codeanatomy – Draw Code Anatomy*

Usage with listings

Hồng-Phúc Bùi†

Released 2023/01/24

Contents

1 General Usage with Package listings

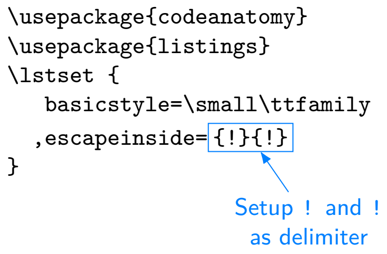
1.1 Setup Package listings

The most important setup for the package listings is the delimiter to escape LaTeX commands in Listing. With this escape delimiter we can mark a piece of code as with `\cPart`. In this example we use `!` and `!` as delimiter. Code between `!` and `!` is evaluated as LaTeX-code.

---

[*] This file describes v0.4-Beta, last revised 2023/01/24.

[†] E-mail: hong-phuc.bui (at) htwsaar dot de



```
\usepackage{codeanatomy}
\usepackage{listings}
\lstset {
   basicstyle=\small\ttfamily
  ,escapeinside={!}{!}
}
```

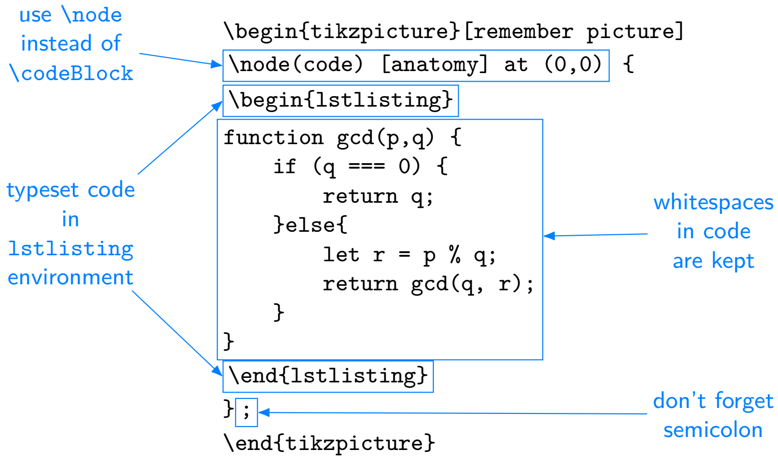
Setup ! and ! as delimiter

Delimiter can also be reset in `document`-Environment, typical just before a new `\begin{lstlisting}` environment so each anatomy can have different delimiter. The fact is, in this document I use `+` and `+` for the above listing, so that I can typeset `!` in this listing.

You may also want to set option `keepspaces` to `true`, sothat your reader can easy copy past your example code.

1.2 Typeset Code

The command `\codeBlock` does not work if the environment `lstlisting` is passed to its argument. So instead of `\codeBlock` we must use the Ti*k*Z command `\node`:



```
\begin{tikzpicture}[remember picture]
\node(code) [anatomy] at (0,0) {
\begin{lstlisting}
function gcd(p,q) {
    if (q === 0) {
        return q;
    }else{
        let r = p % q;
        return gcd(q, r);
    }
}
\end{lstlisting}
};
\end{tikzpicture}
```

use \node instead of \codeBlock

typeset code in lstlisting environment

whitespaces in code are kept

don't forget semicolon

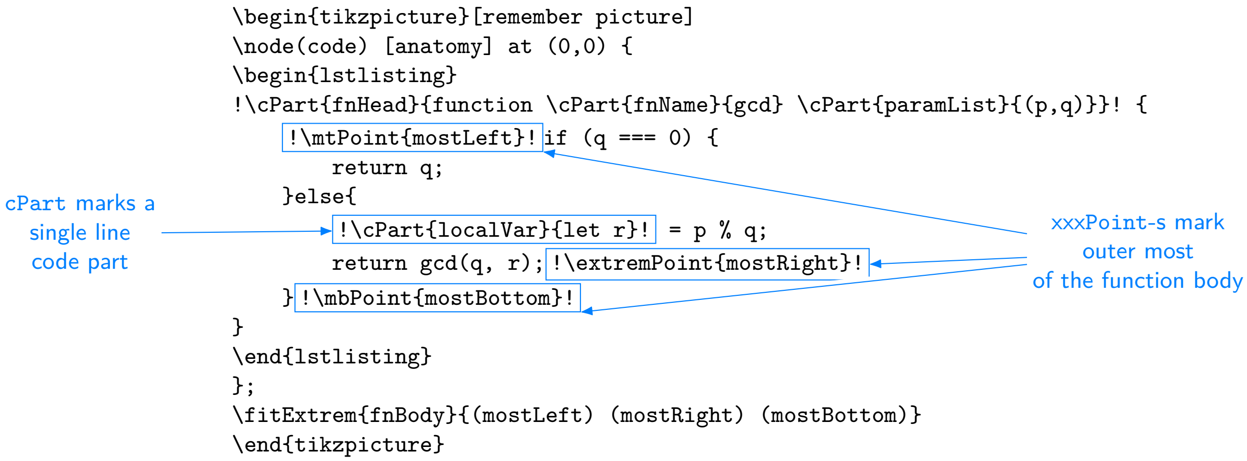
Figure 1 shows result of the above code.

1.3 Mark Code

The command `\cPart` can be used to mark single-line code parts. For multiple-line code parts once can use `\xxxPoint` family to mark the outer most points of code parts and `\fitExtrem` to cover exterm points of a code part. These commmands must be put between escape delimiter, here `!` and `!`.

```
function gcd(p,q) {
    if (q === 0) {
        return q;
    }else{
        let r = p % q;
        return gcd(q, r)
    }
}
```

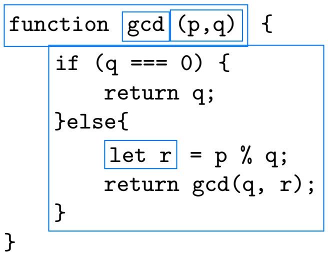
Figure 1: Code Listing is formatted



```
\begin{tikzpicture}[remember picture]
\node(code) [anatomy] at (0,0) {
\begin{lstlisting}
!\cPart{fnHead}{function \cPart{fnName}{gcd} \cPart{paramList}{(p,q)}}! {
    !\mtPoint{mostLeft}!if (q === 0) {
        return q;
    }else{
        !\cPart{localVar}{let r}! = p % q;
        return gcd(q, r);!\extremPoint{mostRight}!
    }!\mbPoint{mostBottom}!
}
\end{lstlisting}
};
\fitExtrem{fnBody}{(mostLeft) (mostRight) (mostBottom)}
\end{tikzpicture}
```

cPart marks a single line code part
xxxPoint-s mark outer most of the function body

Figure 2 shows the result of the above code.



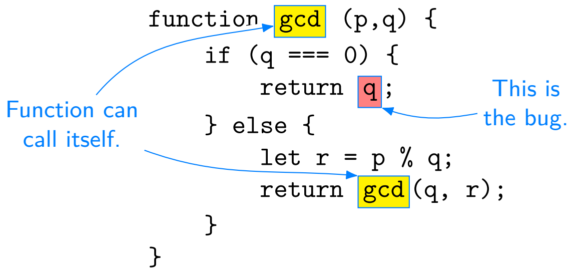
```
function gcd(p,q) {
    if (q === 0) {
        return q;
    }else{
        let r = p % q;
        return gcd(q, r);
    }
}
```

Figure 2: Code Listing with mark of code parts

1.4 Highlight some tokens

```
\lstset{escapeinside={!}{!},basicstyle=\linespread{1.8}}
\begin{tikzpicture}[remember picture]
\tikzstyle{token} = [code part, fill=yellow]
```

```
\tikzstyle{bug} = [code part, fill=red!50]
\node(code) [anatomy] at (0,0) {
\begin{lstlisting}
function !\cPart[token]{fnName}{gcd}! (p,q) {
    if (q === 0) {
        return !\cPart[bug]{bug}{q}!;
    } else {
        let r = p % q;
        return !\cPart[token]{recursive}{gcd}!(q, r);
    }
}
\end{lstlisting}
};
\codeAnnotation{recursiveText}    (-1,2) {Function can\\call itself.}
\codeAnnotation{bugText}         ( 5,2.25) {This is\\the bug.}

\draw[->, annotation] (recursiveText) to[out=50, in=190] (fnName);
\draw[->, annotation] (recursiveText) to[out=-20, in=175] (recursive.north west);
\draw[->, annotation] (bugText)       to[out=190,in=-20] (bug.south east);
\end{tikzpicture}
```



```
function gcd (p,q) {
    if (q === 0) {
        return q;
    } else {
        let r = p % q;
        return gcd(q, r);
    }
}
```

1.5 Add Annotations to Listing

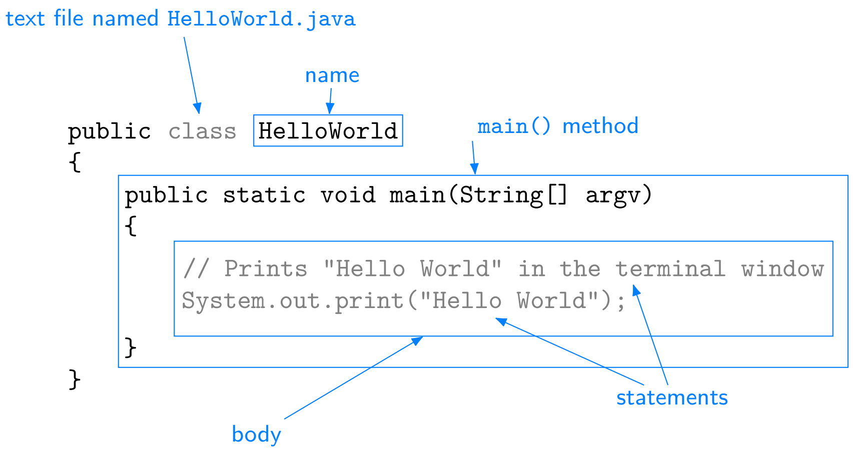
This step is the same as the description in the usage document of package codeanatomy. Readers can typeset annotations to the above listing like an exercise.

2 Some examples

Most of examples in this section are redrawn from the textbook [1].

2.1 Anatomy of a Java Program [1, p. 5]

```
\lstset{escapeinside={!}{!}}
\begin{tikzpicture}[remember picture]
\node(code) [anatomy] at (0,0){%
\begin{lstlisting}
public !\iPart{class}{class}! !\cPart{className}{HelloWorld}!
{
    !\mtPoint{mainLeft}!public static void main(String[] argv)
    {
        !\hmtPoint{left}\iPart{assign}{
            \bgcode{// Prints "Hello World" in the terminal window}}
            \extremPoint{fnR} \extremPoint{mR}!
        !\iPart{fnCall}{System.out.print("Hello World");}\dmbPoint{mostBottom}!
    }!\mbPoint{mainBottom}!
}
\end{lstlisting}
};

\fitExtrem{classBody}{(mainLeft) (mR) (mainBottom)}
\fitExtrem{functionBody}{(left) (fnR) (mostBottom)}


\codeAnnotation{fileNameText} (1.5,5) {text file named \texttt{HelloWorld.java}}
\codeAnnotation{classNameText} (3.5,4.25) {name}
\codeAnnotation{classBodyText} (6.5,3.6) {\texttt{main()} method}
\codeAnnotation{functionBodyText} (2.5,-0.5) {body}
\codeAnnotation{statement} (8,0) {statements}

{[on background layer]
\draw[->,annotation] (fileNameText) -- (class);
\draw[->,annotation] (classNameText) -- (className);
\draw[->,annotation] (classBodyText.south west) -- (classBody);
\draw[->,annotation] (functionBodyText) -- (functionBody);
\draw[->,annotation] (statement) -- (assign.353);
\draw[->,annotation] (statement) -- (fnCall.350);
}
\end{tikzpicture}
```

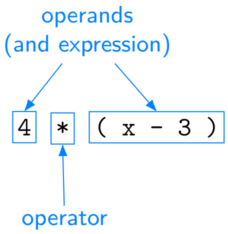


```
public class HelloWorld
{
    public static void main(String[] argv)
    {
        // Prints "Hello World" in the terminal window
        System.out.print("Hello World");
    }
}
```

2.2 Anatomy of an expression [1, p. 17]

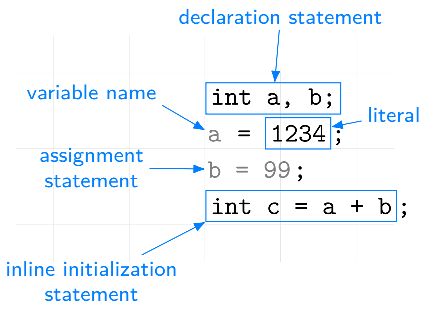
```
\lstset{escapeinside={!}{!}}
\begin{tikzpicture}[remember picture]
\codeBlock{\cPart{op1}{4} \cPart{op}{*} \cPart{op2}{( x - 3 )} }

\codeAnnotation{operand}  (1,1.5) {operands\\(and expression)}
\codeAnnotation{operator} (0.7,-1) {operator}
```

```
\draw[->,annotation] (operand) -- (op1.north);
\draw[->,annotation] (operand) -- (op2.north);
\draw[->,annotation] (operator) -- (op.south);
\end{tikzpicture}
```



2.3 Using a primitive Data Type [1, p. 17]

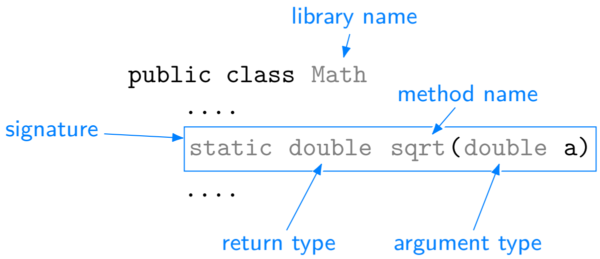
```
\lstset{escapeinside={!}{!}}
\begin{tikzpicture}[
     remember picture %
    ,code annotation/.append style = { % customize style of annotation text
        font=\sffamily\footnotesize
    }
]
{[on background layer]\draw[code grid debug] (-2.5,-0.5) grid (2.5,2.5);}
\node(code) [anatomy] at (0,0){%
\begin{lstlisting}
!\cPart{d}{int a, b;}!
!\iPart{v}{a}! = !\cPart{l}{1234}!;
!\iPart{a}{b = 99}!;
!\cPart{i}{int c = a + b}!;
\end{lstlisting}
};
% Annotations
\codeAnnotation{declareText}   (   1,2.75 )   {declaration statement}
\codeAnnotation{literalText}   (  2.5,1.45)   {literal}
\codeAnnotation{varText}       (-1.5,1.75 )   {variable name}
\codeAnnotation{assignText}    (-1.5,0.75 )   {assignment\\statement}
\codeAnnotation{initText}      (-1.5,-0.75)   {inline initialization\\statement}
% Arrows
\draw[->,annotation] (declareText) -- (d);
\draw[->,annotation] (literalText) -- (l);
\draw[->,annotation] (varText.south east) -- (v);
\draw[->,annotation] (assignText) -- (a);
\draw[->,annotation] (initText) -- (i.south west);
\end{tikzpicture}
```



```
int a, b;
a = 1234;
b = 99;
int c = a + b;
```

2.4 Anatomy of a method signature [1, p. 30]

```
\lstset{escapeinside={!}{!}}
\begin{tikzpicture}[remember picture]
\node(code) [anatomy] at (0,0) {
\begin{lstlisting}
public class !\iPart{l}{Math}!
    ....
    !\cPart{s}{\bgcode{static} \iPart{r}{double} \iPart{n}{sqrt}(\iPart{a}{double} a)}!
    ....
\end{lstlisting}
};
% Annotation
\codeAnnotation{lText}    (3,2.5)   {library name}
\codeAnnotation{sText}   (-1,1)     {signature}
\codeAnnotation{nText}  (4.5,1.5)   {method name}
\codeAnnotation{rText}  (2.0,-0.51) {return type}
\codeAnnotation{aText}  (4.5,-0.51) {argument type}
% Arrows
\draw[->, annotation] (lText) -- (l);
\draw[->, annotation] (nText) -- (n);
\draw[->, annotation] (sText) -- (s);
\draw[->, annotation] (rText) -- (r);
\draw[->, annotation] (aText) -- (a);
\end{tikzpicture}
```

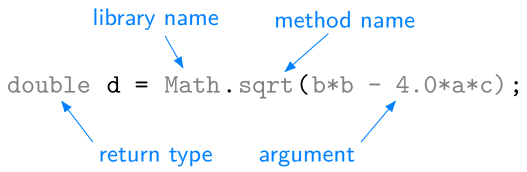


```
public class Math
    ....
    static double sqrt(double a)
    ....
```

2.5 Using a library method [1, p. 30]

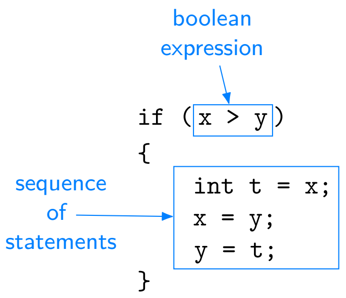
```
\begin{tikzpicture}[remember picture]
```

```
\codeBlock{%
\iPart{r}{double} d = \iPart{l}{Math}.\iPart{m}{sqrt}(\iPart{a}{b*b - 4.0*a*c)};
}
% Annotation
\codeAnnotation{lText}   (2, 1.125) {library name}
\codeAnnotation{mText} (4.5, 1.125) {method name}
\codeAnnotation{rText}   (2,-0.7)   {return type}
\codeAnnotation{aText}   (4,-0.7)   {argument}
% Arrows
\draw[->,annotation] (lText) -- (l);
\draw[->,annotation] (mText) -- (m);
\draw[->,annotation] (rText.north west) -- (r);
\draw[->,annotation] (aText.north east) -- (a);
\end{tikzpicture}
```



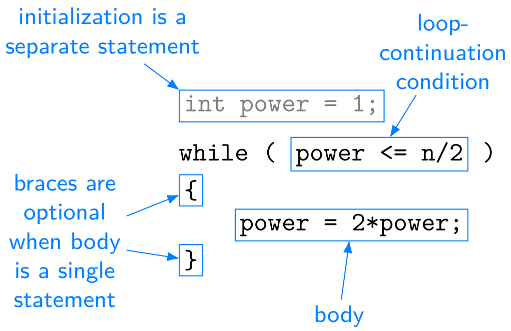
2.6 Anatomy of an `if` statement [1, p. 51]

```
\lstset{escapeinside={!}{!}}
\begin{tikzpicture}[remember picture]
%    {[on background layer]\draw[code grid debug] (-2.5,-0.5) grid (2.5,2.5);}
\node(code) [anatomy] at (0,0) {%
\begin{lstlisting}
if (!\cPart{e}{x > y}!)
{
    int t = x;!\mtPoint{tr}!
    x = y;
   !\mbPoint{bl}! y = t;!\extremPoint{br}!
}
\end{lstlisting}
};

\fitExtrem{b}{(tr) (bl) (br)}
% Annotation
\codeAnnotation{eText}  (1,3.5)  {boolean\\expression}
\codeAnnotation{bText} (-1,1.125)  {sequence\\of \extremPoint{bPoint}[0.75ex]\\statements}
% Arrow
\draw[->,annotation] (eText) -- (e);
\draw[->,annotation] (bPoint) -- (b);
\end{tikzpicture}
```



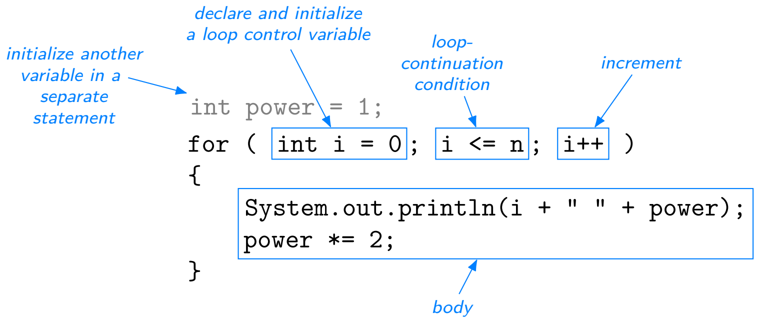
2.7 Anatomy of a `while` loop [1, p. 54]

```
\lstset{escapeinside={!}{!}}
\begin{tikzpicture}[remember picture]
%    {[on background layer]\draw[code grid debug] (-2.5,-0.5) grid (2.5,2.5);}
\node(code) [anatomy] at (0,0) {
\begin{lstlisting}
!\cPart{i}{\bgcode{int power = 1;}}\phantom{\rule[-2ex]{0.1ex}{0.1ex}}!
while ( !\cPart{c}{power <= n/2}! )
!\cPart{po}{\{}!
    !\cPart{b}{power = 2*power;}!
!\cPart{pc}{\}}!
\end{lstlisting}
};

% Annotation
\codeAnnotation{iText}  (-1,3.25) {initialization is a\\separate statement}
\codeAnnotation{cText} (3.5,3)    {loop-\\continuation\\condition}
\codeAnnotation{pText} (-1.5,0.5) {braces are\\optional\\when body\\is a single\\statement}
\codeAnnotation{bText} (2.125,-0.5) {body}
% Arrows
\draw[->,annotation] (iText) -- (i.north west);
\draw[->,annotation] (cText) -- (c);
\draw[->,annotation] (bText) -- (b);
\draw[->,annotation] (pText) -- (po);
\draw[->,annotation] (pText) -- (pc);
\end{tikzpicture}
```



2.8 Anatomy of a `for` loop [1, p. 59]

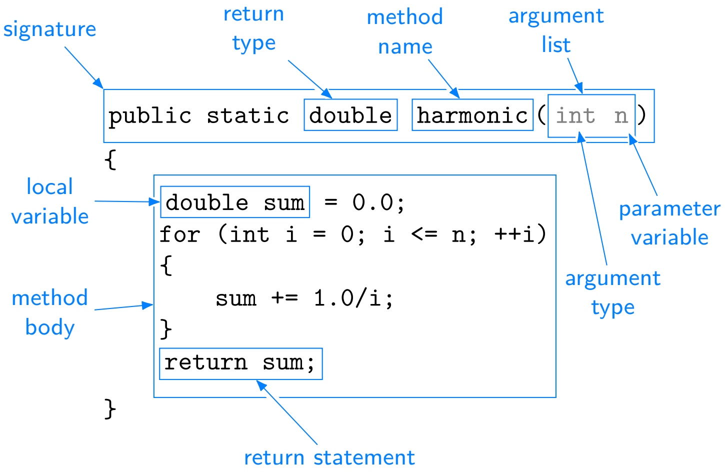
```
\lstset{escapeinside={!}{!}}
\begin{tikzpicture}[
     remember picture
    ,code annotation/.append style={%
        font=\sffamily\itshape\scriptsize
    }
]
    % {[on background layer]\draw[code grid debug] (-2.5,-0.5) grid (5.5,3.5);}
\node(code) [anatomy] at (0,0){%
\begin{lstlisting}
!\iPart{init}{\bgcode{int power = 1;}}!
for ( !\cPart{i}{int i = 0}!; !\cPart{c}{i <= n}!; !\cPart{u}{i++}! )
{
    !\mtPoint{left}!System.out.println(i + " " + power);!\mtPoint{right}!
    power *= 2;!\mbPoint{bottom}!
}
\end{lstlisting}
};
\fitExtrem{b}{(left) (right) (bottom)}
% Annotations
\codeAnnotation{initText} (-1.5,2.7)   {initialize another\\
                                        variable in a \extremPoint{initPoint}[0.75ex]\\
                                        separate\\statement}
\codeAnnotation{iText}       (1.2,3.5)   {declare and initialize\\
                                        a loop control variable}
\codeAnnotation{cText}     (3.5,3)     {loop-\\continuation\\condition}
\codeAnnotation{uText}       (6,3)     {increment}
\codeAnnotation{bText}     (3.5,-0.25) {body}
% arrows on the background
{[on background layer]
\draw[->,annotation] (initPoint) -- (init.north west);
\draw[->,annotation] (iText) -- (i);
\draw[->,annotation] (cText) -- (c);
\draw[->,annotation] (uText) -- (u);
\draw[->,annotation] (bText) -- (b);
}
\end{tikzpicture}
```



```
int power = 1;
for ( int i = 0; i <= n; i++ )
{
    System.out.println(i + " " + power);
    power *= 2;
}
```

2.9 Anatomy of a static method [1, p. 196]

```
\lstset{escapeinside={!}{!}}
\begin{tikzpicture}[remember picture]
    %{[on background layer]\draw[code grid debug] (-2.5,-0.5) grid (8.5,3.5);}
\node(code) [anatomy] at (0,0) {%
\begin{lstlisting}
!\cPart{s}{public static \cPart{rt}{double} \cPart{fn}{harmonic}(\cPart{al}{\iPart{at}{int} \iPart{pv}{n}})}!
{
    !\hmtPoint{left}\cPart{lv}{double sum}! = 0.0;
    for (int i = 0; i <= n; ++i)!\extremPoint{right}!
    {
        sum += 1.0/i;
    }
    !\cPart{rs}{return sum;}\dmbPoint{bottom}!
}
\end{lstlisting}
};

\fitExtrem{b}{(left) (right) (bottom)}

% Annotation
\codeAnnotation{sText}  (-0.7,5.25)  {signature}
\codeAnnotation{rtText}    (2,5.25)  {return\\type}
\codeAnnotation{fnText}  (  4,5.25)  {method\\name}
\codeAnnotation{alText}  (  6,5.25)  {argument\\list}
\codeAnnotation{atText} (6.75,1.75)  {argument\\type}
\codeAnnotation{pvText}  (7.5,2.70)  {parameter\\variable}
\codeAnnotation{lvText} (-0.7,3)     {local\\variable}
\codeAnnotation{bText}  (-0.7,1.5)   {method\\body}
\codeAnnotation{rsText}  (3,-0.4)    {return statement}
% Arrows
\draw[->,annotation] (sText) -- (s.north west);
\draw[->,annotation] (rtText) -- (rt);
\draw[->,annotation] (fnText) -- (fn);
\draw[->,annotation] (alText) -- (al);
\draw[->,annotation] (atText) -- (at);
\draw[->,annotation] (pvText) -- (pv);
\draw[->,annotation] (lvText) -- (lv.west);
\draw[->,annotation] (bText) -- (b);
\draw[->,annotation] (rsText) -- (rs);
\end{tikzpicture}
```



```
public static double harmonic( int n)
{
    double sum = 0.0;
    for (int i = 0; i <= n; ++i)
    {
        sum += 1.0/i;
    }
    return sum;
}
```

References

[1] Robert Sedgewick and Kevin Wayne. *Computer Science. An Interdisciplinary Approach.* Addison-Wesley, 2016.