ClassChart

```
<<constructor>>+ClassChart()
+addClass(class: Class)
+addEdge(edge: Edge)
+getClasses(): array of Class
+getEdges(): array Of Edge
```

1

+classChart

Edge

```
<<constructor>>+Edge(pHead: Class, pTail: Class)
+getHead(): Class
+getTail(): Class
```

* -edges

Association

```
-roleName: Text
-multiplicity: GZ
```
```
<<constructor>>+Association(pRoleName: Text, pMultiplicity: Text)
+getRoleName(): Text
+getMultiplicity(): GZ
```

Inheritance

* -edges

2 -head, tail

* -classes

Class

```
-name: Text
```
```
<<constructor>>+Class(pName: Text)
+getName(): Text
+getOperations(): array of Operation
+getAttributes(): array of Attributes
+getEdges(): array of Edge
+getParentClasses(): array of Class
+getAncestorClasses(): array of Class
+addOperation(operation: Operation)
+addAttribute(attribute: Attribute)
+addEdge(edge: Edge)
+isAbstract(): Boolean
```

```
-name: Text
```
```
<<constructor>>+Ar
+getName(): Text
+getType(): Type
```

* -argu

stereotype:
```
0 = normal
1 = constructor
2 = destructor
```

* -operations

Operation

```
-name: Text
-returnType: Type
-visibility: GZ
-abstract: Boolean
-stereotype: GZ
```
```
<<constructor>>+Operation(pName: Text, pReturnType: Type, pArguments: array of Argument, pVisibility: GZ)
+getName(): Text
+getReturnType(): Type
+getArguments(): array of Argument
+getVisibility(): GZ
+isAbstract(): Boolean
+getStereotype(): GZ
```

Result

Type

+umlName(): Text
+createFromCppName(pType: Text): Type

Parser

+Parser(begin: char*, end: char*)
+getResult(): Result

Private Operationen des Parsers sind dem Funktionsbaum z

-result

1

1

-type

-returnType 1

-type 1

1 -type

PrimitiveType

-typeEnum: GZ

<<constructor>>+PrimitiveType(pTypeEnum: GZ)
+umlName(): Text

ClassType

-name: Text

<<constructor>>+ClassType(pName: Text)
+umlName(): Text

typeEnum:
0 = Integer
1 = Float
2 = Boolean
3 = Text
4 = Character

Argument

---
gument(pName: Text, pType: Type)

uments

-command: Text

<<constructor>>
+getText(): Text

-attributes

*

Attribute

-name: Text
-type: Type
-visibility: GZ

<<constructor>>+Attribute(pName: Text, pType: GZ, pVisibility: GZ)
+getName(): Text
+getType(): Type
+getVisibility(): GZ

-condition: Text
-headControlled: Boolean

<<constructor>>+LoopBlock(pCondition: Text,
+getCondition(): Text
+getBody(): BlockSequence
+getHeadControlled(): Boolean

visibility:
0 = private
1 = public
2 = protected

StructureChart

-headline: Text

<<constructor>>+StructureChart(pHeadline: Text, pRoot: BlockSequence, pDeclarations: array of Declaration)
+getHeadline(): Text
+getRoot(): Text
+getDeclarations(): array of Declaration

+structureCharts

*

u entnehmen.

* -declarations

Declaration

-varName: Text

<<constructor>>+Declaration(pType: Type, pVarName: Text)
+getType(): Text
+getVarName(): Text

1 -root

BlockSequence

<<constructor>>+BlockSequence(pBlocks: array of
+getBlocks(): array of Block

-yes, no 2

-body 1

* -blocks

Block

<<destructor>>+~Block()

SimpleBlock

+SimpleBlock(pCommand: Text)

LoopBlock

Body: BlockSequence, pHeadControlled: Boolean`)

+values of sequenc

-expression: Text
-sequences: map<String, Block

<<constructor>>+SwitchBlock
+getExpression(): Text
+getSequences(): map<String,

IfElseBlock

-condition: Text

<<constructor>>+IfElseBlock(pCondition: Text, pYes: BlockSequence, pNo: BlockSequence)
+getCondition(): Text
+getYes(): BlockSequence
+getNo(): BlockSequence

StructureChartDrawer

+scene: QGraphicsScene
+chart: StructureChart
+top: GZ
+left: GZ
+initialWidth: GZ
+width: GZ

<<constructor>>+StructureChartDrawer()
+drawBody(group: QGraphicsItem, vector: array of Block)
+drawStructureChart(pChart: StructureChart): QGraphicsItem
+drawHead(group: QGraphicsItem)
+wrapText(inputItem: QGraphicsSimpleTextItem, maximumWidth: GZ)
+drawEmptySign(rect: QGraphicsRectItem, group: QGraphicsItem)
+drawLoopHeading(loopHeading: QGraphicsSimpleTextItem)
+drawSurroundingRect(group: QGraphicsItem)

ClassChartDrawer

+drawClassChart(classChart: ClassChart): QGraphicsItem
+visibilityToString(visibility: GZ): Text
+attributeToString(attribute: Attribute): Text
+operationToString(operation: Operation): Text
+drawClassBox(class: Class): QGraphicsItem
+calculateArrowLine(r1: Rect, r2: Rect): Line
+drawArrow(tail: QGraphicsItem, head: QGraphicsItem, tailsym: QGraphicsItem, headsym: QGraphicsItem): QGraphicsItem

---

SwitchBlock

:Sequence>

:(expression: Text, sequences: map<String, BlockSequence>)

. BlockSequence>

es *

Block)