CSE 121 – Lesson 3

Miya Natsuhara
Spring 2023

Music: 121 23sp Lecture Vibes 🌸

TAs: Jasmine Atharva Mia Justin
Shananda Julia Archit Aishah
Vidhi Anju Grace Claire
Larry Lydia Kailye Lydia
Jacqueline Jonus Joshua Kai
Afifah Hugh James

sli.do #cse121
Announcements, Reminders

• C0 was due last night
• P0 was released on Wednesday and is due Tuesday, April 11
  • This will be our typical schedule for assignments!
• Quiz 0 scheduled for April 20 (about 2 weeks away)
  • More details about quizzes will be released in the coming week
(PCM) Variables

• 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:       int x;
Initialization:    x = 30;

Or all in one line:

    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
(PCM) 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 -1 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?

\[
\begin{align*}
\text{A.} & \quad s\text{.substring}(7) + \text{"ball";} \\
\text{B.} & \quad s = s\text{.substring}(7, 9) + \text{"ball";} \\
\text{C.} & \quad s = s\text{.charAt}(7).\text{toUpperCase()} + \text{"ball";} \\
\text{D.} & \quad s = s\text{.substring}(7, 8).\text{toUpperCase()} + s\text{.substring}(8) + \text{"ball";} \\
\text{E.} & \quad s = s\text{.substring}(7, 8).\text{toUpperCase()} + s\text{.substring}(7, 10) + \text{"ball";} \\
\end{align*}
\]