



WOLLO UNIVERSITY
KOMBOLCHA INSTITUTES OF TECHNOLOGY
College of Informatics
Department of Information System (IS)

Object Oriented Programming, Worksheet I

1. Write a java program to demonstrate all the four access modifiers in java?
2. Write a java program that will print out all the multiples of 3 from 3 to 36, that is: 3 6 9 12 15 18 21 24 27 30 33 36.
3. Show the exact output that would be produced by the following main() routine:

```
public static void main(String[] args) {  
    int N;  
    N = 1;  
    while (N <= 32) {  
        N = 2 * N;  
        System.out.println (N);  
    }  
}
```

4. Write a complete static method that finds the largest value in an array of ints. The method should have one parameter, which is an array of type int[]. The largest number in the array should be returned as the value of the method.
5. Suppose that A has been declared and initialized with the statement
double[] A = new double[20];
and suppose that A has already been filled with 20 values. Write a program segment that will find the average of all the non-zero numbers in the array. (The average is the

sum of the numbers, divided by the number of numbers. Note that you will have to count the number of non-zero entries in the array.) Declare any variables that you use.

6. What is a constructor? What is the purpose of a constructor in a class?
7. What is meant by the terms instance variable and instance method?
8. Modify the following class so that the two instance variables are private and there is a getter method and a setter method for each instance variable:

```
public class Player {  
    String name;  
    int score;  
}
```

9. For this problem, you should write a very simple but complete class. The class represents a counter that counts 0, 1, 2, 3, 4 ... The name of the class should be Counter. It has one private instance variable representing the value of the counter. It has two instance methods: increment() adds one to the counter value, and getValue() returns the current counter value. Write a complete definition for the class, Counter
10. This problem uses the Counter class from the previous question. The following program segment is meant to simulate tossing a coin 100 times. It should use two Counter objects, headCount and tailCount, to count the number of heads and the number of tails.

11. Suppose that a class, *Employee*, is defined as follows:

```
class Employee {  
    String lastName;  
    String firstName;  
    double hourlyWage;  
    int yearsWithCompany;  
}
```

Suppose that data about 100 employees is already stored in an array:

```
Employee[] employeeData = new Employee[100];
```

Write a code segment that will output the first name, last name, and hourly wage of each employee who has been with the company for 20 years or more.

12. In all versions of the PairOfDice class, the instance variables die1 and die2 are declared to be public. They really should be private, so that they would be protected from being changed from outside the class. Write another version of the PairOfDice class in which the instance variables die1 and die2 are private. Your class will need "getter" methods that can be used to find out the values of die1 and die2. (The idea is to protect their values from being changed from outside the class, but still to allow the values to be read.) Include other improvements in the class, if you can think of any. Test your class with a short program

that counts how many times a pair of dice is rolled, before the total of the two dice is equal to two.

13. The Fibonacci sequence of numbers is 1, 1, 2, 3, 5, 8, 13, 21, 34 ... The first and second numbers are 1 and after that $n_i = n_{i-2} + n_{i-1}$, e.g., $34 = 13 + 21$. A number in the sequence is called a Fibonacci number. Write a method with signature **int closestFibonacci(int n)** which returns the largest Fibonacci number that is less than or equal to its argument. For example, `closestFibonacci(12)` returns 8 because 8 is the largest Fibonacci number less than 12 and `closestFibonacci(33)` returns 21 because 21 is the largest Fibonacci number that is ≤ 33 . `closestFibonacci(34)` should return 34. If the argument is less than 1 return 0. Your solution must **not** use recursion because unless you cache the Fibonacci numbers as you find them, the recursive solution recomputed the same Fibonacci number many times.
14. A number n is called **prime-happy** if there is at least one prime less than n and the sum of all primes less than n is evenly divisible by n .
Recall that a prime number is an integer > 1 which has only two integer factors, 1 and itself
Write a function named **isPrimeHappy** that returns 1 if its integer argument is prime-happy; otherwise it returns 0.
The function prototype is `int isPrimeHappy (int n);`
Examples:

if n is	return	because
5	1	Because 2 and 3 are the primes less than 5, their sum is 5 and 5 evenly divides 5.
25	1	because 2, 3, 5, 7, 11, 13, 17, 19, 23 are the primes less than 25, their sum is 100 and 25 evenly divides 100
32	1	because 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31 are the primes less than 32, their sum is 160 and 32 evenly divides 160
8	0	Because 2, 3, 5, 7 are the primes less than 8, their sum is 17 and 8 does not evenly divide 17.
2	0	Because there are no primes less than 2.

15. Write a function that takes an array of integers as an argument and returns a value based on the sums of the even and odd numbers in the array. Let X = the sum of the odd numbers in the array and let Y = the sum of the even numbers. The function should return $X - Y$
The signature of the function is:
`int f(int[] a)`
Examples

if input array is	return
{ 1 }	1

{1, 2}	-1
{1, 2, 3}	2
{1, 2, 3, 4}	-2
{3, 3, 4, 4}	-2
{3, 2, 3, 4}	0
{4, 1, 2, 3}	-2
{1, 1}	2
{}	0

16. An array is defined to be **stepped** if it is in ascending order and there are 3 or more occurrences of each distinct value in the array. Note that ascending order means $a[n] \leq a[n+1]$. It does not mean $a[n] < a[n+1]$ (this is strictly ascending). Write a function named **isStepped** that returns 1 if its array argument is stepped, otherwise return 0.

If you are programming in Java or C#, the signature is `int isStepped(int[] a)`

If you are programming in C or C++, the signature is `int isStepped(int a[], int len)` where len is the number of elements in the array.

Examples

If the array is	return	reason
{1, 1, 1, 5, 5, 5, 5, 8, 8, 8}	1	It is in ascending order. The distinct values of the array are 1, 5, 8 and there are three or more occurrences of each of these values.
{1, 1, 5, 5, 5, 5, 8, 8, 8}	0	Even though it is in ascending order, there are only two occurrences of the value 1.
{5, 5, 5, 15}	0	Even though it is in ascending order, there is only one occurrence of the value 15.
{3, 3, 3, 2, 2, 2, 5, 5, 5}	0	It is not in ascending order
{3, 3, 3, 2, 2, 2, 1, 1, 1}	0	It is not in ascending order
{1, 1, 1}	1	It is in ascending order and there are three or more occurrences of each distinct value. In this case there is only one distinct value.
{1, 1, 1, 1, 1, 1, 1}	1	It is in ascending order and there are three or more occurrences of each distinct value. In this case there is only one distinct value.