CSE 121 – Lesson 3

Miya Natsuhara
Autumn 2023

Music: 121 23au Lecture Tunes 🌧️

TAs: Trey Christina Sahej Vinay Kriti
    Sebastian Colton Anju Maria Minh
    Annie Janvi Jonus Shreya Vivian
    Jasmine Arkita Lydia Andy Nicole
    Christian Vidhi Luke Nicolas Simon
    Lucas Ritesh Andras Shayna Jessie
    Logan Hibbah Archit Hannah Lydia
    Jacob Julia Ayesha Aishah Yijia
Announcements, Reminders

• C0 was due on Wednesday
• P0 was released on Wednesday and is due Tuesday, October 10
  • This will be our typical schedule for assignments!
• Quiz 0 scheduled for October 19 (about 2 weeks away)
  • More details about quizzes will be released in the coming week
Now that we know about different types and data, we can learn about how to store it!

Java allows you to create variables within a program. A variable has:

- A type
- A name
- (Potentially) a value it is storing

Declaration: \( \text{int } x; \)

Initialization: \( x = 30; \)

Or all in one line:

\( \text{int } x = 30; \)
Variables

They’re made to be manipulated, modified,

```java
int myFavoriteNumber = 7;
int doubleFV = myFavoriteNumber * 2;
myFavoriteNumber = myFavoriteNumber + 3;
```

Notice – this doesn't really make any mathematical sense! That's because, in Java, `=` is *assignment*, not equality!
New Operators!

myFavoriteNumber = myFavoriteNumber + 3;
This type of pattern is so common, we have an even shorter way we can write it!

myFavoriteNumber += 3;

You can do the same for -=, *=-, /=, and %=:

And there are even shorter versions for *incrementing* and *decrementing*!

myFavoriteNumber++;  myFavoriteNumber--;
What do a, b, and c hold after this code is executed?

```c
int a = 10;
int b = 30;
int c = a + b;
c -= 10;
a = b + 5;
b /= 2;
```

A. 10, 30, 40  
B. 35, 15, 30  
C. 35, 15.5, 30  
D. 20, 15, 30
Strings and chars

- String = sequence of characters treated as one, yet can be indexed to get individual parts
- Zero-based indexing

- **Side note**: new data type! `char`, represents a single character, so we use single quotes
  Strings are made up of chars!
### (PCM) String Methods

**Usage:** `<string variable>.<method>(...)`

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>length()</td>
<td>Returns the length of the string.</td>
</tr>
<tr>
<td>charAt(i)</td>
<td>Returns the character at index <code>i</code> of the string.</td>
</tr>
<tr>
<td>indexOf(s)</td>
<td>Returns the index of the first occurrence of <code>s</code> in the string; returns <code>-1</code> if <code>s</code> doesn't appear in the string</td>
</tr>
<tr>
<td>substring(i, j) or substring(i)</td>
<td>Returns the characters in this string from <code>i</code> (inclusive) to <code>j</code> (exclusive); if <code>j</code> is omitted, goes until the end of the string</td>
</tr>
<tr>
<td>contains(s)</td>
<td>Returns whether or not the string contains <code>s</code></td>
</tr>
<tr>
<td>equals(s)</td>
<td>Returns whether or not the string is equal to <code>s</code> (case-sensitive)</td>
</tr>
<tr>
<td>equalsIgnoreCase(s)</td>
<td>Returns whether or not the string is equal to <code>s</code> ignoring case</td>
</tr>
<tr>
<td>toUpperCase()</td>
<td>Returns an uppercase version of the string</td>
</tr>
<tr>
<td>toLowerCase()</td>
<td>Returns a lowercase version of the string</td>
</tr>
</tbody>
</table>
Suppose s contains the String "bubble gum". Which option below would result in s containing "Gumball" instead?

A. `s.substring(7) + "ball";`
B. `s = s.substring(7, 9) + "ball";`
C. `s = s.charAt(7).toUpperCase() + "ball";`
D. `s =
   s.substring(7, 8).toUpperCase()
   + s.substring(8) + "ball";`
E. `s =
   s.substring(7, 8).toUpperCase()
   + s.substring(7, 10) + "ball";`