ICS4U – Unit 2 – Methods Review

Overall Methods Pieces

1. Add one to a variable named a	a++
2. Subtract one from a variable named b	b--
3. If you have the code a=b, which changes, a or b?	A
4. The name of the first line of the method?	Method signature
5. A subprogram	A method
6. The name of the thing that is sent back from the method?	Return type
7. The name of the output of the method	Return type
8. The name of the input of the method	Parameter
9. Why is a method signature important?	It contains all of the information needed to call the method: (1) return type, (2) name, (3) parameters.
10. The name of the things that are sent into the method	Parameter
11. The position of the return type in the method signature	Second word, right after public.
12. What is the in the brackets of the method signature?	Parameter
13. The position of the method name in the method signature	Third word.
14. The opening word of the method signature	Public
15. The position of the parameter type in the method signature	First word in the brackets.

ORATE

16. What does ORATE stand for?	Organization Reusability Abstraction Testing Extensibility
17. What does ORATE represent?	The reasons why methods are useful.
18. Define organization from ORATE	Breaks up code into smaller logical units.
19. Define reusability from ORATE	Instead of copy/pasting code, call the method.
20. Define abstraction from ORATE	To use a method, no understanding is needed. Just call it using signature.
21. Define testing from ORATE	Repeated code has more lines AND more white box test cases. Methods reduce code AND white box testing.
22. Define extensibility from ORATE	Methods mean that future changes can occur in one place. If code is repeated, changes also need repeating.
23. What is an example of abstraction from ORATE?	IO. String methods. You didn't understand it, but could call it.
24. What is an example of reusability from ORATE?	Pizza Party button rolls. You used one method for all 5 buttons
25. What is an example of organization from ORATE?	The screens in your current project Each screen is sent up in a separate method. This keeps all of its code together and make it easy to find.

Recursive Applications

26. What is a method that calls itself?	Recursion
27. What are the first nine terms of the Fibonacci sequence?	1, 1, 2, 3, 5, 8, 13, 21, 34
28. Where does the Fibonacci sequence appear in nature?	1. Proportions of turns in a seashell 2. Proportions of a beautiful face 3. Number of seeds in a spiral of a flower 4. Reproduction patterns of rabbits
29. What is the base case of the Fibonacci sequence?	First term = 1, second term = 1
30. What is the recursive case of the Fibonacci sequence?	Term n is the two previous terms added together
31. What is 1! (one factorial)	1
32. What is 2! (two factorial)	2
33. What is 3! (three factorial)	6
34. What is 4! (four factorial)	24
35. What is 5! (five factorial)	120
36. What is the base case of factorial?	The first factorial is 1
37. What is the recursive case of factorial?	The nth factorial is the previous factorial * n
38. A use of factorials in math.	Probability calculations
39. A recursive picture	A fractal
40. A use of a fractal	CGI – computer generated images Textures (fur, wood grain) Natural shapes (trees, leaves)
41. When would you use recursion and not a loop?	Sorting. Recursive sorts are fastest.
42. When would you use a loop and not recursion?	Printing a sequence. Loops are faster than recursion.
43. Which is easier to learn: loops or recursion	Loops
44. When sorting, which is best, loops or recursion	Recursion
45. What can all recursive methods be coded as?	Loop
46. What can all loops be coded as?	Recursion

Recursion Vs Loops

47. The recursive equivalent of a loop stopping variable.	Parameter
48. The recursive equivalent of a loop stopping condition.	Base case
49. The recursive equivalent of the loop's steps to repeat.	Recursive
50. The recursive equivalent of an infinite loop	Stack Overflow Error
51. The loop equivalent of a recursive parameter	Loop stopping variable
52. The loop equivalent of a recursive base case	Loop stopping condition
53. The loop equivalent of a recursive case in a method	Steps to repeat
54. The loop equivalent of a stack overflow error	Infinite loop
55. What are two parts of a recursive method?	1. Base case 2. Recursive case
56. What is a base case used for?	1. Stops the recursion. 2. Returns the first value that all others build on
57. What is a recursive case use for?	1. Reduces the problem using a smaller parameter 2. Repeats by calling itself
58. Why does recursion have to be in a method?	1. Recursion needs to call a smaller version of itself. 2. This is needed to move the base case AND to repeat. 3. The way you "call" yourself is using a method.
59. Why does recursion need an if?	1. Recursion has two pieces: a base case and recursive case. 2. To CHOOSE between them, we need an if.
60. Why does recursion need a parameter?	1. Parameters get smaller in the recursive case. 2. When they are small, the recursion stops 3. Thus, parameters control the number of times the code is repeated.

String Functions (Return Types and Parameter Types)

61. Return type of charAt	char
62. Return type of toUpperCase	String
63. Return type of replace	String
64. Return type of length	int
65. Return type of indexOf	int
66. Return type of substring	String
67. Return type of compareTo	int
68. Return type of equals	boolean
69. Parameter type of charAt	int
70. Parameter type of toUpperCase	none
71. Parameter type of replace	char
72. Parameter type of length	none
73. Parameter type of indexOf	char
74. Parameter type of substring	int
75. Parameter type of compareTo	String
76. Parameter type of equals	String