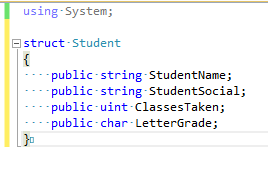
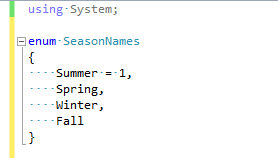
1. False
2. False
3. False
4. True
5. True
6. False (They retain their values across all instances of the class)
7. False
8. False (They are used to access hidden data)
9. False
10. True
11. B. []
12. C. overloading
13. B. aiArray[9]
14. A. int[ , ] aiArray;
15. B. passed by value
16. B. within a function
17. A. int [] numbers = new int [100];
18. B. get / set
19. A. Str1 ==Str2
20. B. Point.MyMethod();
21. The class we instantiate from gets to use the methods within the objects class and depending upon the objects class potentially the variables in that class. The result of this is easily re-usable code. We can create classes which are then able to be called upon by our other classes and access their methods to prevent the need for re-writing code. It also allows us to separate our code into manageable chunks. If one section of our code breaks down it is less likely that the errors will spread into other portions of our code, which results in a quicker resolution of program bugs.
22. 0 1 4 9 16 25 36 49 64 81
23. The code will stop with a divide by zero error on the line “long x = 1 / (1 – nFactorial);”
24. The Console will display “Shape”
25. 
26. 
    1. Display = 9 5 -2 4 5 5
    2. By nesting the loop using y as a counter inside the if statement, which replaces all later values, except the final one (xlist[6] = 5) with the next integer in the array you are changing the values inside the array. This causes the value of -2 to never be checked in the if statement. If the intention of this code was to find and replace all 3s in the array and there were ever two 3s in a row the code would miss the second 3.
    3. Display = -2 7
    4. One unusual thing about this program is that it contains a loop which only iterates one time and could therefore be removed from the program. However, this could be a test to check logic for the loop to go through a larger or variable sized array. In the case of a larger array the loop will go through each element of the array and compare it to the following one. The larger of the two will be moved to the next place and the current place will be replaced with the smaller of the two. The last element of the array will be the largest. If this was nested inside another loop for the length of the array it could be used to organize a larger array in ascending order.