Topic: 4.2.2 Jackson Structured Programming (JSP)

Jackson structured programming (JSP) is a method for program design and modelling "in the small". It begins with considerations about what is known and develops a program design that becomes more complete as the model is put through continued iterations. It emerged in the 1970's as a design method that concentrated on the structure of data using data-structure methods. Because it builds the program design from an incomplete model. JSP structures the programs and data in terms of sequences, iterations and selection.

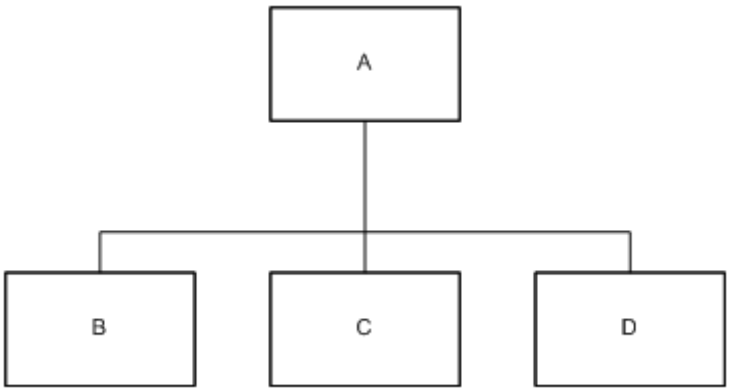
The method begins by describing a program's inputs in terms of the four fundamental component types (fundamental operations, sequences, iterations, selections) it then goes on to describe the program's outputs in the same way.

JSP uses a diagramming notation to describe the structure of inputs, outputs and programs with diagram elements for each of the fundamental component types.

A simple **operation** is drawn as a box.

An operation

A **sequence** of operations is represented by boxes connected with lines. In the example below, operation A consists of the sequence of operations B, C and D.

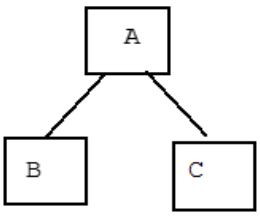


A sequence

Topic: 4.2.2 Jackson Structured Programming (JSP)

A sequence is a composite component that has two or more parts occurring once each, in a specific order. Take the following diagram as an example.

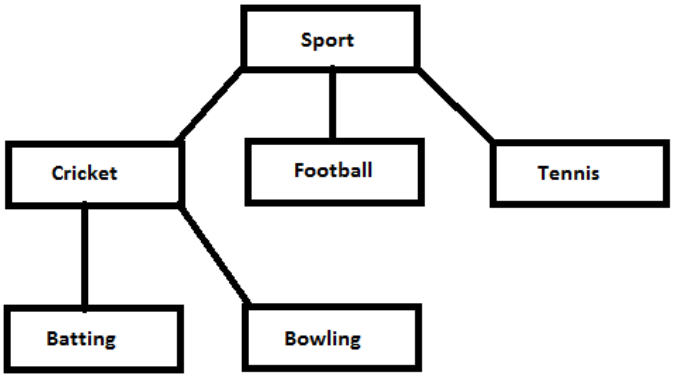
Jackson structure diagram



Pseudocode

```
begin
   do B;
   do C;
end
```

Take the following structure diagram for example:



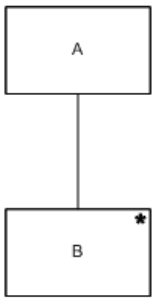
Pseudocode can be written as:

```
Begin sport
      do Cricket;
      do Football;
      do Tennis;
End sport


Begin Cricket
      do Batting;
      do Bowling;
End Cricket
```

Topic: 4.2.2 Jackson Structured Programming (JSP)

An **iteration** is again represented with joined boxes. In addition, the iterated operation has a star in the top right corner of its box. An iteration is a composite component that consists of one part that repeats zero or more times. In the example below, operation A consists of an iteration of zero or more invocations of operation B.

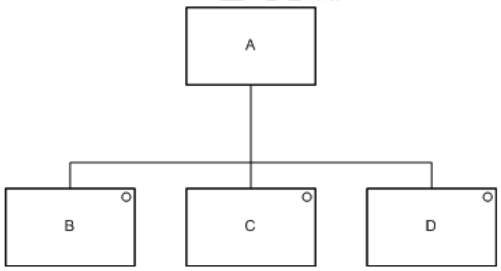


An iteration

Pseudocode

```
while   <cond>
   do B;
endwhile
```

A **selection** is similar to a sequence, but with a circle drawn in the top right hand corner of each optional operation. A selection is a composite component that consists of 2 or more parts, only one of which is selected once. In the example, operation A consists of one and only one of operations B, C or D.

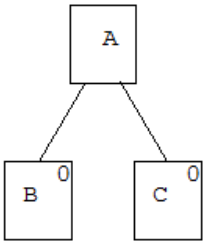


A selection

Topic: 4.2.2 Jackson Structured Programming (JSP)

Consider the following example.

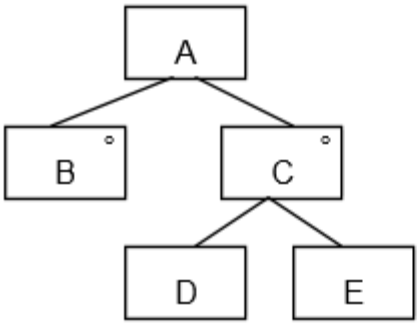
Jackson structure diagram



Pseudocode

```
if <cond-1>then
  do B;
else if  <cond-2>then
  do C;
endif
```

Jackson structure diagram



Pseudocode

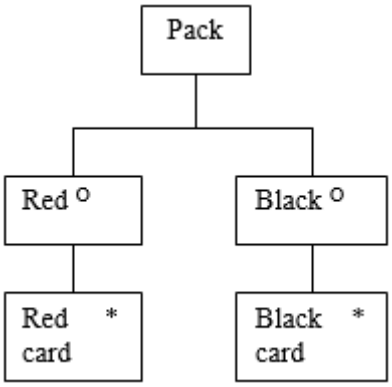
```
if  <cond-1> then
  do B;
else if   <cond-2> then
  do D;
  do E;
endif
```

```
if <cond-1>
   Then
        do B
   Else
        do D
        do E
   End if
```

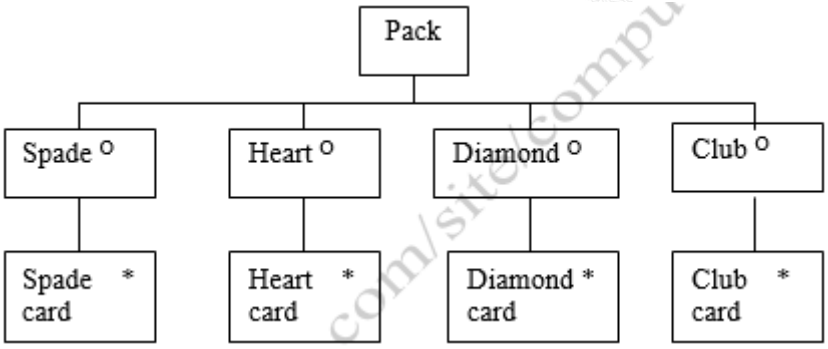
A selection with 2 paths is an "if then else" and a selection with 3 or more paths has to be a "case select".

Topic: 4.2.2 Jackson Structured Programming (JSP)

Consider a pack of ordinary playing cards which have been divided into red and black suits. Illustrated in the figure below. The top level shows we are using a pack; the second layer shows that the pack is divided into red and black components. The third level shows that the red component consists of many (or maybe zero) cards. Similarly, black consists of many cards.

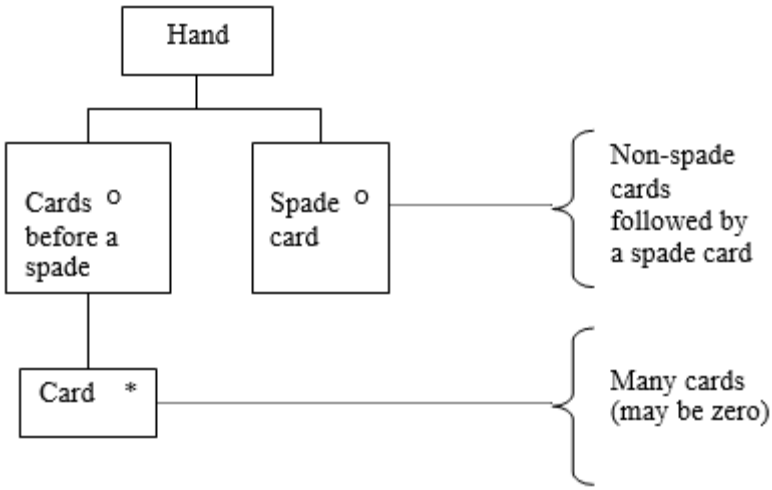


The pack can further be divided into suits.



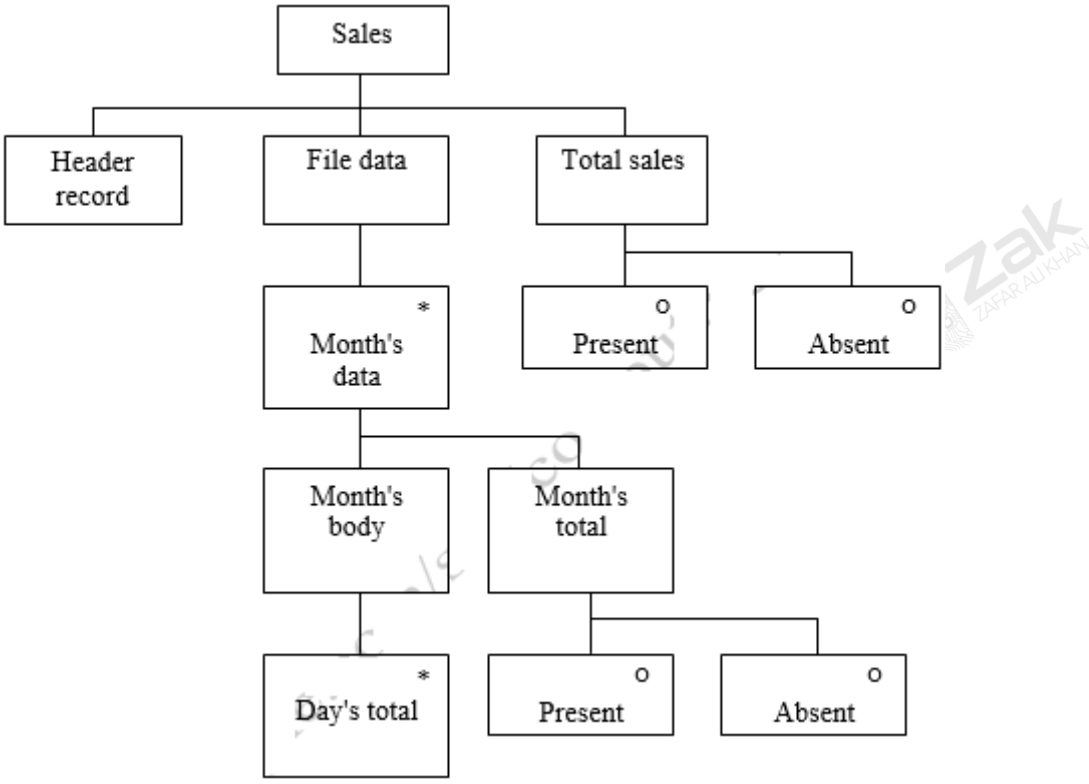
Topic: 4.2.2 Jackson Structured Programming (JSP)

Now suppose we deal a hand of cards from a shuffled pack until you pick up a spade card. The data structure is shown below:



Topic: 4.2.2 Jackson Structured Programming (JSP)

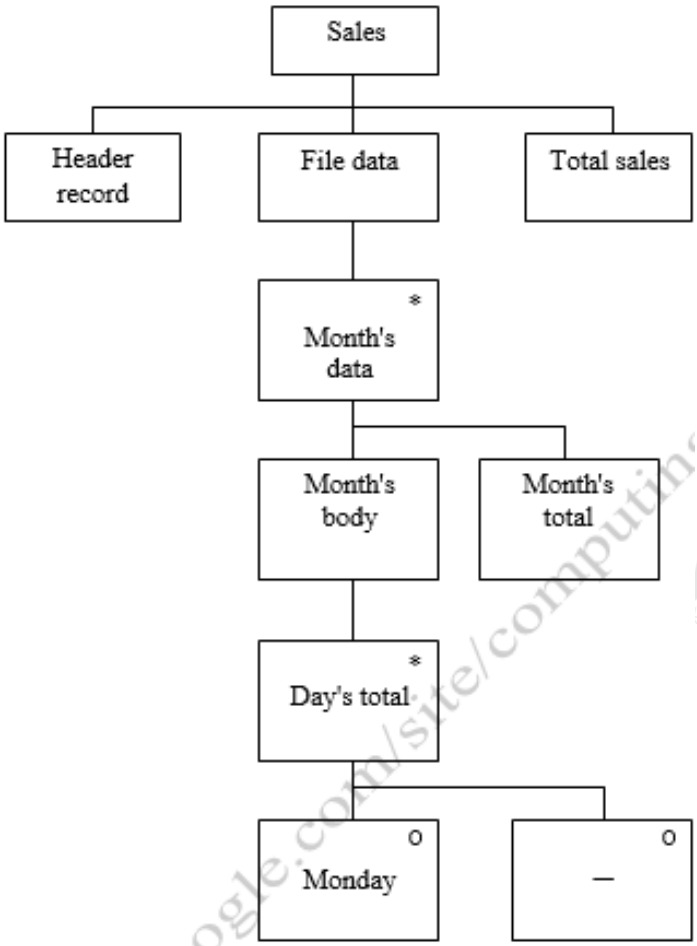
Now consider a sequential file that contains daily takings of a shop in date order. At the end of each month's takings is a grand total for the month. At the end of the file is the total for all the months recorded. These totals act as validation checks. At the start of the file is a header record describing the contents of the file. Present and absent are processes that are performed according to the presence or absence of the respective totals.



The JSP diagrams we have seen so far have shown physical data structures. Logical data structures describe the data with respect to a given application.

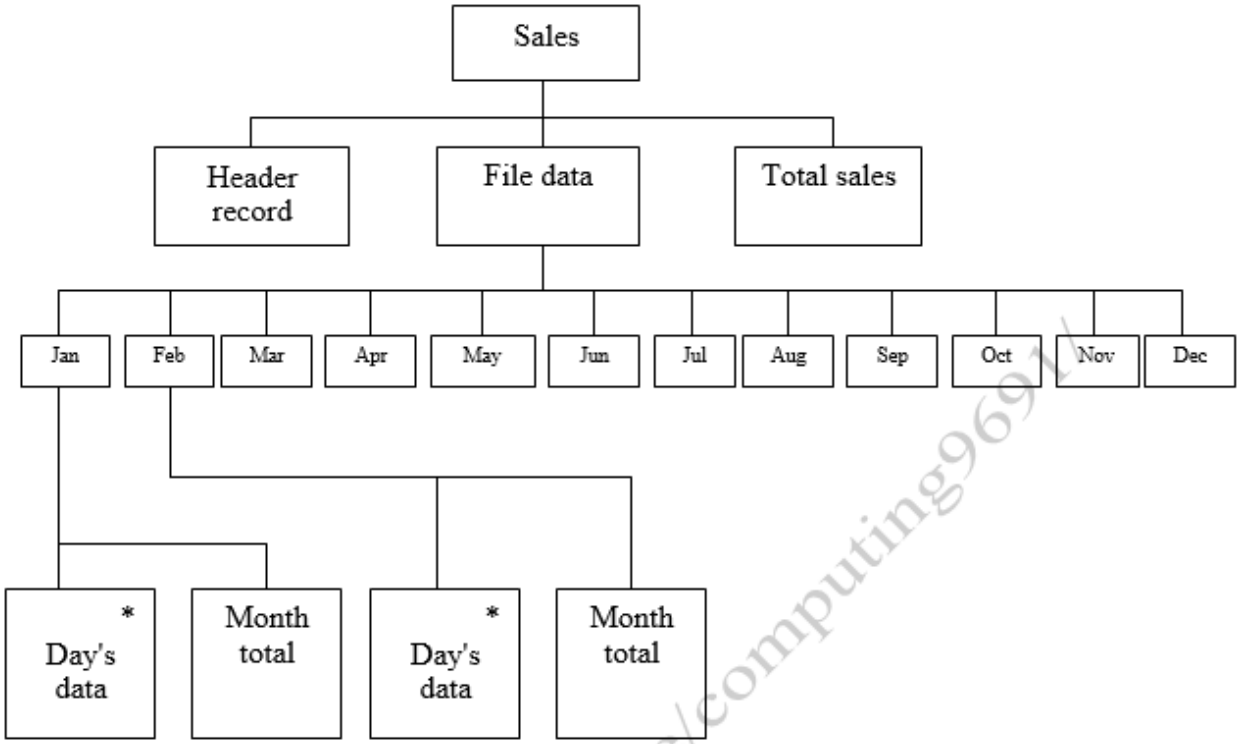
Suppose we wish to extract the takings for all Mondays from the file shown above. The logical diagram is shown below. Notice that the diagram does not violate the data structure shown in the previous diagram. Also, notice the use of null (—) in the decision at the bottom of the diagram; this shows that the ELSE part of the decision does nothing. Further there are no decisions below the totals in this diagram because they are not being used.

Topic: 4.2.2 Jackson Structured Programming (JSP)



Now suppose the application is to extract February's data. In this case we need to read past January totals so these must be shown in the logical data structure. However, once we have dealt with February, we do not wish to read the remaining months' data. The figure below shows the logical structure for this problem.

Topic: 4.2.2 Jackson Structured Programming (JSP)



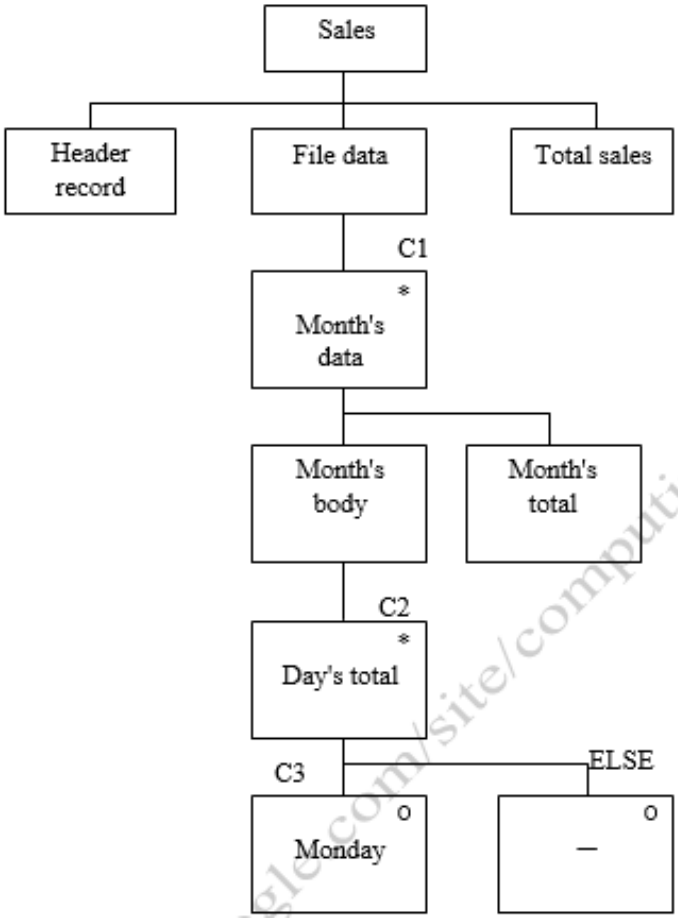
The next step is to enter, on the diagram, the constraints. This is done by numbering the constraints and then listing their meanings. Placing the constraints on the previous diagram will produce the diagram below where:

Topic: 4.2.2 Jackson Structured Programming (JSP)

C1 is until end of file data

C2 is until end of month's body

C3 is day = Monday



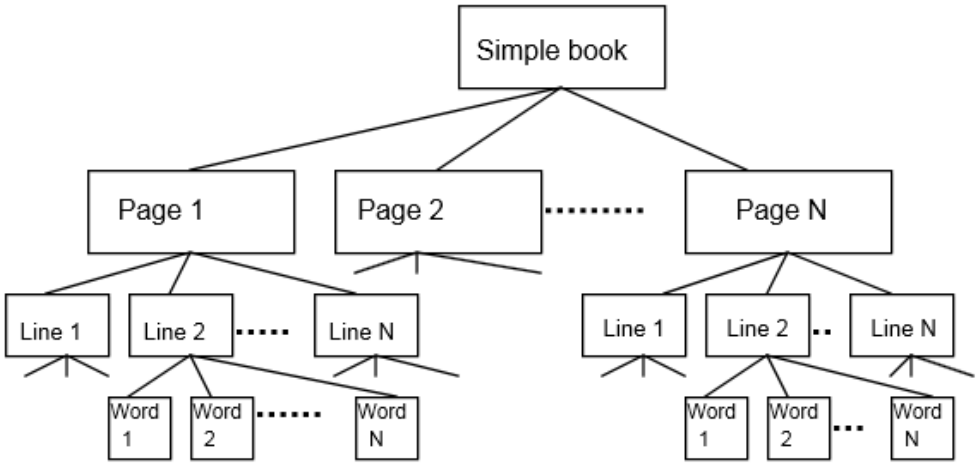
In this example, separate procedures can be written for each box in the logical structure. That is, each section in the diagram can be a separate procedure or function. Each procedure and function can use parameters to pass data to and from the calling procedure or function.

Topic: 4.2.2 Jackson Structured Programming (JSP)

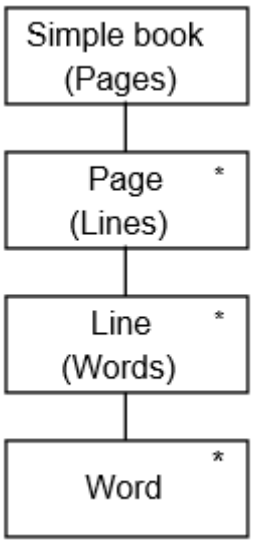
Example: **A simple book**

A simple book consists of pages; a page consists of lines of text; a line consists of words.

Here is a first attempt at a structure diagram:



We can simplify this effort by replacing each sequence by an iteration:



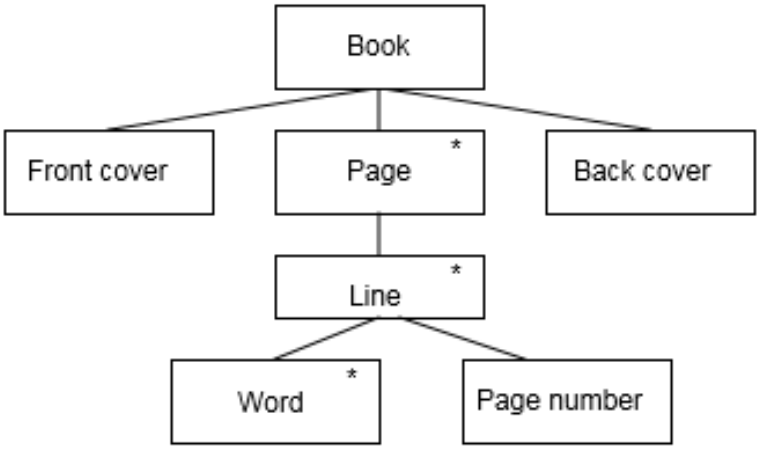
We see from this example, that an iteration is a generalization of a sequence. Note that a simple book consists of pages; a simple book (pages) is an iteration—the part that iterates is a single page. Similarly, a page is equivalent to lines; a page (lines) is an iteration – the part that iterates is a single line. And so on with line (words).

Topic: 4.2.2 Jackson Structured Programming (JSP)

Example: **A complex book**

A book consists of front and back cover with pages in between. Each page consists of lines of text; each line consists of words. At the bottom of each page is a page number.

A first effort to draw the structure diagram for a book as described above might yield something like the following:



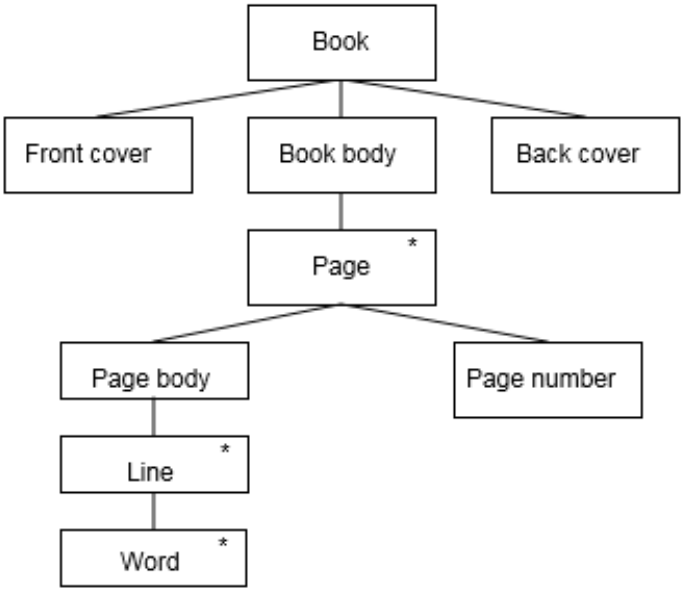
This structure diagram is incorrect, first of all, formally, that is grammatically: For, we may ask, what kind of component is a book? It appears that a book is an iteration, since the pages iterate. On the other hand, book appears to be a sequence, since there are three consecutive parts – front cover, page, and back cover but this is an impossible situation a composite component must either be a sequence, a selection, or an iteration – it cannot be a hybrid combination. While a sequence does have three parts, none of them repeat – each occurs exactly once; while an iteration has a part that repeats, it has one and only one part.

A similar, formal error in the diagram occurs in the part that shows a line as having 2 parts, one of which iterates. So, what kind of component is a line? It cannot be an iteration, since it has 2 parts; it cannot be a sequence since one part iterates and the other doesn't. It is an impossible construct that violates the grammar of structure diagram construction.

A third error is in the placement of page number. From the diagram, a page number appears after all of the words in each line. In this representation, it shows the page number at the end of each line.

Topic: 4.2.2 Jackson Structured Programming (JSP)

The correct structure diagram for a book is as shown below:



In the structure diagram, note that we created a name for an iteration that is part of a sequence “Book Body" is the part of a sequence that comes after the front cover but before the back cover, “page body” is the part of a page that comes before the page number.