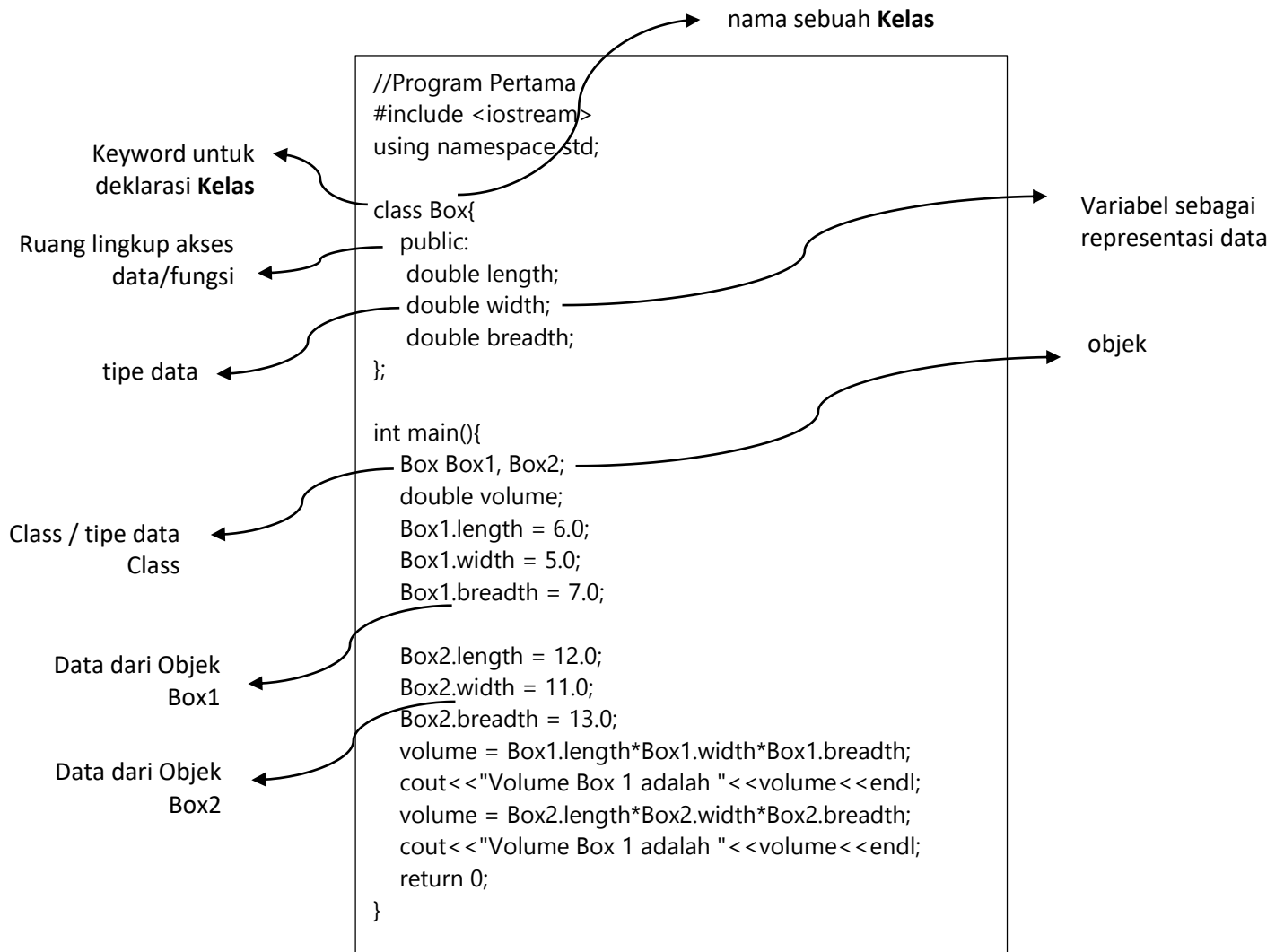


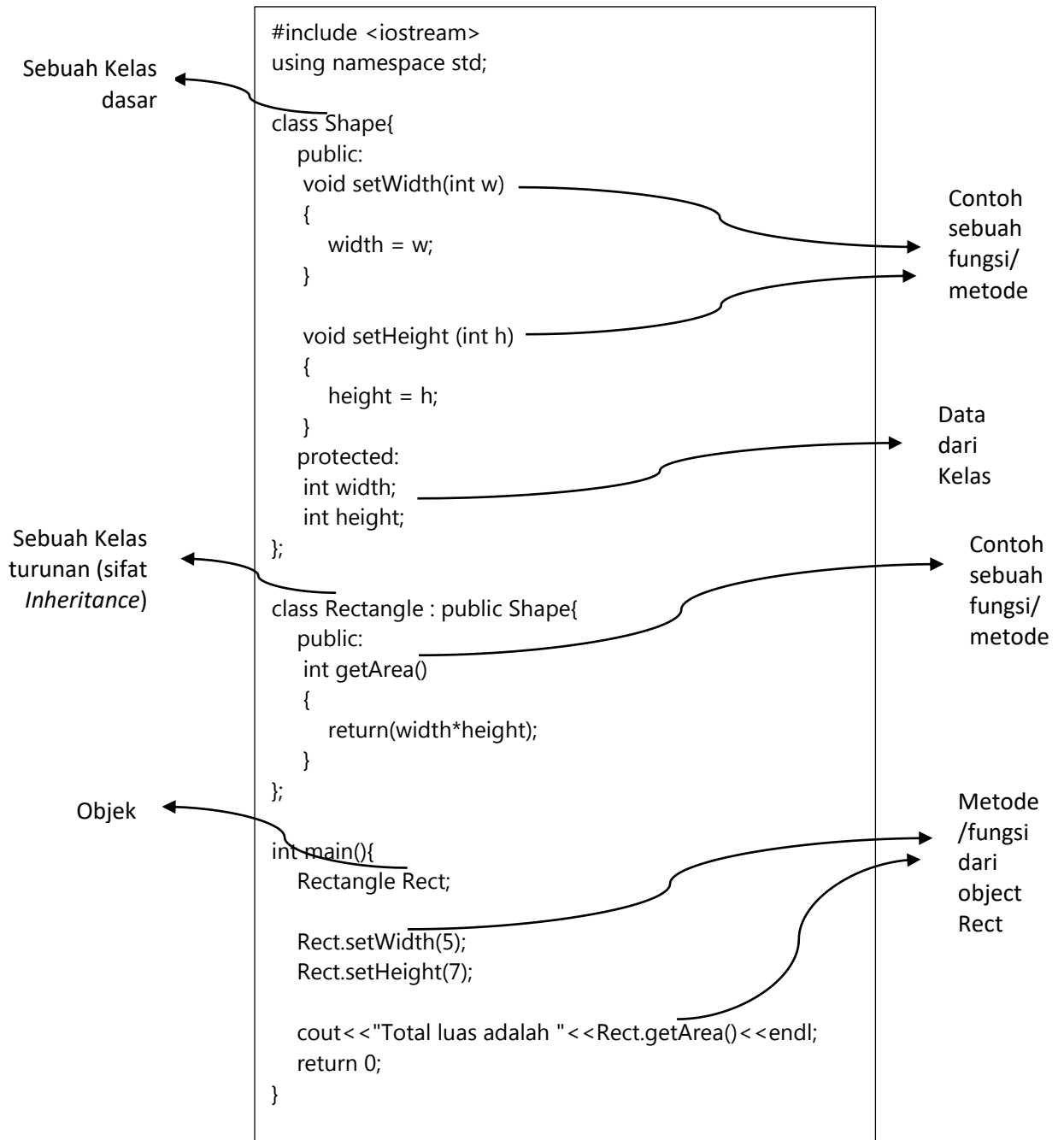
KELAS (CLASS) dan OBJEK

Kelas digunakan untuk menspesifikasikan bentuk suatu objek yang mengkombinasikan representasi data dengan metode (fungsi) untuk memanipulasi data tersebut dalam satu paket. Data dan fungsi dalam suatu kelas disebut anggota kelas. Kelas didefinisikan dengan keyword **class** dan diikuti nama kelas.

Perhatikan program berikut :



Program kedua



INHERITANCE (Pewarisan)

Pewarisan adalah salah satu konsep dalam paradigma Pemrograman Berorientasi Objek yang intinya suatu kelas dasar (dalam contoh program kedua adalah Shape) dapat diturunkan menjadi suatu kelas yang lain atau kelas turunan dengan sifat yang sama dengan kelas dasarnya (dalam contoh program kedua adalah Rectangle).

Jika suatu objek dideklarasikan dengan tipe data kelas turunan (contoh : Rectangle Rect) maka objek tersebut akan memiliki data dan fungsi yang dimiliki oleh baik kelas dasar maupun kelas turunan dengan tetap memperhatikan penyematan ruang lingkup akses yang menyertai data serta fungsi yang ada.

Adapun ragam ruang akses adalah public, private dan protected. Private adalah default ruang lingkup data dan fungsi dalam sebuah kelas. Maka jika data dan fungsi dalam kelas dalam deklarasinya tidak diawali dengan public ataupun protected maka data dan fungsi tersebut bersifat private.

Ruang lingkup private berarti suatu data dan fungsi hanya dapat diakses dalam kelas dimana data dan fungsi tersebut dideklarasikan.

Ruang lingkup protected berarti suatu data dan fungsi dapat diakses oleh kelas dasar dan kelas turunannya, tetapi tidak untuk kelas diluar kelas tersebut.

Ruang lingkup public berarti suatu data dan fungsi dapat diakses secara umum oleh kelas apapun.

Perhatikan contoh program ketiga berikut ini:

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

class Shape{
public:
    void setWidth(int w)
    {
        width = w;
    }

    void setHeight (int h)
    {
        height = h;
    }
protected:
    int width;
    int height;
};

class PaintCost{
public:
    int getCost(int area)
    {
        return area*70;
    }
};

class Rectangle : public Shape, public PaintCost{
public:
    int getArea()
    {
        return(width*height);
    }
};

int main(){
    Rectangle Rect;
    int area;

    Rect.setWidth(5);
    Rect.setHeight(7);

    area=Rect.getArea();
    cout<<"Total area = "<<area<<endl;
    cout<<"Total cost = "<<Rect.getCost(area)<<endl;

    return 0;
}
```

OVERLOADING

Overloading adalah kemampuan program mengeksekusi suatu fungsi yang sama tetapi dengan memiliki parameter dengan tipe data berbeda.

Pada contoh keempat berikut ini tampak bahwa terdapat sebuah kelas printData.

Perhatikan contoh program keempat berikut ini:

```
//OVERLOADING
#include <iostream>
using namespace std;

class printData{
public:
    void print(int i)
    {
        cout<<"Printing int:"<<i<<endl;
    }

    void print(double f)
    {
        cout<<"Printing float :"<<f<<endl;
    }
};

int main(){
    printData pd;
    pd.print(5);
    pd.print(500.23);
    return 0;
}
```