KLASY ABSTRAKCYJNE

CO WSPÓLNEGO MAJĄ ZE SOBĄ:

Koło

Prostokąt

Trójkąt

Ciężarówka

Samochód osobowy

Autobus

CO WSPÓLNEGO MAJĄ ZE SOBĄ:

Koło

Prostokąt

Trójkąt

FIGURA 2D

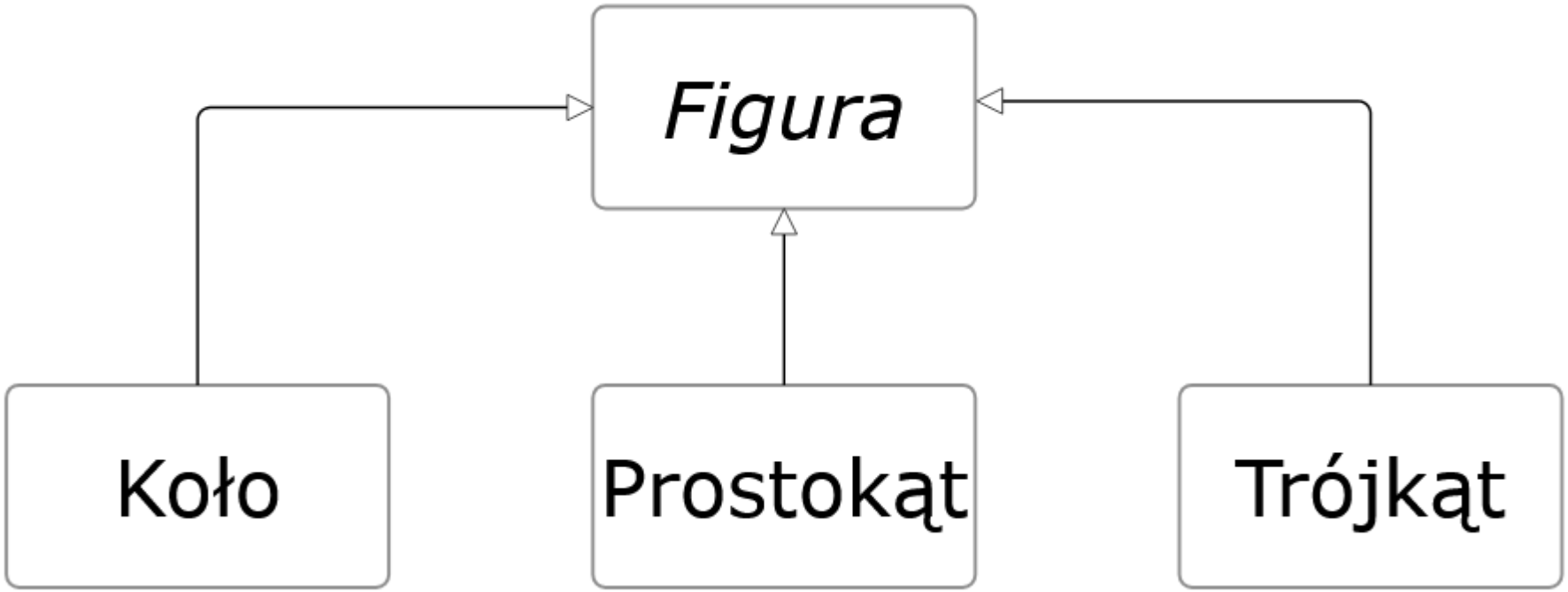
Ciężarówka

Samochód osobowy

Autobus

POJAZD SILNIKOWY

DIAGRAM KLAS UML



Klasa Figura ma sens tylko jako klasa podstawowa jakiejś klasy pochodnej, ponieważ nie można dostarczyć sensownej definicji jej funkcji wirtualnych.

```
class Figura {
public:
    virtual void obróć(int) { błąd("Figura::obróć"); }
    virtual void rysuj() { błąd("Figura::rysuj"); }
    // ...
};
```

Lepszym rozwiązaniem jest zadeklarowanie funkcji wirtualnych klasy Figura jako czystych funkcji wirtualnych.

Funkcja wirtualna staje się czysta, jeśli inicjator ma postać =0

```
class Figura {                          // klasa abstrakcyjna
public:
    virtual void obróć(int) = 0;        // czysta funkcja wirtualna
    virtual void rysuj() = 0;           // czysta funkcja wirtualna
    // ...
};
```

Klasa z jedną lub wieloma czystymi funkcjami wirtualnymi jest klasą abstrakcyjną.

Nie można tworzyć żadnych obiektów takiej klasy, czyli

```
Figura f;       // błąd: zmienna abstrakcyjnej klasy Figura
```

Klasy abstrakcyjnej można użyć jedynie jako interfejsu i jako klasy podstawowej dla innej klasy, np.

```
class Punkt { /* ... */ };

class Okrąg : public Figura {
public:
    void obróć(int) { }          // zastąp Figura::obróć
    void rysuj();                // zastąp Figura::rysuj
    Okrąg(Punkt p, int pr);
private:
    Punkt środek;
    int promień;
};
```

KOMPLETNY PROGRAM – CZĘŚĆ I

```
#include <iostream>
#include <vector>
#include <cmath>
using namespace std;

class Figura {
public:
    virtual double pole() const = 0;
    virtual double obwod() const = 0;
};
```

KOMPLETNY PROGRAM – CZĘŚĆ II

```
class Kolo : public Figura {
    double promien;
public:
    Kolo(double p) : promien(p) {}
    double pole() const;
    double obwod() const;
};

double Kolo::pole() const {
    return M_PI * promien * promien;
}

double Kolo::obwod() const {
    return 2 * M_PI * promien;
}
```

KOMPLETNY PROGRAM – CZĘŚĆ III

```
class Prostokat : public Figura {
    double bok_a, bok_b;
public:
    Prostokat(double a, double b) : bok_a(a), bok_b(b) {}
    double pole() const;
    double obwod() const;
};

double Prostokat::pole() const {
    return bok_a * bok_b;
}

double Prostokat::obwod() const {
    return 2 * bok_a + 2 * bok_b;
}
```

KOMPLETNY PROGRAM – CZĘŚĆ IV

```
int main()
{
    vector<Figura*> v;
    v.push_back(new Kolo(1));
    v.push_back(new Prostokat(2, 3));

    for(int i=0; i<(int)v.size(); ++i)
    {
        cout << "Figura " << i << ": " << endl;
        cout << "   pole = " << v[i]->pole() << endl;
        cout << "   obwod = " << v[i]->obwod() << endl;
    }
}
```