public class MaxDivisors {

    /* This program finds the integer between 1 and 10000 that has
     * the largest number of divisors. It prints out the maximum
     * number of divisors and an integer that has that many divisors.
     */

    public static void main(String[] args) {

        // One of the integers whose divisors we have to count.
        int n;
        // Maximum number of divisors seen so far.
        int maxDivisors = 0;
        // A value of n that had the max number of divisors.
        int numWithMax = 0;

        /* Process all the values of n from 1 to 10000, and update
         * the values of maxDivisors and numWithMax whenever we find
         * a value of n that has more divisors than the current value
         * of maxDivisors.
         */

        for ( n = 1;  n <= 10000;  n++ ) {

            // A number to be tested to see if its a divisor of n.
            int d;
            // Number of divisors of n.
            int divisorCount = 0;

            // Count the divisors of n.
            for ( d = 1;  d <= n;  d++ ) {
                if ( n % d == 0 )
                    divisorCount++;
            }

            if (divisorCount > maxDivisors) {
                maxDivisors = divisorCount;
                numWithMax = n;
            }

        }

        System.out.println("Among integers between 1 and 10000, ");
        System.out.println("The maximum number of divisors is "+maxDivisors);
        System.out.println("A number with "+maxDivisors+" divisors is "+numWithMax);
    }
}

Examples
A hexadecimal integer is a sequence of hexadecimal digits, such as 34A7, FF8, 174204, or FADE. The hexadecimal digits are the ordinary, base-10 digits '0' through '9' plus the letters 'A' through 'F'. In the hexadecimal system, these digits represent the values 0 through 15, respectively.

public class Hex2Dec {
    /* This program takes a hexadecimal number input on the
     * command line and prints the base-10 equivalent. If
     * the input contains characters that are not hexadecimal
     * numbers, then an error message is printed.
     */

    /** Method that returns the hexadecimal value of ch, or returns
     * -1 if ch is not one of the hexadecimal digits.
     */
    public static int hexValue(char ch) {
        switch (ch) {
            case '0':
                return 0;
            case '1':
                return 1;
            case '2':
                return 2;
            case '3':
                return 3;
            case '4':
                return 4;
            case '5':
                return 5;
            case '6':
                return 6;
            case '7':
                return 7;
            case '8':
                return 8;
            case '9':
                return 9;
            case 'a':
            case 'A':
                return 10;
            case 'b':
            case 'B':
                return 11;
            case 'c':
            case 'C':
                return 12;
            case 'd':
            case 'D':
                return 13;
            case 'e':
            case 'E':
                return 14;
            case 'f':
            case 'F':
                return 15;
            default:
                return -1;
        }
    }
}
public static void main(String[] args) {

    // Make sure user supplied a menu choice
    if (args.length == 0) {
        System.out.println("Please enter a Hexidecimal number");
        System.out.println("Example: java Hex2Dec 9F");
        System.exit(0);
    }

    // Input from user, containing a hexadecimal number.
    String hex = args[0];

    // Decimal (base-10) equivalent of hexadecimal number.
    long dec = 0;

    for ( int i = 0; i < hex.length(); i++ ) {
        int digit = hexValue( hex.charAt(i) );

        if (digit == -1) {
            System.out.println("Error: Input is not a hexadecimal number. ");
            System.exit(0);
        }

        dec = 16*dec + digit;
    }

    // print the decimal equivalent to the screen
    System.out.println("Base-10 value: " + dec);
}
}
public class Scope {
    // instance variable
    private int x;

    // Constructor initializes instance variable to 1
    public Scope() {
        this.x = 1;
    }

    public void start() {
        // variable local to method start
        int x = 5;
        System.out.println("Local x in method start is " + x);
        System.out.println("First call to methodA()");
        this.methodA();
        System.out.println("First call to methodB()");
        this.methodB();
        System.out.println("Second call to methodA()");
        methodA();
        System.out.println("Second call to methodA()");
        methodB();
        System.out.println("Local x in method start is " + x);
    }

    public void methodA() {
        // initialized each time methodA() is called
        int x = 25;
        System.out.println("Local x in methodA is " + x);
        ++x;
        System.out.println("Local x after incrementing in methodA is " + x);
    }

    public void methodB() {
        System.out.println("Instance variable x is " + x);
        x *= 10;
        System.out.println("Instance variable x * 10 is " + x);
    }

    public static void main(String args[]) {
        Scope testScope = new Scope();
        testScope.start();
    }
}
public class PairOfDice {
    private int die1;
    private int die2;

    // Constructor. Rolls the dice, so that they initially
    // show some random values.
    public PairOfDice() {
        roll();
    }

    // Roll the dice by setting each of the dice to be
    // a random number between 1 and 6.
    public void roll() {
        die1 = (int)(Math.random()*6) + 1;
        die2 = (int)(Math.random()*6) + 1;
    }

    // Return the number showing on the first die.
    public int getDie1() {
        return die1;
    }

    // Return the number showing on the second die.
    public int getDie2() {
        return die2;
    }

    // Return the total showing on the two dice.
    public int getTotal() {
        return die1 + die2;
    }
}
/*  Rolls a pair of dice until the dice come up snake eyes
 * (with a total value of 2).  Counts and reports the
 * number of rolls.
 */

public class SnakeEyes {

    public static void main(String[] args) {

        PairOfDice dice;
        int rollCount;

        dice = new PairOfDice();
        rollCount = 0;

        /* Roll the dice until they come up snake eyes. */
        do {
            dice.roll();
            System.out.println("You rolled " + dice.getDie1()
                    + " and " + dice.getDie2());
            rollCount++;
        } while (dice.getDie1() != 1 || dice.getDie2() != 1);

        /* Report the number of rolls. */
        System.out.println("It took " + rollCount + " rolls to get a 2.");
    }
}
Exercises

1. Create a program (Append.java) that loops through all the command-line arguments to create a comma-delimited list.
2. Use a StringBuffer as the holder of the list. Be sure there is no comma after the last element.
3. Print out the list after all the arguments are combined.

When run with this sample command line:

```
java Append one two three
```

The results should be:

```
one, two, three
```

---

1. Create a class called CaseofStrings.
2. Declare a variable of type char. Assign the first command-line argument to the variable.
3. Declare a variable of type String called sentence that says, "It's fun developing applications in the Java programming language."
4. Use a switch statement to create a menu that takes the char A, B, C, D. When using type char, you must enclose the char in ". For example, 'A'.
5. For case A, have the application print out the length of the String, if the length is greater than 0. Otherwise, print a message saying there wasn't anything to count.
6. For case B, print the String to all uppercase letters.
7. For case C, print a substring, using the startIndex of 40 and an endIndex of 65 as the arguments.
8. For case D, replace the letters t with Z throughout the String.
9. For any other menu selection, print the message "Sorry, you didn't pick a valid menu option".

When run with this sample command line:

```
Java CaseofStrings A
```

The results should be:

```
66
```

When run with this sample command line:

```
Java CaseofStrings B
```

The results should be:

```
IT'S FUN DEVELOPING APPLICATIONS IN THE JAVA PROGRAMMING LANGUAGE.
```
1. Create a program that takes an `ArrayList` of objects as input.
   a. Make 3 of the objects strings.
   b. Make 3 of the objects numbers.
   c. Add another element that is neither a `String` nor a number.
2. Iterate through the elements.
   a. Concatenate the strings.
   b. Sum the numbers.
   c. Ignore the rest.
3. Use the keyword `instanceof` to determine what type each element is.
4. Cast the element to the appropriate type before you access the data.
5. Output the concatenated strings.
6. Output the sum of the numbers.