Unit Testing – component testing

Unit testing, also known as Component testing verifies the modules of the software (e.g. classes, functions/methods, modules etc.) that are separately testable.

Unit Testing – component testing

The developer writes code to test modules in the software under test.

Unit test framework support the developer.

Unit testing should be done in isolation from the rest of the system.

Stubs and drivers are used to replace the missing software and simulate the interface between the software components.

Unit Testing – component testing

A stub is called from the software component to be tested.

A driver calls a component to be tested.

Test cases are derived from work products such as the software design or the data model

Unit tests and **test suites** for Java programs can be developed in an integrated development environment, e.g. Eclipse and Netbeans.

Exercise: Unit Testing

The Java program : PerfectNumbers.java finds perfect numbers up to a given limit.

- **Use Eclipse to develop JUnit test cases for the three methods in the file *PerfectNumbers.java*.**

- **Create a JUnit test suite of all the test cases.**

(To run the program, you must add the file **PerfectTest.java**.)

Exercise: Unit Testing

For an added challenge you can try to make the program yourself!

If you need a Unit Test guide, see https://www.youtube.com/watch?v=v2F49zLLj-8

Exercise: Unit Testing

What is a perfect number?

An integer equal to the sum of all its real factors, including one (1)

Real factor means a factor less than the number itself

Examples

Integer	Real factors	Sum		Perfect?	
4	1, 2	3	1+2 = 3	NO	4≠3
6	1, 2, 3	6	1+2+3 = 6	YES	6=6
12	1, 2, 3, 4, 6	20	1+2+3+4+6 = 20	NO	12≠20
28	1, 2, 4, 7, 14	28	1+2+4+7+14 = 28	YES	28=28

Exercise: Unit Testing

PerfectNumbers.java

Calculates perfect numbers

perfect(int number): boolean

Is the given number perfect?

factorSum(int number): String

Calculate factor sum of number

findPerfectNumbers(int limit)

Find perfect numbers given limit

```
public class PerfectNumbers {

  public static boolean perfect( int number ) {
  int factorSum = 1;

  for ( int divisor = 2; divisor <= number / 2; divisor++ ) {
    if ( number % divisor == 0 )
      factorSum += divisor;
  }
  return (factorSum == number);
}

public static String factorSum( int number ) {
  String sum = "1";
  for ( int divisor = 2; divisor <= number / 2; divisor++ ) {
    if ( number % divisor == 0 ) {
      sum += " + " + divisor;
    }
  }
  return sum;
}

public static String findPerfectNumbers( int limit ) {
  String result = "perfect  number less or equals " + limit + "\n";
  for ( int i = 2; i <= limit; i++ ) {
    if ( perfect( i ) ) {
      result += i + " = " + factorSum( i ) + "\n";
    }
  }
  return result;
}
}
```