# Programming 2 - Assignment 3

Felix Dreßler (k12105003)
email FelixDressler01@gmail.com

May 10, 2022

# 1 The Program

## 1.1 The Program - Polygon

### 1.1.1 Polygon.h

```
1  #pragma once
2
3  #include"LinkedList.h"
4
5
6  class Polygon
7  {
8  private:
9      unsigned int color = 0xFFFF;
10     LinkedListArr points;
11
12 public:
13     // create polygon in denoted color (default black)
14     Polygon(unsigned int color = 0);
15
16     // copy constructor, copy assignment operator, destructor
17     Polygon(Polygon& p);
18     Polygon& operator=(Polygon& p);
19     virtual ~Polygon();
20
21     // create a heap-allocated copy of this polygon
22     virtual Polygon* clone();
23
24     // add point with relative coordinates (x,y) to polygon
25     void add(double x, double y);
26
27     // draws the polygon at absolute coordinates (x0,y0) scaled by factor f;
28     // thus every point (x,y) is drawn at position (x0+x*f, y0+y*f)
29     void draw(double x0 = 0, double y0 = 0, double f = 1);
30
31     void setColor(unsigned int c){
32         this->color = c;
33     }
34
35     unsigned int getColor() {
36         return color;
37     }
38
39     int pNum() {
40         return points.length();
41     }
42
43     int getCoord(int i, int n) {
44         return points.get(i, n);
45     }
46 };
47
48 class RegularPolygon : public Polygon
49 {
50 private:
51     double a;
52     double x;
```

```
53      double y;
54      double r;
55      int n;
56  public:
57      RegularPolygon(double x, double y, double r, int n,
58          double a = 0, unsigned int c = 0);
59
60      virtual ~RegularPolygon();
61
62      virtual void draw(double x0 = 0, double y0 = 0, double f = 1);
63
64      virtual RegularPolygon* clone();
65  };
66
67  class Square : public RegularPolygon
68  {
69  public:
70      Square(double x, double y, double r,
71          double a = 0, unsigned int c = 0);
72  };
73
74  class Hexagon : public RegularPolygon
75  {
76  public:
77      Hexagon(double x, double y, double r,
78          double a = 0, unsigned int c = 0);
79  };
```

### 1.1.2 Polygon.cpp

```
1   #include"Polygon.h"
2   #include"LinkedList.h"
3   #include"Drawing.h"
4
5   #define _USE_MATH_DEFINES
6
7   #include<cmath>
8   #include<iostream>
9
10  using namespace std;
11  using namespace compsys;
12
13  //constructor
14  Polygon::Polygon(unsigned int color) {
15      this->color = color;
16      this->points;
17  }
18
19  Polygon::Polygon(Polygon& p) {
20      this->color = p.color;
21      int length = p.points.length();
22
23      for (int i = 0; i < length; i++) {
24          this->add(p.points.get(i, 0), p.points.get(i, 1));
25      }
26  }
27
28  Polygon& Polygon::operator=(Polygon& p) {
```

```
29
30      //check if it is already the same
31
32      this->color = p.color;
33      int length = p.points.length();
34
35      for(int i = 0; i < length; i++) {
36          this->add(p.points.get(i, 0), p.points.get(i, 1));
37      }
38
39
40      return *this;
41  }
42
43  Polygon::~Polygon() {
44
45  }
46
47  Polygon* Polygon::clone() {
48      Polygon* p = new Polygon(this->color);
49
50      int length = this->points.length();
51
52      for (int i = 0; i < length; i++) {
53          p->add(this->points.get(i,0), this->points.get(i, 1));
54      }
55
56      return p;
57  }
58
59  void Polygon::add(double x, double y) {
60      int e[2] = { x,y };
61      points.insert(e);
62  }
63
64  void Polygon::draw(double x0, double y0, double f) {
65      int length = points.length();
66      int* tempx = new int[length];
67      int* tempy = new int[length];
68
69      for (int i = 0; i < length; i++) {
70          tempx[i] = x0 + points.get(i,0)*f;
71          tempy[i] = y0 + points.get(i, 1)*f;
72      }
73
74      drawPolygon(length, tempx, tempy, color);
75
76      delete[] tempx;
77      delete[] tempy;
78  }
79
80  //
        ********************************************************************************************
81
82  RegularPolygon::RegularPolygon(double x, double y,
83      double r, int n, double a, unsigned int c):Polygon(c) {
84
85      for (double i = 0; i < n; i ++) {
```

```
86          this->add(x - (cos(a + i * 2 * M_PI / n) * r), y - (sin(a + i * 2 * M_PI / n) * r
                ));
87      }
88      this->x = x;
89      this->y = y;
90      this->n = n;
91      this->a = a;
92      this->r = r;
93      setColor(c);
94  }
95
96  void RegularPolygon::draw(double x0, double y0, double f) {
97      Polygon::draw(x0,y0,f);
98      drawPoint(x0 + x, y0 + y, getColor());
99  }
100
101 RegularPolygon::~RegularPolygon() {
102
103 }
104
105 RegularPolygon* RegularPolygon::clone() {
106     RegularPolygon* p = new RegularPolygon(this->x, this->y, this->r, this->pNum(), this
            ->a, this->getColor());
107
108     return p;
109 }
110
111 //
        ********************************************************************************************
112
113 Square::Square(double x, double y,
114     double r, double a, unsigned int c):RegularPolygon(x,y,r,4,a,c) {
115
116 }
117
118 //
        ********************************************************************************************
119
120 Hexagon::Hexagon(double x, double y,
121     double r, double a, unsigned int c) :RegularPolygon(x, y, r, 6, a, c) {
122
123 }
124
125 //
        ********************************************************************************************
```

## 1.2 The Program - Picture

### 1.2.1 Picture.h

```
1  #pragma once
2
3  #include"LinkedListPointer.h"
4  #include"Polygon.h"
```

```
5
6  class Picture
7  {
8  private:
9      LinkedListPointer Polygons;
10     unsigned int h;
11     unsigned int w;
12 public:
13     Picture();
14     Picture(Picture& p);
15     Picture& operator=(Picture& p);
16     ~Picture();
17     void add(Polygon& p);
18     void draw(double x, double y, double w, double h, double f = 1.0);
19 };
```

### 1.2.2 Picture.cpp

```
1  #include"Polygon.h"
2  #include"LinkedListPointer.h"
3  #include"Drawing.h"
4  #include"Picture.h"
5
6  #define _USE_MATH_DEFINES
7
8  #include<iostream>
9  #include<cmath>
10
11 using namespace std;
12
13 Picture::Picture() {
14     h = 200;
15     w = 200;
16 }
17
18 Picture::Picture(Picture& p) {
19     this->h = p.h;
20     this->w = p.w;
21
22     int length = Polygons.length();
23     for (int i = 0; i < length; i++) {
24         this->Polygons.insert(p.Polygons.get(i));
25     }
26 }
27
28 //copies pointers not values, needs to be changed
29 Picture& Picture::operator=(Picture& p) {
30     this->h = p.h;
31     this->w = p.w;
32
33     int length = Polygons.length();
34     for (int i = 0; i < length; i++) {
35         this->Polygons.insert(p.Polygons.get(i));
36     }
37
38     return *this;
39 }
40
```

```
41 Picture::~Picture() {
42     //delete clones in heap
43 }
44
45 void Picture::add(Polygon &p) {
46     this->Polygons.insert(p.clone());
47 }
48
49 void Picture::draw(double x, double y, double w, double h, double f) {
50     Polygon frame(0);
51     frame.add(0, 0);
52     frame.add(0, h);
53     frame.add(w, h);
54     frame.add(w, 0);
55     frame.draw(x,y,f);
56
57     int length = Polygons.length();
58
59     for (int i = 0; i < length; i++) {
60         Polygons.get(i)->draw(x,y,f);
61     }
62 }
```

## 1.3 The Program - Linked Lists

### 1.3.1 LinkedListArr.h

```
1  //****************************************************************
2  // "LinkedList.h"
3  //
4  // contains the class "LinkedList", that implements a linked list
5  // based on the linked list that was presented in the lecture slides.
6  //
7  // created by Felix Dressler,
8  //****************************************************************
9
10 #pragma once
11
12 class LinkedListArr
13 {
14   class Node;
15 private:
16   Node* head;
17   int number; //starts with 0 for the first element
18 public:
19   LinkedListArr();
20   ~LinkedListArr();
21
22   int length() const;
23   LinkedListArr& insert(int* e);
24   int get(int i, int n) const;
25 };
```

### 1.3.2 LinkedListArr.cpp

```
1  //*************************************************************
2  // "LinkedList.h"
3  //
4  // contains the methods of the class "LinkedList" based on the
5  // functionality as presented in the lecture slides.
6  //
7  // created by Felix Dressler,
8  //*************************************************************
9
10 #include "LinkedList.h"
11
12
13 class LinkedListArr::Node {
14   friend class LinkedListArr;
15 private:
16   int* value; Node* next;
17   Node(int* v, Node* n) {
18     next = n;
19     value = new int[2];
20     value[0] = v[0];
21     value[1] = v[1];
22   }
23 };
24
25 LinkedListArr::LinkedListArr() {
26   head = 0;
27   number = 0;
28 }
29
30 LinkedListArr::~LinkedListArr() {
31   Node* node = head;
32   while (node != 0) {
33     Node* node0 = node->next;
34     delete node;
35     node = node0;
36   }
37 }
38
39 int LinkedListArr::length() const {
40   return number;
41 }
42 LinkedListArr& LinkedListArr::insert(int* e) {
43   Node* node = new Node(e, head);
44   head = node;
45   number = number + 1;
46   return *this;
47 }
48 int LinkedListArr::get(int i, int n) const {
49   Node* node = head;
50   for (int j = 0; j < number - i - 1; j++)
51     node = node->next;
52   return node->value[n];
53 }
54
55 //
    //*************************************************************************************************
```

### 1.3.3 LinkedListPointer.h

```
#pragma once

#include"Polygon.h"

class LinkedListPointer
{
    class PointerNode;
private:
    PointerNode* head;
    int number; //starts with 0 for the first element
public:
    LinkedListPointer();
    ~LinkedListPointer();

    int length() const;
    LinkedListPointer& insert(Polygon* e);
    Polygon* get(int i) const;
};
```

### 1.3.4 LinkedListPointer.cpp

```
#include"LinkedListPointer.h"
#include"Polygon.h"

#include<iostream>

using namespace std;

class LinkedListPointer::PointerNode {
    friend class LinkedListPointer;
private:
    Polygon* value; PointerNode* next;
    PointerNode(Polygon* v, PointerNode* n) {
        next = n;
        value = v;
    }
};

LinkedListPointer::LinkedListPointer() {
    head = 0;
    number = 0;
}

LinkedListPointer::~LinkedListPointer() {
    PointerNode* node = head;
    while (node != 0) {
        PointerNode* node0 = node->next;
        delete node;
        node = node0;
    }
}

int LinkedListPointer::length() const {
    return number;
}
```

```
35 LinkedListPointer& LinkedListPointer::insert(Polygon* e) {
36     PointerNode* node = new PointerNode(e, head);
37     head = node;
38     number = number + 1;
39     return *this;
40 }
41 Polygon* LinkedListPointer::get(int i) const {
42     PointerNode* node = head;
43     for (int j = 0; j < number - i - 1; j++)
44         node = node->next;
45     return node->value;
46 }
```

# 2 Problems

This section will briefly discuss the Problems that have occurred during programming.

## 2.1 warnings

In the *resize* method, line 283 of the *DistPoly.cpp* this warning is displayed: