**TOP-DOWN DESIGN**

*Top-down design* is a process that moves from a high-level design concept towards a lower level. A *high-level design* concept is larger, broader and more general; a *low-level design* concept is smaller, more specific and detailed.

Top-down design arises in every human endeavor and is related to the organization of perception and memory in the human brain.

**Example**

This Civil War era drawing by Edwin Forbes (1839 – 1895) illustrates top-down design in art. The artist started by roughly sketching the entire scene and followed by adding appropriate detail to the mule team driver and his mount.

![Edwin Forbes, 1863, [Public domain] via Library of Congress](Image)
In computer programming top-down design starts with a global description of the program to be developed in which its data and tasks are described at a high-level. This high-level description is repeatedly refined until the design is sufficiently understood and detailed to allow it to be implemented using a programming language such as Java.

**Top-Down Computer Program Design**

Begin with a global, high-level description of the program's components

![](image)

Description of program's components

Is the description correct, complete and detailed enough?

YES → Implement the component

NO → Select a component and refine its description

Top-down design is facilitated by tools that are flexible enough to describe design elements at various levels of detail; pseudo-code is such a tool.

**Example**

As an illustration of top-down computer program design, start with a high-level description of a program that finds the median of a list of students’ test scores.

find the median of a list of students’ test scores

The median is the “middle” score. For example, 60 is the middle of 30, 50, 60, 90 and 100. To find the median we must sort the scores into ascending order. This leads to the following refinement of the algorithm:
The “middle score” depends on whether the number of scores is odd or even. For example, 60 is the middle of 30, 50, 60, 90 and 100 whereas 70 is the middle of 30, 50, 60, 80, 90 and 100. Therefore, if the number of scores is even the median is the average of the middle two scores. The refined algorithm is:

Further refinement depends on refining what is meant by a list of scores. A common data structure in programming is the array, which is a sequence of values, each of which is identified by a unique subscript that begins with 0 and increases by 1. For example, if the list of students’ scores were stored into the array \( L = (30, 50, 60, 90, 100) \) then \( L_0 \) denotes the 30, \( L_1 \) denotes the 50, \( L_2 \) the 60 and so forth.

With the scores placed into an array, we can refine what is meant by select the middle score and select the two middle scores.

If there are 5 scores in the array then the middle score is at position 2. If there are 11 scores, the middle score is at position 5. In general, if \( n \) is the number of scores and \( n \) is odd then \( \frac{n-1}{2} \) is the position of the middle score. Draw several more examples to convince yourself of this.

Similarly, if there are 6 scores in the array then the two middle scores are at positions 2 and 3. If there are 12 scores, the two middle scores are at positions 5 and 6. In general, if \( n \) is the number
of scores and $n$ is even then the position of the two middle scores are $\frac{n}{2} - 1$ and $\frac{n}{2}$. Again, if you need convincing, draw some examples.

This analysis leads to the following refinement of the algorithm:

```plaintext
input the list of students' scores into array L
n = the number of scores
sort array L into ascending order
if n is odd then
    median = $L_{\frac{n}{2}}$
else
    median = $\frac{L_{\frac{n}{2} - 1} + L_{\frac{n}{2}}}{2}$
end if
output the median
```

More refinements of this algorithm are possible but what has been done so far serves to illustrate the top-down computer program design process.

Refinement continues until the programmer understands the algorithm well enough to be able to code it into a computer language such as Java.
History
Top-down computer program design was independently invented many years ago by IBM Corporation\textsuperscript{1}, Nicklaus Wirth\textsuperscript{2} and Edsger Dijkstra\textsuperscript{3}. IBM called its method \textit{top-down design}; Wirth called his \textit{stepwise refinement} and Dijkstra \textit{hierarchical decomposition}.

\textbf{Edsger Dijkstra}  
\textit{(1930 2002)}

\textbf{Nicklaus Wirth}

\textsuperscript{1} IBM Corporation, \textit{HIPO - A Design Aid and Documentation Technique}, Publication Number GX20-1851, White Plains, NY, October, 1974.
