SCHETT MATTHIAS

SEN-ÜBUNG 3

Inhaltsverzeichnis

Aufgabe 1

Lösungsidee

Die Object Klasse dient nur als Basisklasse und verfügt nur einen Konstruktor der leer ist und über einen rein virtuellen Destruktor[1]. Die Klasse GraphicObject erbt public von Object und verfügt über 4 virtuelle Methoden, 2 davon sind wieder rein virtuell[2].

[1] Dadurch wird Objekt zu einer abstrakten Klasse gemacht.

[2] Dadurch wird auch GraphicObject zu einer abstrakten Klasse

- void draw(); ist rein virtuell muss also in einer abgeleiteten Klasse überschrieben werden.
- TBox getBoundingBox() ist ebenfalls rein virtuell.[3]
- getPositionHelper() const; wird benutzt falls eine abgeleitete Klasse ihre Implementierung für das Ermitteln der Position verändern will - ist protected
- moveToHelper(TCoord const& position); gilt das gleiche wie bei getPositonHelper();

[3] Rein virtuell, da die Basisklasse nicht wissen kann, wie abgeleitete Klasse gezeichnet werden oder ihren Rahmen ermitteln.

Weiters gibt es moveTo(TCoord const& position) und getPositon(); diese beiden Methoden sind inline deklariert und rufen nur die Helper Methoden auf. Dadurch wird gewährleistet, dass es immer eine Standardimplementierung gibt und Kunde nicht vergessen, die virutellen Methoden zu überschreiben.

In den abgeleiteten Klassen gibt es nur Implementierugen für die virutellen Methoden draw() und getBoundingBox(), die restlichen Methoden werden übernommen.

Die Klasse Picture wiederum ist ein Spezialfall. Sie verfügt über einen std::vector als Member der GraphicObject* aufnehmen kann, dadurch kann sie alles aufnehmen dass von GraphicObject ableitet, also auch weitere Pictures. Die weiteren Funktionen rufen in Schleifen, die jeweilige Methode der im std::vector befindlichen Objekte auf.

Testfälle

Listing 1: Testausgabe

```
1 Line::draw() begins at (2,3) goes until (3,5)
2 Line::boundingbox((2,3),(3,5))
3
```

```
4  Rectangle::draw() has corner at(4,5) another corner at
       (9,8)
5  Rectangle::boundingbox((4,5),(9,8))
6
7  Circle::draw() middle at (10,11) with radius 15
8  Circle::boundingbox((10,11),(25,11))
9
10 Picture Test
11 Line::draw() begins at (2,3) goes until (3,5)
12 Rectangle::draw() has corner at(4,5) another corner at
       (9,8)
13 Circle::draw() middle at (10,11) with radius 15
14 Circle::draw() middle at (67,17) with radius 6
15 Picture::boundingbox((67,17),(73,17))
16
17 Picture in Picture Test
18 Line::draw() begins at (2,3) goes until (3,5)
19 Line::draw() begins at (2,3) goes until (3,5)
20 Rectangle::draw() has corner at(4,5) another corner at
       (9,8)
21 Circle::draw() middle at (10,11) with radius 15
22 Circle::draw() middle at (67,17) with radius 6
23 Picture::boundingbox((10,11),(25,11))
```

Anhang A
Aufgabe 1

Listing A.1: Typdefs

```
1  ////////////////////////////////////
2  // Workfile    : Types.h
3  // Author      : Matthias Schett
4  // Date        : 23-06-2013
5  // Description : used for common typedefs
6  // Remarks     : -
7  // Revision    : 0
8  ////////////////////////////////////
9
10 #ifndef TYPES_H
11 #define TYPES_H
12
13 #include <utility>
14
15 typedef int TUnit;
16 typedef std::pair<TUnit, TUnit> TCoord;
17 typedef std::pair<TCoord, TCoord> TBox;
18
19 #endif TYPES_H
```

Listing A.2: Object Header

```
1  ////////////////////////////////////
2  // Workfile    : Object.h
3  // Author      : Matthias Schett
4  // Date        : 23-06-2013
5  // Description : Object header
6  // Remarks     : -
7  // Revision    : 0
8  ////////////////////////////////////
9
10 #ifndef OBJ_H
11 #define OBJ_H
12
13 class Object {
14 public:
```

```
15      Object(){}
16      virtual ~Object() = 0;
17  };
18
19  inline Object::~Object(){}
20
21  #endif OBJ_H
```

Listing A.3: GraphicObject Header

```
1   ////////////////////////////////////
2   // Workfile    : GraphicObject.h
3   // Author      : Matthias Schett
4   // Date        : 23-06-2013
5   // Description : Graphic Object header
6   // Remarks     : -
7   // Revision    : 0
8   ////////////////////////////////////
9
10  #ifndef GRAPHOBJ_H
11  #define GRAPHOBJ_H
12
13  #include "object.h"
14  #include "Types.h"
15  #include <iostream>
16
17  class GraphicObject : public Object {
18
19  protected:
20      TCoord mPosition;
21
22  public:
23
24      virtual ~GraphicObject();
25      virtual void draw() = 0;
26      virtual TBox getBoundingBox() const = 0;
27
28      TCoord getPosition() const {
29          return getPositionHelper();
30      }
31
32      void moveTo(TCoord const &position){
33          moveToHelper(position);
34      }
35
36      void printTBox(TBox box);
37      void printTCoord(TCoord position);
38
39  protected:
40      GraphicObject();
```

```
41      virtual TCoord getPositionHelper() const;
42      virtual void moveToHelper(TCoord const &position);
43  };
44
45  #endif GRAPHOBJ_H
```

Listing A.4: GraphicObject Implementierung

```
1   ////////////////////////////////////
2   // Workfile    : GraphicObject.h
3   // Author      : Matthias Schett
4   // Date        : 23-06-2013
5   // Description : Graphic Object header
6   // Remarks     : -
7   // Revision    : 0
8   ////////////////////////////////////
9
10  #include "GraphicObject.h"
11
12
13  GraphicObject::GraphicObject() {
14
15  }
16
17
18  GraphicObject::~GraphicObject() {
19
20  }
21
22
23  void GraphicObject::moveToHelper(TCoord const &position){
24      mPosition = position;
25  }
26
27  TCoord GraphicObject::getPositionHelper() const{
28      return mPosition;
29  }
30
31  void GraphicObject::printTCoord(TCoord position){
32      std::cout << "(" << position.first << "," << position.second << ")";
33  }
34
35  void GraphicObject::printTBox(TBox box){
36      std::cout << "(";
37      printTCoord(box.first);
38      std::cout << ",";
39      printTCoord(box.second);
40      std::cout << ")";
41  }
```

Listing A.5: Line Header

```
1  ////////////////////////////////////
2  // Workfile    : Line.h
3  // Author      : Matthias Schett
4  // Date        : 23-06-2013
5  // Description : Line header
6  // Remarks     : -
7  // Revision    : 0
8  ////////////////////////////////////
9
10 #ifndef LINE_H
11 #define LINE_H
12
13 #include "GraphicObject.h"
14
15
16 class Line : public GraphicObject {
17
18 private:
19     TCoord mExtent;
20
21 public:
22     Line();
23     Line(TCoord mPos, TCoord mExt);
24     ~Line();
25     virtual TBox getBoundingBox() const;
26     virtual void draw();
27 };
28
29 #endif LINE_H
```

Listing A.6: Line Implementierung

```
1  ////////////////////////////////////
2  // Workfile    : Line.cpp
3  // Author      : Matthias Schett
4  // Date        : 23-06-2013
5  // Description : Line implementation
6  // Remarks     : -
7  // Revision    : 0
8  ////////////////////////////////////
9
10 #include "Line.h"
11
12
13 Line::Line() {
14 }
15
16 Line::Line(TCoord mPos, TCoord mExt){
17     mPosition = mPos;
```

```
18      mExtent = mExt;
19  }
20
21  Line::~Line() {
22  }
23
24  TBox Line::getBoundingBox() const {
25      return TBox(mPosition, mExtent);
26  }
27
28  void Line::draw(){
29      std::cout << "Line::draw() begins at ";
30      printTCoord(mPosition);
31      std::cout << " goes until ";
32      printTCoord(mExtent);
33      std::cout << std::endl;
34  }
```

Listing A.7: Rectangle Header

```
1   ////////////////////////////////////
2   // Workfile    : Rectangle.h
3   // Author      : Matthias Schett
4   // Date        : 23-06-2013
5   // Description : Circle header
6   // Remarks     : -
7   // Revision    : 0
8   ////////////////////////////////////
9
10  #ifndef RECTANGLE_H
11  #define RECTANGLE_H
12
13  #include "GraphicObject.h"
14  class Rectangle : public GraphicObject {
15
16  private:
17      TCoord mExtent;
18
19  public:
20      Rectangle();
21      Rectangle(TCoord mPos, TCoord mExt);
22      ~Rectangle();
23      virtual TBox getBoundingBox() const;
24      virtual void draw();
25  };
26
27  #endif RECTANGLE_H
```

Listing A.8: Rectangle Implementierung

```
1   ////////////////////////////////////
```

```
2  // Workfile    : Rectangle.cpp
3  // Author      : Matthias Schett
4  // Date        : 23-06-2013
5  // Description : Rectangle implementation
6  // Remarks     : -
7  // Revision    : 0
8  ////////////////////////////////////
9
10 #include "Rectangle.h"
11
12 Rectangle::Rectangle(){
13 }
14
15 Rectangle::Rectangle(TCoord mPos, TCoord mExt){
16     mPosition = mPos;
17     mExtent = mExt;
18 }
19
20 Rectangle::~Rectangle(){
21 }
22
23 TBox Rectangle::getBoundingBox() const {
24     return TBox(mPosition, mExtent);
25 }
26
27 void Rectangle::draw(){
28     std::cout << "Rectangle::draw() has corner at";
29     printTCoord(mPosition);
30     std::cout << " another corner at ";
31     printTCoord(mExtent);
32     std::cout << std::endl;
33 }
```

Listing A.9: Circle Header

```
1  ////////////////////////////////////
2  // Workfile    : Circle.h
3  // Author      : Matthias Schett
4  // Date        : 23-06-2013
5  // Description : Circle header
6  // Remarks     : -
7  // Revision    : 0
8  ////////////////////////////////////
9
10 #ifndef CIRCLE_H
11 #define CIRCLE_H
12
13 #include "GraphicObject.h"
14
15 class Circle :public GraphicObject {
```

```
16
17 private:
18     TUnit mRadius;
19
20 public:
21     Circle();
22     Circle(TCoord mPos, TUnit radius);
23     ~Circle();
24     virtual void draw();
25     virtual TBox getBoundingBox() const;
26 };
27
28 #endif CIRCLE_H
```

Listing A.10: Circle Implementierung

```
1  ////////////////////////////////////
2  // Workfile    : Circle.cpp
3  // Author      : Matthias Schett
4  // Date        : 23-06-2013
5  // Description : Circle implementation
6  // Remarks     : -
7  // Revision    : 0
8  ////////////////////////////////////
9
10 #include "Circle.h"
11
12
13 Circle::Circle() {
14 }
15
16 Circle::Circle(TCoord mPos, TUnit radius){
17     mPosition = mPos;
18     mRadius = radius;
19 }
20
21 Circle::~Circle() {
22 }
23
24 TBox Circle::getBoundingBox() const {
25     TCoord corner(mRadius + mPosition.first, mPosition.second);
26     return TBox(mPosition, corner);
27 }
28
29 void Circle::draw(){
30     std::cout << "Circle::draw() middle at ";
31     printTCoord(mPosition);
32     std::cout << " with radius " << mRadius << std::endl;
33 }
```

Listing A.11: Picture Header

```
1  ////////////////////////////////////
2  // Workfile    : Picture.h
3  // Author      : Matthias Schett
4  // Date        : 23-06-2013
5  // Description : Picture header
6  // Remarks     : -
7  // Revision    : 0
8  ////////////////////////////////////
9
10 #ifndef PICTURE_H
11 #define PICTURE_H
12
13 #include "GraphicObject.h"
14 #include <vector>
15
16 class Picture : public GraphicObject {
17
18     std::vector<GraphicObject *> mElements;
19
20 public:
21     Picture();
22     ~Picture();
23     void Add(GraphicObject* Element);
24     virtual void draw();
25     virtual TBox getBoundingBox() const;
26     size_t getNumOfElements() const;
27
28 protected:
29     virtual void moveToHelper(TCoord const &position);
30 };
31
32 #endif PICTURE_H
```

Listing A.12: Picture Implementierung

```
1  #include "Picture.h"
2  #include <algorithm>
3
4  Picture::Picture() {
5  }
6
7
8  Picture::~Picture() {
9  }
10
11 void Picture::Add(GraphicObject* Element){
12     mElements.push_back(Element);
13 }
14
```

```
15 void Picture::draw(){
16     std::vector<GraphicObject *>::iterator begin = mElements.begin();
17 
18     while(begin != mElements.end()){
19         (*begin)->draw();
20         begin++;
21     }
22 }
23 
24 TBox Picture::getBoundingBox() const {
25     std::vector<TBox> boundingBoxes;
26     std::vector<GraphicObject *>::const_iterator begin = mElements.begin();
27     while(begin != mElements.end()){
28         boundingBoxes.push_back((*begin)->getBoundingBox());
29         begin++;
30     }
31 
32     return *std::max_element(boundingBoxes.begin(), boundingBoxes.end());
33 }
34 
35 size_t Picture::getNumOfElements() const{
36     return mElements.size();
37 }
38 
39 void Picture::moveToHelper(TCoord const &position) {
40     std::vector<GraphicObject *>::iterator begin = mElements.begin();
41     while (begin != mElements.end()) {
42         (*begin)->moveTo(TCoord( ( (*begin)->getPosition().first - mPosition.first +
            position.first ),
43             ((*begin)->getPosition().second - mPosition.second + position.second) ));
44         begin++;
45     }
46     mPosition.first  = position.first;
47     mPosition.second = position.second;
48 }
```

Listing A.13: Testtreiber

```
1  ////////////////////////////////////
2  // Workfile    : Main.cpp
3  // Author      : Matthias Schett
4  // Date        : 23-06-2013
5  // Description : Test driver
6  // Remarks     : -
7  // Revision    : 0
8  ////////////////////////////////////
9 
10 #include <iostream>
11 #include "Rectangle.h"
12 #include "Circle.h"
```

```
13 #include "Line.h"
14 #include "Picture.h"
15
16 using namespace std;
17
18 int main(){
19
20     TCoord linePos(2, 3);
21     TCoord lineExt(3, 5);
22     Line l1(linePos, lineExt);
23     l1.draw();
24     cout << "Line::boundingbox";
25     l1.printTBox(l1.getBoundingBox());
26
27     cout << endl << endl;
28
29     TCoord rectPos(4,5);
30     TCoord rectExtent(9,8);
31     Rectangle r1(rectPos, rectExtent);
32     r1.draw();
33     cout << "Rectangle::boundingbox";
34     r1.printTBox(r1.getBoundingBox());
35
36     cout << endl << endl;
37
38     TCoord circlePos(10, 11);
39     TUnit circleRadius = 15;
40     Circle c1(circlePos, circleRadius);
41     c1.draw();
42     cout << "Circle::boundingbox";
43     c1.printTBox(c1.getBoundingBox());
44
45     GraphicObject* g1 = &l1;
46     GraphicObject* g2 = &r1;
47     GraphicObject* g3 = &c1;
48
49     TCoord circlePos1(67, 17);
50     TUnit circleRadius1 = 6;
51     Circle *c2 = new Circle(circlePos1, circleRadius1);
52
53     cout << endl << endl;
54     cout << "Picture Test" << endl;
55
56
57     Picture p1;
58     p1.Add(g1);
59     p1.Add(g2);
60     p1.Add(g3);
61     p1.Add(c2);
```

```
62
63      p1.draw();
64      cout << "Picture::boundingbox";
65      p1.printTBox(p1.getBoundingBox());
66
67      cout << endl << endl;
68      cout << "Picture in Picture Test" << endl;
69
70
71      Picture p2;
72      GraphicObject* g4 = &p1;
73      p2.Add(g1);
74      p2.Add(g4);
75
76      p2.draw();
77      cout << "Picture::boundingbox";
78      p2.printTBox(c1.getBoundingBox());
79
80
81
82
83      delete c2;
84
85      cin.get();
86      return 0;
87  }
```