CSE 121 Lesson 5:
Nested for loops, Math, Random

Elba Garza & Matt Wang
Winter 2024

TAs: Abby Aishah Anju Annie Archit Ayesha Christian
Hannah Heather Hibbah Jacob James Janvi Jasmine
Jonus Julia Lucas Luke Maria Nicole Shananda
Shayna Trey Vidhi Vivian

sli.do #CSE121-5

Today’s playlist:
CSE 121 24w lecture beats :D
Announcements, Reminders

• Creative Project 1 is out, due Tues Jan 23rd
• Resubmission Cycle 0 released yesterday, due Thurs Jan 25th
  • Eligible for submission: C0 & P0
  • Even if you're not resubmitting – read your feedback!!
• reminder: no code screenshots (accessibility!)
Last time: for loops!

For loops are our first control structure
A syntactic structure that controls the execution of other statements.

for ( initialization ; test ; update ) {
  body (statements to be repeated)
}


Fencepost Pattern

Some task where one piece is repeated $n$ times, and another piece is repeated $n-1$ times and they alternate

h-u-s-k-i-e-s
Fencepost Pattern

Some task where one piece is repeated $n$ times, and another piece is repeated $n-1$ times and they alternate.
(PCM) Nested for loops

```java
for (int outerLoop = 1; outerLoop <= 5; outerLoop++) {
    System.out.println("outer loop iteration "+outerLoop);
    for (int innerLoop = 1; innerLoop <= 3; innerLoop++) {
        System.out.println(" inner loop iteration "+innerLoop);
    }
    System.out.println(outerLoop);
}
```
What output is produced by the following code?

```java
for (int i = 1; i <= 5; i++) {
    for (int j = 1; j <= i; j++) {
        System.out.print(i);
    }
    System.out.println();
}
```

A.  
1  
12  
123  
1234  
12345  

B.  
1  
22  
333  
4444  
55555  

C.  
1  
22  
333  
4444  
55555  

Poll in with your answer!
Poll in with your answer!

What code produces the following output?

A. 
```java
for (int i = 1; i <= 5; i++) {
    for (int j = 1; j <= i; j++) {
        System.out.print(i);
    }
    System.out.println();
}
```

B. 
```java
for (int i = 1; i <= 5; i++) {
    for (int j = 1; j <= i; j++) {
        System.out.print(j);
    }
    System.out.println();
}
```

C. 
```java
for (int i = 1; i <= 5; i++) {
    for (int j = 1; i <= j; j++) {
        System.out.print(j);
    }
    System.out.println();
}
```

D. 
```java
for (int i = 1; i <= 5; i++) {
    for (int j = 1; j <= i; i++) {
        System.out.print(j);
    }
    System.out.println();
}
```
Pseudo-Randomness

Computers generate numbers in a predictable way using mathematical formulas.

Input may include current time, mouse position, etc.

True randomness is hard to achieve – we rely on natural processes
  • e.g., atmospheric noise, lava lamps
**Random**

A Random object generates *pseudo*-random numbers.

- The Random class is found in the `java.util` package
  ```java
  import java.util.*;
  ```
- We can “seed” the generator to make it behave deterministically (helpful for testing!)

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>nextInt()</code></td>
<td>Returns a random integer</td>
</tr>
<tr>
<td><code>nextInt(max)</code></td>
<td>Returns a random integer in the range <code>[0, max)</code>, or in other words, 0 to <code>max-1</code> inclusive</td>
</tr>
<tr>
<td><code>nextDouble()</code></td>
<td>Returns a random real number in the range <code>[0.0, 1.0)</code></td>
</tr>
</tbody>
</table>
Poll in with your answer!

Assuming you’ve declared: \( \text{Random randy} = \text{new Random();} \)

Which of these best models picking a random card? (1-13 inclusive)

A. `randy.nextInt()`  
B. `randy.nextInt(13)`  
C. `randy.nextInt(13) + 1`  
D. `randy.nextInt(14)`
### (PCM) Math

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Math.abs(value)</td>
<td>Returns the absolute value of value</td>
</tr>
<tr>
<td>Math.ceil(value)</td>
<td>Returns value rounded up</td>
</tr>
<tr>
<td>Math.floor(value)</td>
<td>Returns value rounded down</td>
</tr>
<tr>
<td>Math.max(value1, value2)</td>
<td>Returns the larger of the two values</td>
</tr>
<tr>
<td>Math.min(value1, value2)</td>
<td>Returns the smaller of the two values</td>
</tr>
<tr>
<td>Math.round(value)</td>
<td>Returns value rounded to the nearest whole number</td>
</tr>
<tr>
<td>Math.sqrt(value)</td>
<td>Returns the square root of value</td>
</tr>
<tr>
<td>Math.pow(base, exp)</td>
<td>Returns base raised to the exp power</td>
</tr>
</tbody>
</table>

Calling:

```
Math.<method>(...)
```
This week’s food for thought is:

• one of matt’s favourite areas of computer science
• less related to tech & society than the others...
• also **the most ambitious**, so don’t stress about it
  – sit back, enjoy the ride :)
Wouldn’t it be nice...

We’ve seen that some for loops go on forever:

```java
for (int i = 0; i < 10; i--) {
    System.out.println(i);
}

for (; true;) {
    //
}
```

Wouldn’t it be nice if Java (or “the compiler”) could catch this for us? I mean, the loop “obviously” never ends...
The Halting Problem (1/2)

Benedict Cumberbatch showed that it’s impossible to generally solve this problem. Regardless of:

• how good (big, fast) your computer is
• how good your algorithm is
• what people come up with the future!

Given a Java program, it is impossible to always know if it eventually stops (or loops infinitely).
The Halting Problem (2/2)

Benedict Cumberbatch—Alan Turing showed that it’s impossible to generally solve this problem. Regardless of:

• how good (big, fast) your computer is
• how good your algorithm is
• what people come up with the future!

Given a Java program, it is impossible to always know if it eventually stops (or loops infinitely).

Alan Turing at 24 (1936). He had a storied (if also very tragic and short) life.
Many, many problems are unsolvable.

I don’t mean “we currently don’t know how to solve them”. I mean, “there is no algorithm that will ever solve them”.

Here are some related “undecidable” problems:

- given a Java program, are all the types correct?
- given a polynomial equation, does it have integer solution(s)?
- given any Magic: The Gathering board, does either player have a guaranteed winning strategy?
In search of perfection (1/2)

In fact, there’s an even more concerning result: **math itself is inconsistent.** There is at least one math statement that we can’t prove true or false.
In search of perfection (2/2)

In fact, there’s an even more concerning result: math itself is inconsistent. There is at least one math statement that we can’t prove true or false.

Yet, we still:

• try avoiding infinite loops
• type-check our Java programs
• play Magic: The Gathering (?)
• try to prove things in (and do) math!
Dessert for Thought!

I argue there are two takeaways:

1. Don’t let perfection be the enemy of the good!
   - applies to you in CSE 121 and as a programmer :)
   - fundamental basis of much of computer science

2. Like thinking about these sorts of problems?
   This is also computer science!
   (not all CS is just coding...) See: CSE 311, CSE 417/431