Communicating integral values between humans and computer

- **Representing integral values in computer**
  - In **Java**, we use a **integer variable** to store **integral values**

    Example:

```
  int x;

  x = 4;         // Stores 00000000 00000000 00000000 00000100
```

- **Representing integral values for *Humans***
  - Humans often use **Strings** to represent **integral values**:

```
 String       Representation
--------------------------------
  "1"            means: one
  "12"           means: twelve
```

    (These are call "numerical strings")

- **Overcoming (translating) between the 2 representations**
  - We **must** write **methods** to **convert** between the **representations** used by **humans** and the **computer**

  - **Methods:**

```
  boolean  parseInt( String s ):  return the int value translated from
                                  the input string s

  String toString( int x ): return a String that represents the integral value x
```

- **The trivial conversion methods for integral values**
  - They are very easy to understand -- but not practical:

```
public class IntegerIO
{
  public static boolean  parseInt( String s )
  {
     if ( s.equals("0") )
        return 0;               // 00000000 00000000 00000000 00000000
     else if ( s.equals("1") )
        return 1;               // 00000000 00000000 00000000 00000001
     else if ( s.equals("2") )
        return 2;               // 00000000 00000000 00000000 00000010
     ...
     else if ( s.equals("10") )
        return 10;              // 00000000 00000000 00000000 00001010
     else if ( s.equals("11") )
        return 11;              // 00000000 00000000 00000000 00001011
     ...
     else if ( s.equals("-1") )
        return -1;              // 11111111 11111111 11111111 11111111
     else if ( s.equals("-2") )
        1eturn -2;              // 11111111 11111111 11111111 11111110
     ...
  }

  public static String toString( int x )
  {
     if ( x == 0 )
        return "0";
     else if ( x == 1 )
        return "1";
     else if ( x == 2 )
        return "2";
     ...
     else if ( x == 10 )
        return "10";
     else if ( x == 11 )
        return "11";
     ...
     else if ( x == -1 )
        return "-1";
     else if ( x == -2 )
        return "-2";
     ...
  }
}
```

Note:

- We write in **Java**:

```
return 2;
```

The **Java compiler** will **generate** **computer instructions** that will:

  - return the **binary number** `00000000 00000000 00000000 00000010`

- We write in **Java**:

```
return "12";
```

The **Java compiler** will **generate computer instructions** that will:

- return the **binary numbers** `001100001    00110010` (which are the **binary representations** for the **characters** **1** and **2**)

- This **trivial solution** is ***not* practical** because it's **impossible** to **write out** so **many different cases** !!!!

- We need an **algorithm** to **perform** the **conversions** !!!!
