



COMPUTER SCIENCE COMPETITION - JAVA TOPIC LIST 2017-2018

IMPORTANT NOTES: Java is the official programming language for UIL Computer Science. Contest content for 2017-2018 will conform to the Java Platform Standard Edition, Version 8. This list is intended as a guideline and is not all-inclusive. Knowledge of basic Java concepts is assumed. Visit the UIL web site at uiltexas.org/academics/computer-science for a list of Java resources and other important contest information.

Base Conversions and Arithmetic
User-Defined Classes (constructors, methods, instance variables, private vs. public, overloading, overriding, <code>final</code> local variables, <code>static final</code> class variables, <code>static</code> methods, <code>static non-final</code> variables)
Constructors and initialization of static variables, default initialization of instance variables
Concepts of inheritance, abstract classes, interfaces and polymorphism
<code>null</code> , <code>this</code> , <code>super</code> , <code>super.method(args)</code> , <code>super(args)</code> , <code>this.var</code> , <code>this.method(args)</code> , <code>this(args)</code> , <code>instanceof</code>
Conversion to supertypes and (Subtype) casts
Comparison of reference types (<code>equals()</code> , <code>==</code> , <code>!=</code> , <code>Comparable.compareTo()</code>)
Primitive types (<code>int</code> , <code>double</code> , <code>boolean</code> , <code>short</code> , <code>long</code> , <code>byte</code> , <code>char</code> , <code>float</code>), casting of primitives
Arrays, including arrays of arrays and initialization of named arrays
Arithmetic operators (<code>+</code> , <code>-</code> , <code>*</code> , <code>/</code> , <code>%</code> , <code>++</code> , <code>--</code>) and string concatenation
Using the values of <code>++</code> , <code>--</code> expressions in other expressions
Assignment operators (<code>=</code> , <code>+=</code> , <code>-=</code> , <code>*=</code> , <code>/=</code> , <code>%=</code>)
Boolean expressions and operators (<code>==</code> , <code>!=</code> , <code><</code> , <code><=</code> , <code>></code> , <code>>=</code> , <code>&&</code> , <code> </code> , <code>!</code> , <code>&</code> , <code> </code> , <code>^</code>) including short-circuit evaluation
Bitwise operators (<code><<</code> , <code>>></code> , <code>&</code> , <code>~</code> , <code> </code> , <code>^</code> , <code>>>=</code> , <code><<=</code> , <code>!=</code> , <code>&=</code> , <code> =</code> , <code>^=</code>)
Branching (<code>if</code> , <code>if/else</code> , <code>?:</code> , <code>switch</code> , <code>break</code>)
Looping (<code>while</code> , <code>for</code> , <code>enhanced for</code> , <code>return</code> , <code>do/while</code> , <code>break</code> , <code>continue</code>)
<code>System.out.print()</code> , <code>println()</code> , <code>printf()</code> , <code>%f</code> , <code>%d</code> , <code>%s</code> , escape sequences <code>\n</code> , <code>\\</code> , <code>\"</code>
Parsing (<code>String.split()</code> , <code>Integer.parseInt()</code> , <code>Double.parseDouble()</code>)
Pattern class, <code>Regex (. + * \d \D \s \S \w \W [abc] [^abc] [a-zA-Z])</code> , <code>matches</code>
Java Standard Library (<code>String</code> , <code>Integer</code> , <code>Double</code> , <code>Character</code> , <code>Math</code> , <code>Object</code> , <code>Comparable</code> , <code>Scanner</code> , <code>Random</code> , <code>Arrays</code>) See supplemental class reference list.
Generic collections (<code>Collection</code> , <code>List</code> , <code>Set</code> , <code>Map</code> , <code>Stack</code> , <code>Queue</code> , <code>PriorityQueue</code> , <code>ArrayList</code> , <code>LinkedList</code> , <code>HashSet</code> , <code>TreeSet</code> , <code>HashMap</code> , <code>TreeMap</code>) See supplemental class reference list.
<code>Arrays.sort()</code> and <code>Collections.sort()</code>
Recursion, Dynamic Programming
Stacks, Queues, Binary Trees, Linked Lists, Heaps, Hash Tables, Priority Queues, Graphs
Sorts (Selection, Insertion, Mergesort, Quicksort) and Searches (Sequential, Binary) – same canonicals as AP
Analysis of algorithms: informal comparison of running times, exact calculation of statement execution counts, Big-O notation, best case / worst case / average case time and space analysis
Digital Electronics – symbolic representation of Boolean expressions using logic gates NOT, AND, XOR, OR, NAND, NOR, NXOR
Two's complement binary representation of negative 8-bit integers – conversion both ways between base 10 and base 2
Polish notation – representation, analysis, and conversion of simple infix, prefix, and postfix expressions
Boolean simplification using generic notation ($A * B, A + B, A \oplus B, \bar{A}, \overline{A * B}, \overline{A + B}, \overline{A \oplus B}$ – using truth tables and Boolean Identities to analyze and simplify Boolean expressions (see list of identities on UIL CS website)

UIL Computer Science Topic List 2017-2018

Written Test First 15 Questions

1. **Number base concepts**, arithmetic, conversion
2. **Simple literal math expression** with mixed operations
3. **Simple output** involving print, println, printf (%d, %f, %s, \", \\, \n)
4. **String class methods**
5. **Simple Boolean logic** (AND, OR, XOR, NOT) - Java based
6. **Math class methods** (no advanced topics like trig - save that for later in the test)
7. **Simple variable expression** with mixed operations
8. **Conditionals** (if, if/else, switch - not ternary)
9. **Simple output loop**
10. **1D primitive array**, basic concepts
11. **Input concepts** – use of Scanner and File classes
12. **Accumulation loop** – summation, product accumulation, etc.
13. **Order of operations** (beyond just the math expressions - testing knowledge of the full Java spectrum of order of precedence)
14. **Java specific data type concepts**, memory size, max and min limits, wrap around, complements (~)
15. **ArrayList** – generics only

Topics for last 5 written test questions (40 question test)

(three multiple choice, two free response, one discrete answer each)

- **Boolean algebra concepts** – truth table analysis, use of Boolean identities for simplifying expressions.
- **Digital Electronics** - interpretation of symbols, sketching DE circuits from expressions
- **Polish notation** - representation, analysis, and conversion of simple infix, prefix, and postfix expressions
- **2s complement binary negative representation** (limit 8 bits)
- **Graph theory** - simple paths, cycles, analysis
- **Data structure theory** - stacks, queues, priority queues, binary trees (heaps, search, expression, generic)

Specifications for the first two programming problems to be added – final topic list with that addition will be posted in September 2017