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& Fahrzeug::operator=(const Fahrzeug &other) {
17    delete[] name;
18
19    if (other.name != nullptr) { // handling empty pointer as parameter,
          preventing strlen from crashing
20       name = new char[strlen(other.name) + 1];
21       strcpy(name, other.name);
22    }
23    return *this;
24 }
```

Figure 1 Fahrzeug Program

2 Exercise 3

2.1 EBook Headerfile

```
1  #ifndef _EBOOK_H_
2  #define _EBOOK_H_
3
4  #include <string>
5  #include <iostream>
6
7  using namespace std;
8
9  class EBook {
10 private:
11    string title, content;
12 public:
13    EBook();
14    EBook(string title, string content);
15    void setTitle(string title);
16    string getTitle() const;
17    void setContent(string content);
18    string getContent() const;
19    void print() const;
20    friend ostream &operator<<(ostream &output, const EBook &book);
21 };
22
23 #endif
```

Figure 2 Header of EBook Program

In the header file the declaration of the private and the public members of the Ebook class takes place.
The private members are two Strings named 'title' and 'content'. The public members of the class EBook are the default constructor and an overloaded constructor with the parameters title and content.
Then we also declared the public 'getter' and 'setter' methods. Furthermore we need a print method and we must overload the &operator<< method.

2.2 Implementation of the Ebook Class

```
1  #include "eBook.h"
2  #include <iostream>
3
4  using namespace std;
5
6  EBook::EBook() : title(""), content("") {};
7
8  EBook::EBook(string title, string content) : title(title),
       content(content) {};
9
10 void EBook::setTitle(string title) {
11    if (title != "") {
12       this->title = title;
```

```
13     } else {
14        cout << "Title not set!" << endl;
15     }
16  }
17
18  string EBook::getTitle() const {
19     return this->title;
20  }
21
22  void EBook::setContent(string content) {
23     if (content != "") {
24        this->content = content;
25     } else {
26        cout << "Content not set!" << endl;
27     }
28  }
29
30  string EBook::getContent() const {
31     return this->content;
32  }
33
34  void EBook::print() const {
35     cout << "Title: " << this->title << '\n';
36     cout << "Content: " << this->content << '\n';
37  }
38
39  ostream & operator<<(ostream &output, const EBook &book) {
40     book.print();
41     return output;
42  }
```

Figure 3 Implementation of Ebook class

In the Ebook.cpp file we implemented the declared methods of Ebook.h.
First we implemented the standard constructor initializing the member variables 'title' and 'content' using an initializer list.
Furthermore we implemented a overloaded constructor Ebook with the arguments 'title' and 'content' and initialized the 'title' and 'content' with the passed arguments 'title' and 'content'.

Then we implemented the 'getter' and 'setter' methods.(You knnow how they work)
Also we implemented the 'print' method. This method is for printing the 'title' and the 'content' and we overloaded the &operater<< function.

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 of Ebook program

The main method executes the Ebook Class.

3 Exercise 4

3.1 Box Headerfile

```
1  #ifndef _BOX_H_
2  #define _BOX_H_
3
4  class Box {
5  private:
6     double xMin, xMax, yMin, yMax;
7  public:
8
9     Box();
10    double getXMin() const;
11    double getXMax() const;
12    double getYMin() const;
13    double getYMax() const;
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() const;
20 };
21 #endif
```

Figure 5 Header of Box

In the header file the declaration of the private and the public members of the Box class takes place.
The private members are four double named 'xMin', 'xMax', 'yMin' and 'yMax'. The public members of the class Box are the default constructor.
Then we also declared the public 'getter' and 'setter' methods. Furthermore we need also a 'print' method and we musted overload the $+operator <<$ method.

3.2 Circle Headerfile

```
1  #ifndef _CIRCLE_H_
2  #define _CIRCLE_H_
```

```
3
4  #include "Form.h"
5
6  class Circle : public Form
7  {
8  private:
9     double radius;
10 public:
11    Circle();
12    Circle(double rad);
13    void move(double dX, double dY);
14    void setUpBox();
15 private:
16    void moveBox(double dX = 0, double dY = 0);
17 };
18 #endif
```

Figure 6 Header of Circle

In the header file the declaration of the private and the public members of the Circle class takes place.
The private members are one double with the name 'radius'. The public members of the class Circle are the default constructor initialized with the constructor of Form and intialize the x and y coordinates of box.
Then we overloaded the Circle constructor with the parameter 'rad' and initialize radius with rad.
Also we need a move method with the arguments 'dX' and 'dY', a setUpBox method and a private moveBox method with the arguments 'dX' and 'dY.

3.3 Form Headerfile

```
1  #ifndef _FORM_H_
2  #define _FORM_H_
3
4  #include "Box.h"
5
6  class Form {
7  private:
8     double xCenter, yCenter;
9  protected:
10    Box box;
11
12 public:
13    Form();
14    void move(double dX, double dY);
15
16    //how to get const ref???
17    Box &getBoxRef();
18 };
19 #endif
```

Figure 7 Header of Form

In the header file the declaration of the private and the public members of the Form class takes place.
The private members are two double with the name 'xCenter' and 'yCenter'. The protected member is a Object Box with the name 'box'. The public members of the class Form are the default constructor.
Also we need a move method with the arguments 'dX' and 'dY' and a method getBoxRef that returns a reference to the attribute Box.

3.4 Rectangle Headerfile

```
1  #ifndef _RECTANGLE_H_
2  #define _RECTANGLE_H_
3
4  #include "Form.h"
5
6  class Rectangle: public Form {
7  private:
8     double width, height;
9
10 public:
11    Rectangle();
12    Rectangle(double h, double w);
13    //MOVE FOR RECT
14    void move(double dX, double dY);
15    void setUpBox();
16 private:
17    void moveBox(double dX = 0, double dY = 0);
18
19 };
20 #endif
```

Figure 8 Header of Rectangle

In the header file the declaration of the private and the public members of the Form class takes place.
The private members are two double with the name 'xCenter' and 'yCenter'. The protected member is a Object Box with the name 'box'. The public members of the class Form are the default constructor.
Also we need a move method with the arguments 'dX' and 'dY' and a method getBoxRef that returns a reference to the attribute Box.

3.5 Box Class

```
1  #include "Box.h"
2  #include <iostream>
```

```
3  #include <algorithm>
4
5  using namespace std;
6
7  Box::Box() : xMin(0.0), xMax(0.0), yMin(0.0), yMax(0.0) {
8
9  }
10
11 double Box::getXMin() const {
12    return xMin;
13 }
14
15 void Box::setXMax(double val) {
16    this->xMax = val;
17 }
18
19 double Box::getXMax() const {
20    return xMax;
21 }
22
23 void Box::setXMin(double val) {
24    this->xMin = val;
25 }
26
27 double Box::getYMin() const {
28    return yMin;
29 }
30
31 void Box::setYMin(double val) {
32    this->yMin = val;
33 }
34
35 double Box::getYMax() const {
36    return yMax;
37 }
38
39 void Box::setYMax(double val) {
40    this->yMax = val;
41 }
42
43 void Box::print() const {
44    cout << "xMax: " << xMax << endl;
45    cout << "xMin: " << xMin << endl;
46    cout << "yMax: " << yMax << endl;
47    cout << "yMin: " << yMin << endl;
48 }
49
50 Box operator+(Box left, Box right) {
51    Box newLeft, newRight;
52    if (left.getXMax() > right.getXMax()) {
```

```
53        newLeft = right;
54        newRight = left;
55     } else {
56        newLeft = left;
57        newRight = right;
58     }
59
60     Box result;
61
62     //check if the boxes collide
63     if (right.getXMin() < left.getXMax() && right.getYMin() <
           left.getYMax()) {
64        result.setXMin(min(newLeft.getXMin(), newRight.getXMin()));
65        result.setXMax(max(newLeft.getXMax(), newRight.getXMax()));
66        result.setYMin(min(newLeft.getYMin(), newRight.getYMin()));
67        result.setYMax(max(newLeft.getYMax(), newRight.getYMax()));
68        return result;
69     } else {
70        cout << "The boxes of these two objects don't collide." << '\n';
71     }
72
73  }
```

Figure 9 Box Class

In the Box class we implemented getter and setter Methods. Also we implemented the print method to show the Min and Max coordinate. Then we overload the operator+ method.

3.6 Circle Class

```
1  #include "Circle.h"
2
3  Circle::Circle() : radius(0.0) {
4     Form();
5     this->box.setXMax(0.0);
6     this->box.setXMin(0.0);
7     this->box.setYMax(0.0);
8     this->box.setYMin(0.0);
9  }
10
11 Circle::Circle(double rad) : radius(rad) {
12
13 }
14
15 void Circle::setUpBox() {
16    this->box.setXMax(this->radius);
17    this->box.setXMin(-this->radius);
18    this->box.setYMax(this->radius);
19    this->box.setYMin(-this->radius);
```

```
}

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"

Rectangle::Rectangle() : width(0.0), height(0.0) {
   Form();
   this->box.setXMax(0.0);
   this->box.setXMin(0.0);
   this->box.setYMax(0.0);
   this->box.setYMin(0.0);
}

Rectangle::Rectangle(double h, double w) : width(w), height(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);
```

```
30     this->box.setXMin(-width / 2);
31     this->box.setYMax(height / 2);
32     this->box.setYMin(-height / 2);
33 }
```

Figure 11 Rectangle Class

3.8 Main Class

```
1  #include <iostream>
2  #include <string>
3  #include "Circle.h"
4  #include "Rectangle.h"
5
6  using namespace std;
7
8  //checks if the user typed 'circle', returns bool
9  bool inputIsCircle(string);
10
11 //checks if the user typed 'rectangle', returns bool
12 bool inputIsRect(string);
13
14 //asks for needed values and calls circle constructor
15 Circle* circleCreator(bool isTrue);
16
17 //asks for needed values and calls rectangle constructor
18 Rectangle* rectCreator(bool isTrue);
19 Box addBoxes(Circle* c1, Circle* c2, Rectangle* r1, Rectangle* r2);
20
21 int main() {
22
23    //arguments for move(...) function
24    double movX, movY;
25
26    string prompt = "";
27
28    std::cout << "_______________________________________" << endl;
29
30    Circle *circle1 = NULL;
31    Rectangle *rect1 = NULL;
32
33    cout << "Enter first form (rectangle or circle): ";
34    cin >> prompt;
35
36    if (inputIsCircle(prompt)) {
37       circle1 = circleCreator(inputIsCircle(prompt));
38       circle1->getBoxRef().print();
```

```
39
40          cout << "Move Circle in X direction for: ";
41          cin >> movX;
42          cout << "Move Circle in Y direction for: ";
43          cin >> movY;
44
45          circle1->move(movX, movY);
46          cout << "After Move is called: " << endl;
47          circle1->getBoxRef().print();
48       } else if (inputIsRect(prompt)) {
49          rect1 = rectCreator(inputIsRect(prompt));
50          rect1->getBoxRef().print();
51
52          cout << "Move Rectangle in X direction for: ";
53          cin >> movX;
54          cout << "Move Rectangle in Y direction for: ";
55          cin >> movY;
56
57          rect1->move(movX, movY);
58          cout << "After Move is called: " << endl;
59          rect1->getBoxRef().print();
60       }
61
62       Circle *circle2 = NULL;
63       Rectangle *rect2 = NULL;
64       cout << "Enter second form (rectangle or circle): ";
65       cin >> prompt;
66       if (inputIsCircle(prompt)) {
67          circle2 = circleCreator(inputIsCircle(prompt));
68          circle2->getBoxRef().print();
69
70          cout << "Move Circle in X direction for: ";
71          cin >> movX;
72          cout << "Move Circle in Y direction for: ";
73          cin >> movY;
74
75          circle2->move(movX, movY);
76          cout << "After Move is called: " << endl;
77          circle2->getBoxRef().print();
78       } else if (inputIsRect(prompt)) {
79          rect2 = rectCreator(inputIsRect(prompt));
80          rect2->getBoxRef().print();
81
82          cout << "Move Rectangle in X direction for: ";
83          cin >> movX;
84          cout << "Move Rectangle in Y direction for: ";
85          cin >> movY;
86
87          rect2->move(movX, movY);
88          cout << "After Move is called: " << endl;
```

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

```
138
139 bool inputIsRect(string prompt) {
140    string rect = "rectangle";
141    bool result = false;
142    if (prompt.compare(rect) == 0) {
143       result = true;
144    } else {
145       return result;
146    }
147
148    return result;
149 }
150
151 Circle* circleCreator(bool isTrue) {
152    if (isTrue) {
153       double rad;
154
155       cout << "Enter radius: ";
156       cin >> rad;
157       Circle *circle = new Circle(rad);
158       circle->setUpBox();
159       return circle;
160    } else {
161       return NULL;
162    }
163 }
164
165 Rectangle* rectCreator(bool isTrue) {
166    if (isTrue) {
167       double h, w;
168       cout << "Enter height: ";
169       cin >> h;
170       cout << "Enter width: ";
171       cin >> w;
172       Rectangle *rect = new Rectangle(h, w);
173       rect->setUpBox();
174       return rect;
175    } else {
176       return NULL;
177    }
178 }
```

Figure 12 Main Class

4 Foundations

5 Conclusion