Programovacie techniky

Virtuálne metódy a deštruktory

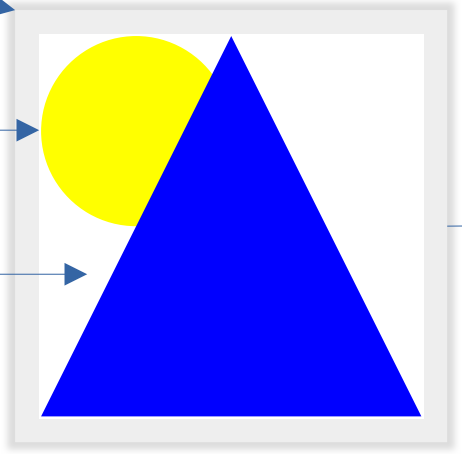
Vladislav Novák

Editor obrázkov

class Image { ..... }

class Circle { ..... }

class Polygon { ..... }



```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
"http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">

<svg version="1.1" width="100" height="100"
     viewBox="0 0 100 100"
     xmlns="http://www.w3.org/2000/svg"
     xmlns:xlink="http://www.w3.org/1999/xlink">
  <circle cx="25" cy="25" r="25" fill="yellow" />
  <polygon points="50, 0, 100, 100, 0, 100, " fill="blue" />
</svg>
```

.svg

Editor obrázkov

```
class Circle {



};
```

```
class Polygon {



};
```

Editor obrázkov

```
struct Point {
    int x;
    int y;
};
```

```
class Circle {



};
```

```
class Polygon {



};
```

Editor obrázkov

```
struct Point {
    int x;
    int y;
};
```

```
class Circle {
private:
    Point centre;
    int radius;
    string color;




};
```

```
class Polygon {




};
```

Editor obrázkov

```
struct Point {
    int x;
    int y;
};
```

```
class Circle {
private:
    Point centre;
    int radius;
    string color;




};
```

```
class Polygon {
private:
    list<Point> points;
    string color;




};
```

Editor obrázkov

```
struct Point {
    int x;
    int y;
};
```

```
class Circle {
private:
    Point centre;
    int radius;
    string color;
    // .....
public:
    // .....
    string toSvg() const {
        // .....
    }
    // .....
};
```

```
class Polygon {
private:
    list<Point> points;
    string color;
    // .....

public:
    // .....
    string toSvg() const {
        // .....
    }
    // .....
};
```

Editor obrázkov

```
class Image {
private:
    list<Circle*> circles;
    list<Polygon*> polygons;
    // .....
};
```

Editor obrázkov

```
class Image {
private:
    list<Circle*> circles;
    list<Polygon*> polygons;
    // .....
};
```

Editor obrázkov

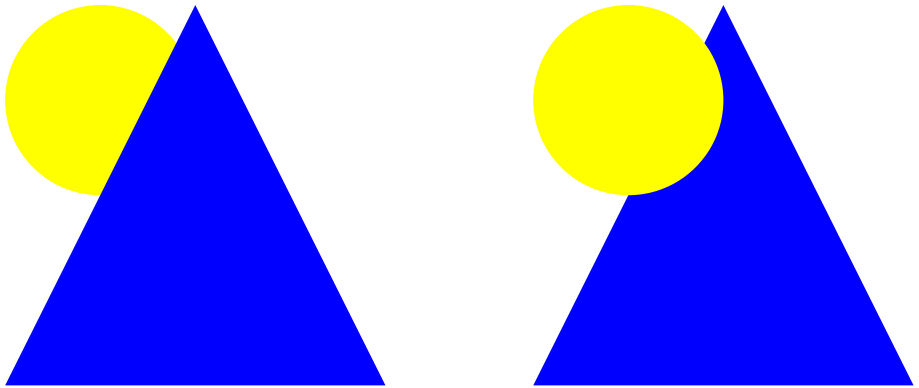
```
class Image {
private:
   list<Circle*> circles;
   list<Polygon*> polygons;
public:
   void addCircle(Circle *newCircle) {
      // .....
   }
   void addPolygon(Polygon *newPolygon) {
      // .....
   }
   // .....
};
```

Editor obrázkov

```
class Image {
private:
   list<Circle*> circles;
   list<Polygon*> polygons;
public:
   // .....
   ~Image() {
      for(Circle *c: circles) {
         delete c;
      }
      for(Polygon *p: polygons) {
         delete p;
      }
   }
};
```

Editor obrázkov

```
class Image {
private:
    list<Circle*> circles;
    list<Polygon*> polygons;
public:
    // .....
    void toSvg() const {
        for(Circle *c: circles) {
            c->toSvg();
        }
        for(Polygon *p: polygons) {
            p->toSvg();
        }
    }
    // .....
};
```



Editor obrázkov

```
class Image {
private:
   list<Circle*> circles;
   list<Polygon*> polygons;
public:
   // .....
   void toSvg() const {
      for(Circle *c: circles) {
         c->toSvg();
      }
      for(Polygon *p: polygons) {
         p->toSvg();
      }
   }
   // .....
};
```

```
class Element {
public:
   virtual string toSvg() const = 0;
};
```

Editor obrázkov

```
class Image {
private:
   list<Circle*> circles;
   list<Polygon*> polygons;
public:
   // .....
   void toSvg() const {
      for(Circle *c: circles) {
         c->toSvg();
      }
      for(Polygon *p: polygons) {
         p->toSvg();
      }
   }
   // .....
};
```

```
class Element {
public:
   virtual string toSvg() const = 0;
};
```

```
class Circle : public Element {
   // .....
public:
   string toSvg() const override {
      // .....
   }
   // .....
};
```

```
class Polygon : public Element {
   // .....
public:
   string toSvg() const override {
      // .....
   }
   // .....
};
```

Editor obrázkov

```
class Image {
private:
    list<Element*> elements;




};
```

```
class Element {
public:
    virtual string toSvg() const = 0;
};
```

```
class Circle : public Element {
    // .....
public:
    string toSvg() const override {
        // .....
    }
    // .....
};
```

```
class Polygon : public Element {
    // .....
public:
    string toSvg() const override {
        // .....
    }
    // .....
};
```

Editor obrázkov

```
class Image {
private:
   list<Element*> elements;
public:
   // .....
   void toSvg() const {
      // .....
      for(Element *e: elements) {
         e->toSvg();
      }
      // .....
   }
   // .....
};
```

```
class Element {
public:
   virtual string toSvg() const = 0;
};
```

```
class Circle : public Element {
   // .....
public:
   string toSvg() const override {
      // .....
   }
   // .....
};
```

```
class Polygon : public Element {
   // .....
public:
   string toSvg() const override {
      // .....
   }
   // .....
};
```

Editor obrázkov

```
class Image {
private:
    list<Element*> elements;
public:


    void addElement(Element *newElement) {
        elements.push_back(element);
    }
    // .....
    ~Image() {
        for(Element *e: elements) {
            delete e;
        }
    }
};
```

```
class Element {
public:
    virtual string toSvg() const = 0;
};
```

```
class Circle : public Element {
    // .....
public:
    string toSvg() const override {
        // .....
    }
    // .....
};
```

```
class Polygon : public Element {
    // .....
public:
    string toSvg() const override {
        // .....
    }
    // .....
};
```

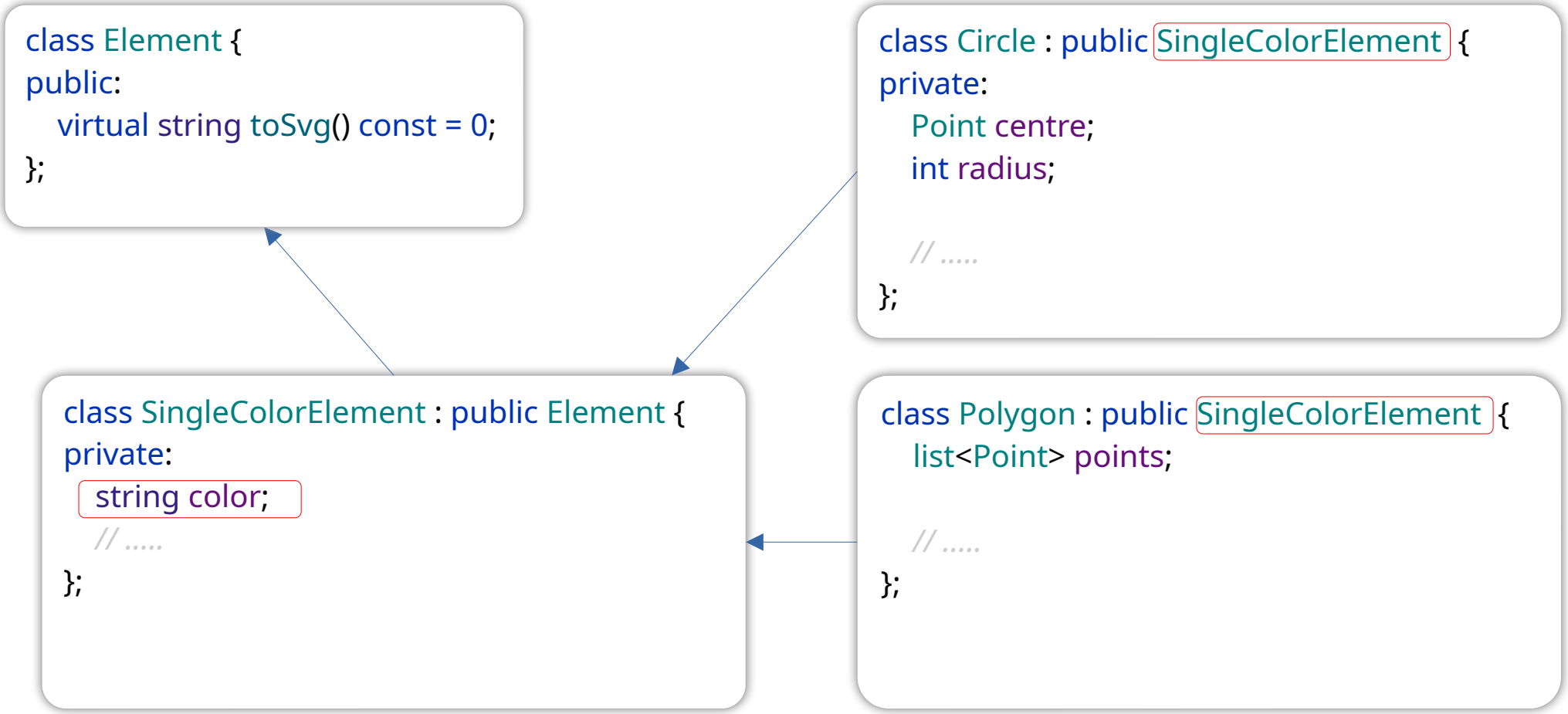
Editor obrázkov

```
class Element {
public:
    virtual string toSvg() const = 0;
};
```

```
class Circle : public Element  {
private:
    Point centre;
    int radius;
    string color;
    // .....
};
```

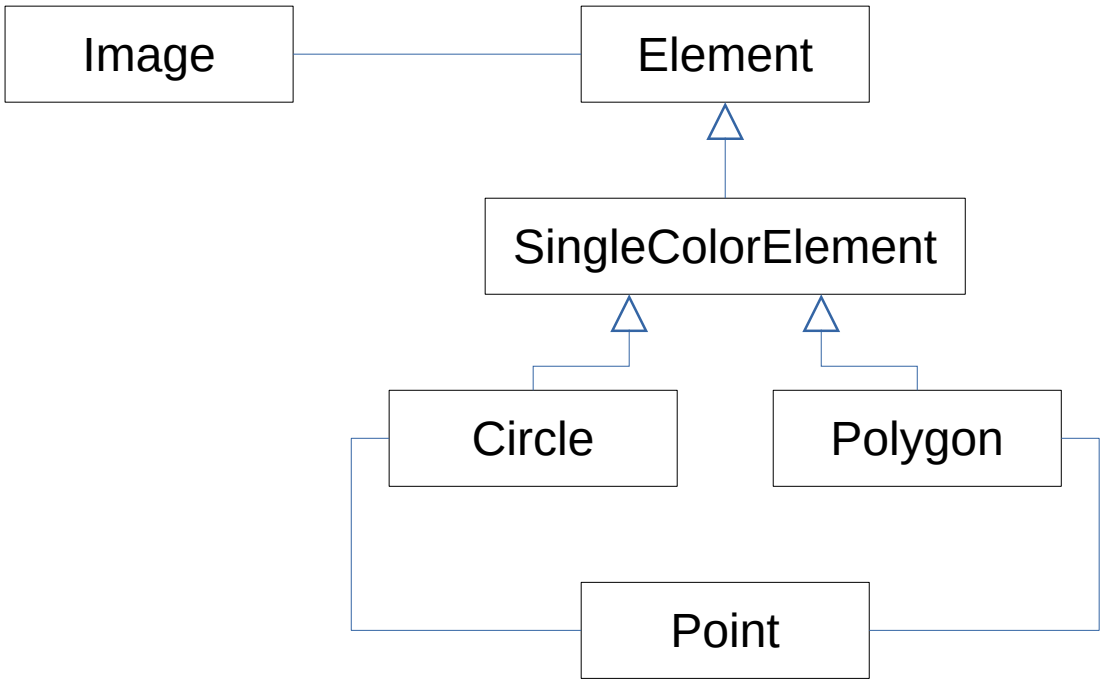
```
class Polygon : public Element  {
    list<Point> points;
    string color;
    // .....
};
```

Editor obrázkov



Editor obrázkov

Diagram tried

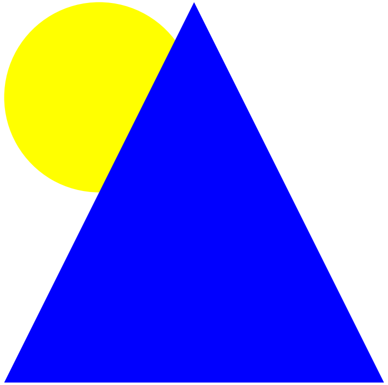


Formát SVG

Zjednodušený formát SVG:

<svg >

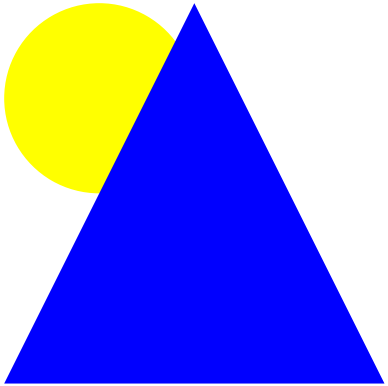
</svg>



Formát SVG

Zjednodušený formát SVG:

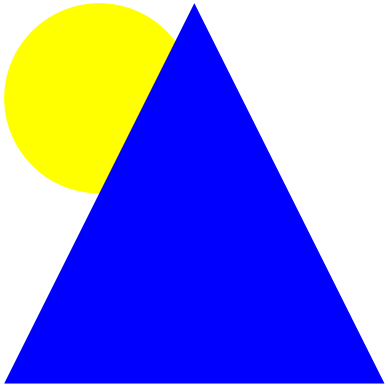
```
<svg version="1.1" width="100" height="100" viewBox="0 0 100 100"  xmlns="http://www.w3.org/2000/svg" >

</svg>
```



Formát SVG

Zjednodušený formát SVG:

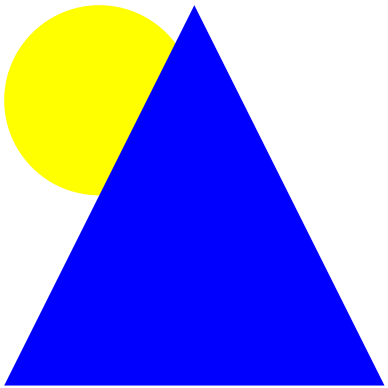
```
<svg version="1.1" width="100" height="100" viewBox="0 0 100 100"  xmlns="http://www.w3.org/2000/svg" >
   <circle                                                  />
   <polygon                                                        />
</svg>
```



Formát SVG

Zjednodušený formát SVG:

```
<svg version="1.1" width="100" height="100" viewBox="0 0 100 100"  xmlns="http://www.w3.org/2000/svg" >
   <circle cx="25" cy="25" r="25" fill="yellow" />
   <polygon points="50, 0, 100, 100, 0, 100" fill="blue" />
</svg>
```



Editor obrázkov

```
class Element {
protected:
   static string quoted(string value) {
      return "\"" + value + "\"";
   }
   static string quoted(int value)  {
      return quoted(to_string(value));
   }
public:
   virtual string toSvg() const = 0;
};
```

```
class SingleColorElement : public Element {
private:
   string color; // farba elementu
protected:
   string colorAsSvg() const {
      return "fill=" + quoted(color) + "";
   }
   // .....
};
```

```
class Circle: public SingleColorElement {
private:
   Point center; // pozicia stredu
   int radius; // polomer
public:
   // .....
   string toSvg() const override {
      return "<circle cx="+quoted(center.x)+" cy="+quoted(center.y)+" r="+quoted(radius)+" "+colorAsSvg()+"/>";
   }
```

Editor obrázkov

```
class Image {
   // .....
   string toSvg() const {
      ostringstream output;
      output << "<svg version=\"1.1\" width=\"100\" height=\"100\" xmlns=\"http://www.w3.org/2000/svg\" >" << endl;
      for(const Element *e: elements) {
         output << "    " << e->toSvg() << endl;
      }
      output << "</svg>" << endl;
      return output.str();
   }
   // .....
};
```

Editor obrázkov

Zdrojový súbor bude zverejný

Editor obrázkov

```
class Element {
public:
   // .....
   virtual ~Element() { ..... }
};
```

```
class Image {
private:
   list<Element*> elements;
public:
   // .....
   ~Image() {
      for(Element *e: elements) {
         delete e;
      }
   }
};
```

ak deštruktor potrebný v podtriedach

```
class Circle : public Element {
   // .....
public:
   // .....
   ~Circle() override { ..... }
};
```

```
class Polygon : public Element {
   // .....
public:
   // .....
   ~Polygon() override { ..... }
};
```

Konštrukcia a deštrukcia objektu

Definícia

```
class A {
   int x;
};
```

Konštrukcia a deštrukcia objektu

Definícia

```
class A {
   int x;
};

class B : public A {
   int y;
};
```

Konštrukcia a deštrukcia objektu

Definícia

```
class A {
   int x;
};

class B : public A {
   int y;
};
```

Konštrukcia objektu typu B

Konštrukcia a deštrukcia objektu

Definícia

```
class A {
   int x;
};

class B : public A {
   int y;
};
```

Konštrukcia objektu typu B

konštrukcia A

Konštrukcia a deštrukcia objektu

Definícia

```
class A {
   int x;
};

class B : public A {
   int y;
};
```

Konštrukcia objektu typu B

konštruktor x } konštrukcia A

Konštrukcia a deštrukcia objektu

Definícia

```
class A {
   int x;
};

class B : public A {
   int y;
};
```

Konštrukcia objektu typu B

konštruktor x
konštruktor A
} konštrukcia A

Konštrukcia a deštrukcia objektu

Definícia

```
class A {
   int x;
};

class B : public A {
   int y;
};
```

Konštrukcia objektu typu B

konštruktor x

konštruktor A

Konštrukcia a deštrukcia objektu

Definícia

```
class A {
   int x;
};

class B : public A {
   int y;
};
```

Konštrukcia objektu typu B

konštruktor x

konštruktor A

konštruktor y

Konštrukcia a deštrukcia objektu

Definícia

```
class A {
   int x;
};

class B : public A {
   int y;
};
```

Konštrukcia objektu typu B

konštruktor x

konštruktor A

konštruktor y

konštruktor B

Konštrukcia a deštrukcia objektu

Definícia

```
class A {
   int x;
};

class B : public A {
   int y;
};
```

Konštrukcia objektu typu B

konštruktor x

konštruktor A

konštruktor y

konštruktor B

Deštrukcia objektu typu B

Konštrukcia a deštrukcia objektu

Definícia

```
class A {
   int x;
};

class B : public A {
   int y;
};
```

Konštrukcia objektu typu B

konštruktor x

konštruktor A

konštruktor y

konštruktor B

Deštrukcia objektu typu B

deštrukcia B-čkovej časti

Konštrukcia a deštrukcia objektu

Definícia

```
class A {
   int x;
};

class B : public A {
   int y;
};
```

Konštrukcia objektu typu B

konštruktor x

konštruktor A

konštruktor y

konštruktor B

Deštrukcia objektu typu B

deštruktor B } deštrukcia B-čkovej časti

Konštrukcia a deštrukcia objektu

Definícia

```
class A {
   int x;
};

class B : public A {
   int y;
};
```

Konštrukcia objektu typu B

konštruktor x

konštruktor A

konštruktor y

konštruktor B

Deštrukcia objektu typu B

deštruktor B

deštruktor y

} deštrukcia B-čkovej časti

Konštrukcia a deštrukcia objektu

Definícia

```
class A {
   int x;
};

class B : public A {
   int y;
};
```

Konštrukcia objektu typu B

konštruktor x

konštruktor A

konštruktor y

konštruktor B

Deštrukcia objektu typu B

deštruktor B

deštruktor y

} deštrukcia A-čkovej časti

Konštrukcia a deštrukcia objektu

Definícia

```
class A {
   int x;
};

class B : public A {
   int y;
};
```

Konštrukcia objektu typu B

konštruktor x

konštruktor A

konštruktor y

konštruktor B

Deštrukcia objektu typu B

deštruktor B

deštruktor y

deštruktor A

} deštrukcia A-čkovej časti

Konštrukcia a deštrukcia objektu

Definícia

```
class A {
   int x;
};

class B : public A {
   int y;
};
```

Konštrukcia objektu typu B

konštruktor x

konštruktor A

konštruktor y

konštruktor B

Deštrukcia objektu typu B

deštruktor B

deštruktor y

deštruktor A

deštruktor x

} deštrukcia A-čkovej časti