assignments.

• If you do not understand the feedback, ask for clarification. (Office hours, or your preceptor)

• 72 hours after release for regrade requests - use the CodePost regrade request feature (see Ed)
Questions

raise your hand and ask

ask on Ed
(use ❤ to upvote)