Software Engeneering Analysis Sientific Report

Kendra Birringer (1229372)
Nader Cacace (1208115)
Steffen Hanzlik (1207417)
Marco Peluso (1228849)
Svetozar Stojanovic (1262287)

Frankfurt University of Applied Sciences

1 Exercise 2

1.1 Fahrzeug

```
1  void Fahrzeug::setName(const char *n)
2  {
3      if (name != nullptr) {
4       delete name;
5       name = nullptr; // handling dangling pointer to freed memory
6      }
7
8      if (n != nullptr) { // handling empty pointer as parameter,
           preventing strlen from crashing
9       name = new char[strlen(n)+1];
10      strcpy(name, n);
11     }
12 }
13
14
15
16 Fahrzeug& operator=(const Fahrzeug &other) {
17
18 }
```

Figure 1 Fahrzeug Program

2 Exercise 3

2.1 Ebook Headerfile

```
1  #ifndef _EBOOK_H_
2  #define _EBOOK_H_
3  #include <string>
```

```
4  #include <iostream>
5  using namespace std;
6  class EBook
7  {
8  private:
9     string title, content;
10 public:
11    EBook() :title{ "" }, content{ "" } {};
12    EBook(string title, string content) :title{ title }, content{ content
          }{};
13    void SetTitle(string title);
14    string GetTitle()const;
15    void SetContent(string content);
16    string GetContent()const;
17    void print();
18    friend ostream &operator<<(ostream &output, const EBook& book);
19 };
20
21
22 #endif // !_EBOOK_H_
```

Figure 2 Header of Ebook Program

2.2 Ebook Class

```
#include "eBook.h"
#include <iostream>
using namespace std;
void EBook::SetTitle(string title)
{
   if (title!="")
   {
      this->title = title;
   }
   else
   {
      cout << "Title not set!" << endl;
   }
}

string EBook::GetTitle() const
{
   return this->title;
}

void EBook::SetContent(string content)
{
   if (content != "")
   {
      this->content = content;
   }
   else
   {
      cout << "Content not set!" << endl;
   }
}

string EBook::GetContent() const
{
   return this->content;
}

void EBook::print()
{
   cout << "Title: " << this->title << '\n';
   cout << "Content: " << '\n' <<this->content << '\n';

}

ostream & operator<<(ostream & output, const EBook & book)
{
```

```
47      output << "Title: " << book.title << '\n' << "Content: " <<
            book.content << '\n';
48      //or alternatively
49
50      //output<<book.print()<<'\n';
51      return output;
52  }
```

Figure 3 Ebook class

2.3 Main Class

```
#include <iostream>
#include "eBook.h"
int main() {
   EBook book("Brown Fox", "The quick brown fox jumps over the lazy
       dog.");
   std::cout << book;


   return 0;
```

Figure 4 Main class

3 Exercise 4

3.1 Box Headerfile

```
1   #ifndef _BOX_H_
2   #define _BOX_H_
3   class Box
4   {
5   private:
6      double xMin, xMax, yMin, yMax;
7   public:
8
9      Box():xMin{ 0.0 }, xMax{ 0.0 }, yMin{ 0.0 }, yMax{ 0.0 }{}
10     double GetXMin() const { return xMin; }
11     double GetXMax() const { return xMax; }
12     double GetYMin() const { return yMin; }
13     double GetYMax() const { return yMax; }
14     void SetXMax(double val);
15     void SetXMin(double val);
16     void SetYMin(double val);
17     void SetYMax(double val);
18     friend Box operator+(Box left, Box right);
```

```
19    void print();
20 };
21
22 #endif // !_BOX_H_
```

Figure 5 Header of Box

3.2 Circle Headerfile

```
1  #ifndef _CIRCLE_H_
2  #define _CIRLE_H_
3  #include "Form.h"
4  class Circle: public Form
5  {
6  private:
7     double radius;
8  public:
9     Circle()
10    {
11       Form();
12       this->box.SetXMax(0.0);
13       this->box.SetXMin(0.0);
14       this->box.SetYMax(0.0);
15       this->box.SetYMin(0.0);
16    }
17    Circle(double rad):radius{rad}{}
18
19    //MOVE FOR CIRCLE
20    void Move(double dX, double dY);
21    void SetUpBox();
22 private:
23    void MoveBox(double dX = 0, double dY = 0);
24
25 };
26
27 #endif // !_CIRCLE_H_
```

Figure 6 Header of Circle

3.3 Form Headerfile

```
1  #ifndef _FORM_H_
2  #define _FORM_H_
3  #include "Box.h"
4  class Form
5  {
6  private:
```

```
7     double xCenter, yCenter;
8  protected:
9     Box box;
10
11 public:
12    Form()
13    {
14       this->xCenter = 0.0;
15       this->yCenter = 0.0;
16    }
17
18    void Move(double dX, double dY) {
19       this->xCenter += dX;
20       this->yCenter += dY;
21
22    }
23    Box & GetBoxRef() { //how to get const ref???
24       return box;
25    }
26 };
27
28 #endif // !_FORM_H_
```

Figure 7 Header of Form

3.4 Rectangle Headerfile

```
1  #ifndef _RECTANGLE_H_
2  #define _RECTANGLE_H_
3  #include "Form.h"
4  class Rectangle: public Form
5  {
6  private:
7     double width, height;
8
9  public:
10    Rectangle()
11    {
12       Form();
13       this->box.SetXMax(0.0);
14       this->box.SetXMin(0.0);
15       this->box.SetYMax(0.0);
16       this->box.SetYMin(0.0);
17
18    }
19    Rectangle(double h, double w):height{h},width{w}{}
20    //MOVE FOR RECT
21    void Move(double dX, double dY);
22    void SetUpBox();
```

```
23  private:
24     void MoveBox(double dX = 0, double dY = 0);
25
26  };
27
28
29
30  #endif // !_RECTANGLE_H_
```

Figure 8 Header of Rectangle

3.5 Box Class

```
1   #include "Box.h"
2   #include <iostream>
3   #include <algorithm>
4   using namespace std;
5
6   void Box::SetXMax(double val)
7   {
8      this->xMax = val;
9
10  }
11
12  void Box::SetXMin(double val)
13  {
14     this->xMin = val;
15
16  }
17
18  void Box::SetYMin(double val)
19  {
20     this->yMin = val;
21  }
22
23  void Box::SetYMax(double val)
24  {
25     this->yMax = val;
26  }
27
28  void Box::print()
29  {
30     cout << "xMax: " << xMax << endl;
31     cout << "xMin: " << xMin << endl;
32     cout << "yMax: " << yMax << endl;
33     cout << "yMin: " << yMin << endl;
34
35  }
36
```

```
37 Box operator+(Box left, Box right)
38 {
39    Box newLeft, newRight;
40    if (left.GetXMax()>right.GetXMax())
41    {
42       newLeft = right;
43       newRight = left;
44    }
45    else
46    {
47       newLeft = left;
48       newRight = right;
49    }
50    Box result;
51    if (right.GetXMin() < left.GetXMax() && right.GetYMin() <
          left.GetYMax()) //check if the boxes collide
52    {
53
54       result.SetXMin(min(newLeft.GetXMin(), newRight.GetXMin()));
55       result.SetXMax(max(newLeft.GetXMax(), newRight.GetXMax()));
56       result.SetYMin(min(newLeft.GetYMin(), newRight.GetYMin()));
57       result.SetYMax(max(newLeft.GetYMax(), newRight.GetYMax()));
58       return result;
59    }
60    else
61    {
62       cout << "The boxes of these two objects don't collide." << '\n';
63    }
64
65 }
```

Figure 9 Box Class

3.6 Circle Class

```
1  #include "Circle.h"
2
3  void Circle::SetUpBox()
4  {
5     this->box.SetXMax(this->radius);
6
7     this->box.SetXMin(-this->radius);
8
9     this->box.SetYMax(this->radius);
10
11    this->box.SetYMin(-this->radius);
12 }
13
14 void Circle::Move(double dX, double dY)
```

```
{
   Form::Move(dX, dY);
   MoveBox(dX, dY);
}

void Circle::MoveBox(double dX, double dY)
{
   this->box.SetXMax(box.GetXMax() + dX);

   this->box.SetXMin(box.GetXMin() + dX);

   this->box.SetYMax(box.GetYMax() + dY);

   this->box.SetYMin(box.GetYMin() + dY);
}
```

Figure 10 Circle Class

3.7 Rectangle Class

```
#include "Rectangle.h"

void Rectangle::Move(double dX, double dY)
{
   Form::Move(dX, dY);
   MoveBox(dX, dY);
}

void Rectangle::MoveBox(double dX, double dY)
{
   this->box.SetXMax(box.GetXMax()+dX);

   this->box.SetXMin(box.GetXMin()+dX);

   this->box.SetYMax(box.GetYMax()+dY);

   this->box.SetYMin(box.GetYMin()+dY);
}

void Rectangle::SetUpBox()
{

   this->box.SetXMax(width / 2);

   this->box.SetXMin(-width / 2);

   this->box.SetYMax(height / 2);

   this->box.SetYMin(-height / 2);
```

```
}
```

Figure 11 Rectangle Class

3.8 Main Class

```
#include <iostream>
#include <string>
#include "Circle.h"
#include "Rectangle.h"
using namespace std;
bool InputIsCircle(string); //checks if the user typed 'circle', returns bool
bool InputIsRect(string); //checks if the user typed 'rectangle', returns bool
Circle* CircleCreator(bool isTrue); //asks for needed values and calls circle constructor
Rectangle* RectCreator(bool isTrue);//asks for needed values and calls rectangle constructor
Box AddBoxes(Circle* c1, Circle* c2, Rectangle* r1, Rectangle* r2);
int main() {

   double movX, movY; //arguments for Move(...) function

   string prompt="";

   std::cout << "_____" << endl;

   Circle *circle1=NULL;
   Rectangle *rect1=NULL;

   cout << "Enter first form (rectangle or circle): ";
   cin >> prompt;

   if (InputIsCircle(prompt))
   {
      circle1 = CircleCreator(InputIsCircle(prompt));
      circle1->GetBoxRef().print();

      cout << "Move Circle in X direction for: ";
      cin >> movX;
      cout << "Move Circle in Y direction for: ";
      cin >> movY;

      circle1->Move(movX, movY);
      cout << "After Move is called: " << endl;
      circle1->GetBoxRef().print();

   }
```

```
40      else if (InputIsRect(prompt))
41      {
42         rect1 = RectCreator(InputIsRect(prompt));
43         rect1->GetBoxRef().print();
44
45         cout << "Move Rectangle in X direction for: ";
46         cin >> movX;
47         cout << "Move Rectangle in Y direction for: ";
48         cin >> movY;
49
50         rect1->Move(movX, movY);
51         cout << "After Move is called: " << endl;
52         rect1->GetBoxRef().print();
53      }
54
55      Circle *circle2 = NULL;
56      Rectangle *rect2 = NULL;
57      cout << "Enter second form (rectangle or circle): ";
58      cin >> prompt;
59      if (InputIsCircle(prompt))
60      {
61         circle2 = CircleCreator(InputIsCircle(prompt));
62         circle2->GetBoxRef().print();
63
64         cout << "Move Circle in X direction for: ";
65         cin >> movX;
66         cout << "Move Circle in Y direction for: ";
67         cin >> movY;
68
69         circle2->Move(movX, movY);
70         cout << "After Move is called: " << endl;
71         circle2->GetBoxRef().print();
72
73      }
74      else if (InputIsRect(prompt))
75      {
76         rect2 = RectCreator(InputIsRect(prompt));
77         rect2->GetBoxRef().print();
78
79         cout << "Move Rectangle in X direction for: ";
80         cin >> movX;
81         cout << "Move Rectangle in Y direction for: ";
82         cin >> movY;
83
84         rect2->Move(movX, movY);
85         cout << "After Move is called: " << endl;
86         rect2->GetBoxRef().print();
87
88      }
89
```

```
90
91
92      //ADD BOUNDING BOXES AND PRODUCE NEW ONE AS SUM
93
94
95
96      Box boundingBox;
97
98      cout << "Bounding Box: " << endl;
99      boundingBox = AddBoxes(circle1, circle2, rect1, rect2);
100     if (!(boundingBox.GetXMax()==0.0 && boundingBox.GetXMin()==0.0 &&
            boundingBox.GetYMin()==0.0 && boundingBox.GetYMax()==0.0))
101     {
102        boundingBox.print();
103     }
104
105     cout << "----------------------------------------" << endl;
106
107     delete circle1, rect1, circle2, rect2;
108
109     return 0;
110  }
111  Box AddBoxes(Circle* c1, Circle* c2, Rectangle* r1, Rectangle* r2) {
112
113     Box result;
114     if (c1==NULL && c2==NULL)
115     {
116        result = r1->GetBoxRef() + r2->GetBoxRef();
117        return result;
118     }
119     else if (c1==NULL && r2==NULL)
120     {
121        result = r1->GetBoxRef()+ c2->GetBoxRef();
122        return result;
123     }
124     else if (r1==NULL && c2==NULL)
125     {
126        result = c1->GetBoxRef()+ r2->GetBoxRef();
127        return result;
128     }
129     else if (r1==NULL && r2==NULL)
130     {
131        result = c1->GetBoxRef() + c2->GetBoxRef();
132        return result;
133     }
134  }
135  bool InputIsCircle(string prompt) {
136     string circle = "circle";
137     bool result = false;
138     if (prompt.compare(circle) == 0)
```

```
139     {
140        result = true;
141
142     }
143     else
144     {
145        return result;
146     }
147  }
148  bool InputIsRect(string prompt) {
149     string rect = "rectangle";
150     bool result = false;
151     if (prompt.compare(rect) == 0)
152     {
153        result = true;
154
155     }
156     else
157     {
158        return result;
159     }
160  }
161  Circle* CircleCreator(bool isTrue) {
162     if (isTrue)
163     {
164        double rad;
165
166        cout << "Enter radius: ";
167        cin >> rad;
168        Circle *circle=new Circle(rad);
169        circle->SetUpBox();
170        return circle;
171     }
172     else
173     {
174        return NULL;
175     }
176
177  }
178  Rectangle* RectCreator(bool isTrue) {
179     if (isTrue)
180     {
181        double h, w;
182        cout << "Enter height: ";
183        cin >> h;
184        cout << "Enter width: ";
185        cin >> w;
186        Rectangle *rect=new Rectangle(h, w);
187        rect->SetUpBox();
188        return rect;
```

```
189     }
190     else
191     {
192        return NULL;
193     }
194  }
```

Figure 12 Main Class

4 Foundations

5 Conclusion