Student one;
if(one!=null & one.getYear() == 2){
    System.out.println(one.getName());
}

• Which of the following results from the code?
  A. Compiler error
  B. Run-time error
  C. There’s no error, but nothing is printed.
  D. A name is printed.
  E. Whatever
Administrative Details

• Lab 1 handout is now online

• Prelab (=should be completed before lab):
  • Set up accounts
  • Complete Lab 1 design doc
    • Take a look at the example
Crash Course in Java

- Variables
  - "content word" school nice ...
  - "function word" is
  - "phrase" very nice
  - "sentence"

- Operators

- Expressions

- Statements

\[ x = x + 3 \]
Variable Types

• **Primitive Types:**
  - boolean, char, byte, short, int, long, float, double

• **Objects:**
  - extend Object
    - arrays String[] args
      - Holds values of a single type
    - (class-based) Objects
      - Can hold information (fields)
      - Can specify behaviors (methods)
(General) Operators

- **Unary**: 1 argument
  - Arithmetic: +, -, ++, -- (prefix and postfix)
  - Logical: !

- **Binary**: 2 args
  - Arithmetic: +, -, *, /, %
  - Relational: ==, !=, <, <=, >, >=
  - Logical: &&, ||
  - Assignment: =, +=, -=, *=, /=, %=  

- **Ternary**: 3 args
  - booleanCondition ? value1 : value2
## Operator Precedence in Java

<table>
<thead>
<tr>
<th>Operators</th>
<th>Precedence</th>
</tr>
</thead>
<tbody>
<tr>
<td>postfix</td>
<td><code>expr++</code> <code>expr--</code></td>
</tr>
<tr>
<td>unary</td>
<td><code>++expr</code> <code>--expr</code> <code>+expr</code> <code>-expr</code> <code>~</code> <code>!</code></td>
</tr>
<tr>
<td>multiplicative</td>
<td><code>*</code> <code>/</code> <code>%</code></td>
</tr>
<tr>
<td>additive</td>
<td><code>+</code> <code>-</code></td>
</tr>
<tr>
<td>shift</td>
<td><code>&lt;&lt;</code> <code>&gt;&gt;</code> <code>&gt;&gt;&gt;</code></td>
</tr>
<tr>
<td>relational</td>
<td><code>&lt;</code> <code>&gt;</code> <code>&lt;=</code> <code>&gt;=</code> <code>instanceof</code></td>
</tr>
<tr>
<td>equality</td>
<td><code>==</code> <code>!=</code></td>
</tr>
<tr>
<td>bitwise AND</td>
<td><code>&amp;</code></td>
</tr>
<tr>
<td>bitwise exclusive OR</td>
<td><code>^</code></td>
</tr>
<tr>
<td>bitwise inclusive OR</td>
<td>`</td>
</tr>
<tr>
<td>logical AND</td>
<td><code>&amp;&amp;</code></td>
</tr>
<tr>
<td>logical OR</td>
<td>`</td>
</tr>
<tr>
<td>ternary</td>
<td><code>?</code> <code>:</code></td>
</tr>
<tr>
<td>assignment</td>
<td><code>=</code> <code>+=</code> <code>-=</code> <code>*=</code> <code>=/=</code> <code>%=</code> <code>&amp;=</code> <code>^=</code> `</td>
</tr>
</tbody>
</table>
Crash Course in Java

• Variables
• Operators
○ Expressions
• Statements
Expressions

Sequence of variables, literals, method calls & operators that evaluate to a value

- An expression returns a value
  - 3+2*5
  - “Your score is ” + 100
  - i <= students.length
  - x++
  - ++x
  - Note, an assignment expression also returns a value
    - y = 4 * (x = 3)
    - while ((line = reader.readLine()) != null){
      //do something with the line
    }
[TAP] Pre- and Post-increment

\[ X = 10; \]
\[ \text{System.out.println}((X++) \times (++X)); \]
\[ X = 10; \]
\[ \text{System.out.println}((++X) \times (X++)); \]

• Which of the following are outputted?
  
  A. 100 & 100
  B. 110 & 110
  C. 120 & 121
  D. None of the above
  E. Whatever

11
Crash Course in Java

- Variables
- Operators
- Expressions
- Statements
Statements

- Statements
  - int x;
  - x = 3;
  - System.out.println("Hello, CS136!");
  - if (x > 3) { ... } else { ... }
  - while (x < 2) { ... }
  - for (int i = 0; i < x; i++) { ... }
Control Flow Statements

Select next statement based on a boolean expression.

• Branching structures: if, if/else, switch
• Looping structures: while, do-while, for
If/else

Example: Encode clubs, diamonds, hearts, spades as 0, 1, 2, 3

```java
if (x == 0 || x == 2){
    System.out.println("Your card is red");
} else if (x == 1 || x == 3){
    System.out.println("Your card is black");
} else {
    System.out.println("Illegal suit code!");
}
```
Example: Encode clubs, diamonds, hearts, spades as 0, 1, 2, 3

```
switch (x) {
    case 0:
    case 2:
        System.out.println("Your card is red");
        break;
    case 1:
    case 3:
        System.out.println("Your card is black");
        break;
    default:
        System.out.println("Illegal suit code!");
        break;
}
```
while & do-while

Example: Count # of flips until “heads”

Random rng = new Random();
int flip, count = 0;
flip = rng.nextInt(2); // returns 0 or 1
count++;
while (flip == 0) {
    flip = rng.nextInt(2);
    count++;
}

VS

do {
    flip = rng.nextInt(2);
    count++;
} while (flip == 0);
Example: Compute the average of test scores

```java
int[] grades = { 100, 78, 92, 87, 89, 90 };
int sum = 0;
int i = 0;
while ( i < grades.length ) {
    sum += grades[i];
    i++;
}
ave = sum / grades.length;
```

VS

```java
for( int i = 0; i < grades.length; i++ )
    sum += grades[i];
```

VS

```java
for ( int g : grades )
    sum += g;
```
break & continue

break (Print the first prime between 100 and 200)
for ( int i = 100; i <= 200 ; i++ ){
    if ( isPrime(i) ) {
        System.out.println( i );
        break;
    }
}

continue (Print all even # between 100 and 200)
for ( int i = 100; i <= 200 ; i++ ) {
    if (i%2 = 1){
        continue;
    }
    System.out.println( i );
}
Crash Course in Java

- Variables
- Operators
- Expressions
- Statements
- Object-oriented Programming
Object-Oriented Programming

- OOP is a programming paradigm, where a program is a set of objects interacting with one another.

Variable Types

- Primitive Types: `true, false, letter, boolean, char, byte, short, int, long, float, double`
- Objects: `extend Object`
  - `arrays`: `String[] args`
    - Holds values of a single type
  - (class-based) Objects
    - Can hold information (fields)
    - Can specify behaviors (methods)
OOP Example

Goal: Keep track of babies at a nursery; for each baby, keep this info: name and age.

```java
Non-OOP

String[] names
int[] age

OOP

Baby
  name
  age
```
public class Baby {
    private String name;
    private int age;

    public Baby(int theAge, String theName) {
        age = theAge;
        name = theName;
    }

    public String getName() { return name; }
    public int getAge() { return age; }

    public void setName(String newName) {
        name = newName;
    }

    public void setAge(int newAge) {
        if (newAge > 0)
            age = newAge;
    }
}

private static String nursery;

public static Baby (18, "Ben")

int u = a.getAge();

name

getName()
# Access Modifiers

<table>
<thead>
<tr>
<th></th>
<th>Same Class</th>
<th>Class in the Same Package</th>
<th>Any Subclass</th>
<th>Any Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>public</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>protected</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>None (package)</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>private</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
</tbody>
</table>

Again, be as restrictive as possible!
Revisiting Hello.java

```java
public class Hello {
    public static void main(String[] args) {
        System.out.println("Hello, CS136!");
    }
}
```