Նախագծման Ձևանմուշներ: Composite

Հրաչյա Թանդիլյան

2020

Composite

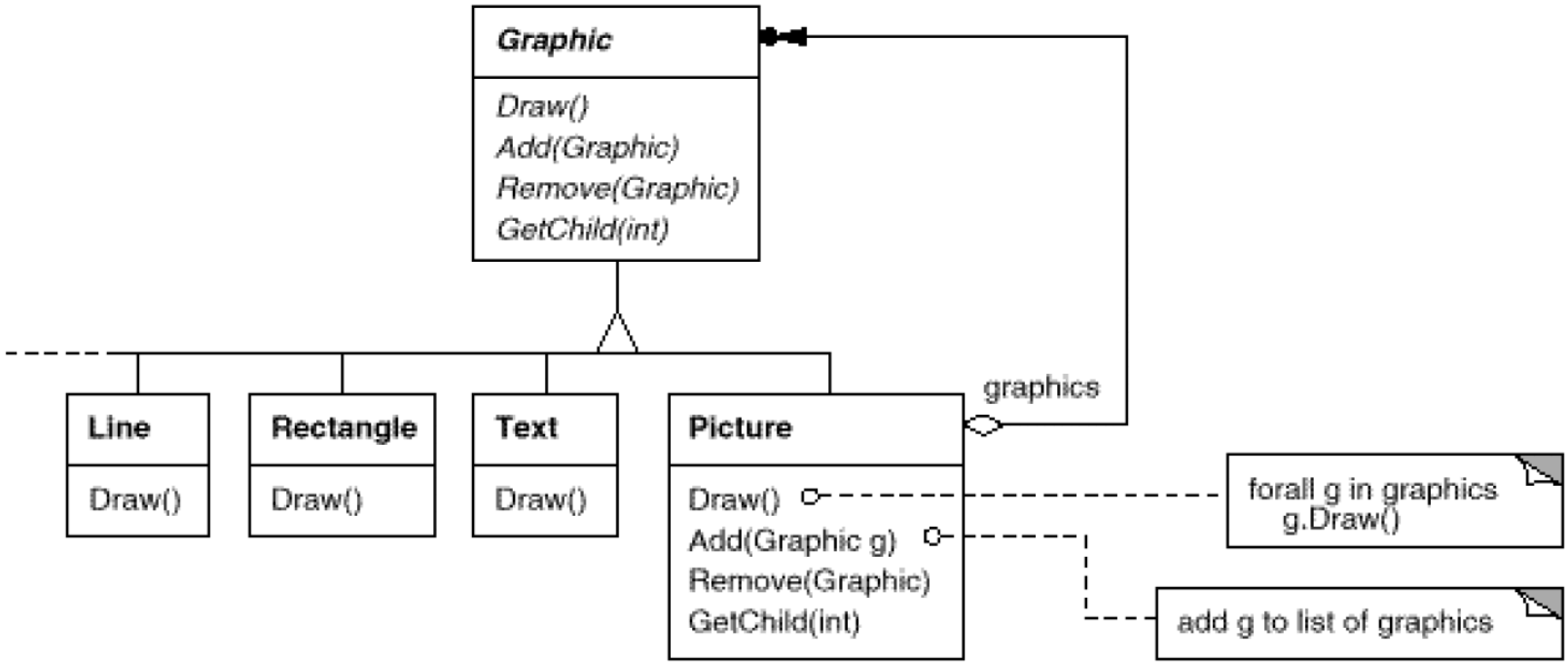
Նպատակը

Խմբավորել օբյեկտները ծառատիպ կառուցվածքներում ամբողջական հիերարխիաներ ներկայացնելու համար:
Այս Ն.Ձ. թույլ է տալիս միանման դիտարկել և՛ անհատական օբյեկտները, և՛ նրանց խմբերը:

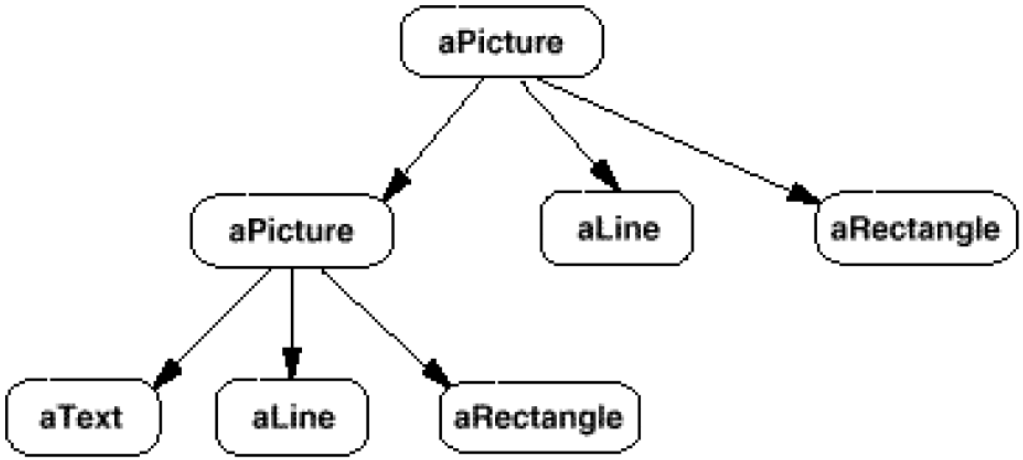
Նաև հայտնի է որպես

- Այլ լայնորեն կիրառվող անուներ չկան:

Մոտիվացիան


Graphic
Draw()
Add(Graphic)
Remove(Graphic)
GetChild(int)
Line
Draw()
Rectangle
Draw()
Text
Draw()
Picture
Draw()
Add(Graphic g)
Remove(Graphic)
GetChild(int)
graphics
forall g in graphics
g.Draw()
add g to list of graphics

Մոտիվացիան


aPicture
aPicture
aLine
aRectangle
aText
aLine
aRectangle

Կիրառելիությունը

Այս Ն.Ձ. պետք է օգտագործել երբ.

- Անհրաժեշտ է ներկայացնել օբյեկտների ամբողջական հիերարխիաներ:

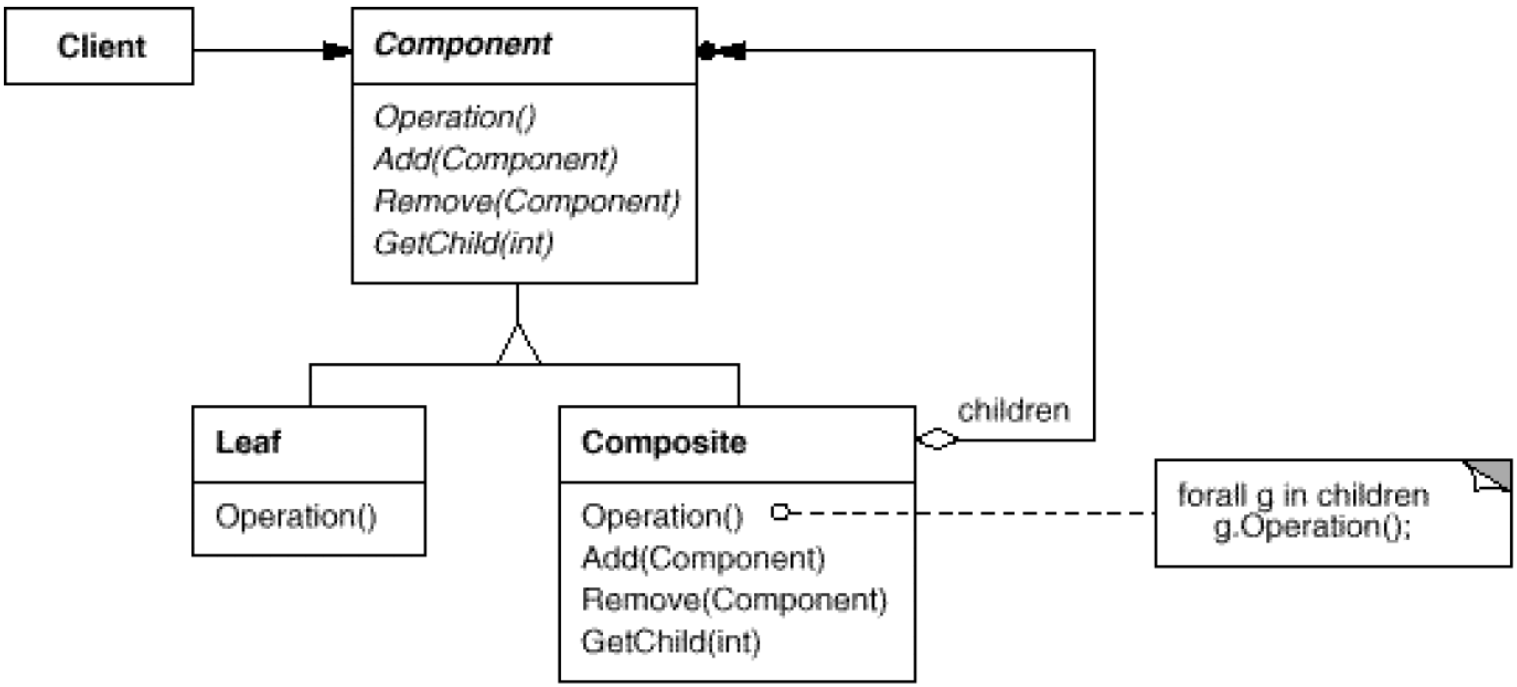
Կիրառելիությունը

Այս Ն.Ձ. պետք է օգտագործել երբ.

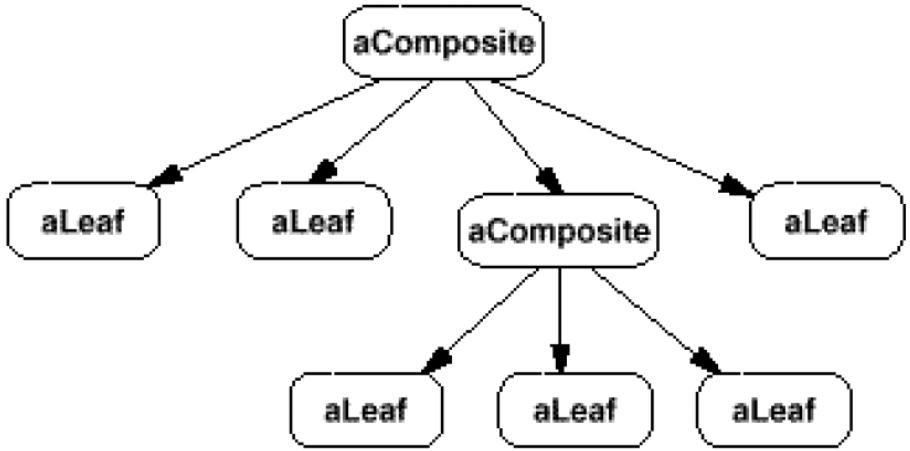
Ա Անհրաժեշտ է ներկայացնել օբյեկտների ամբողջական հիերարխիաներ:

Բ Անհրաժեշտ է, որ օգտագործողը հնարավորություն ունենա թափանցիկ կերպով օգտագործել կոմպոզիտ և անհատական օբյեկտները, անտեսել նրանց տարբերությունները:

Կառուցվածքը



Կառուցվածքը



Հետևանքները

Այս Ն.Ձ. ունի հետևյալ առավելություններն ու թերությունները.

- Սահմանում է օբյեկտների հիերարխիա։

Հետևանքները

Այս Ն.Ձ. ունի հետևյալ առավելություններն ու թերությունները.

Ա Սահմանում է օբյեկտների հիերարխիա:

Բ Պարզեցնում է օգտագործողի կոդը:

Հետևանքները

Այս Ն.Ձ. ունի հետևյալ առավելություններն ու թերությունները.

Ա Սահմանում է օբյեկտների հիերարխիա:

Բ Պարզեցնում է օգտագործողի կոդը:

Գ Հեշտացնում է նոր տիպի կոմպոնենտների ավելացումը:

Հետևանքները

Այս Ն.Ձ. ունի հետևյալ առավելություններն ու թերությունները.

Ա Սահմանում է օբյեկտների հիերարխիա:

Բ Պարզեցնում է օգտագործողի կոդը:

Գ Հեշտացնում է նոր տիպի կոմպոնենտների ավելացումը:

Դ Նախագծի չափազանց ընդհանրություն:

Իրականացումը

Ա. Բացահայտ հղում ծնողին:

Բ. Կոմպոնենտների համատեղ օգտագործում:

Գ. Կոմպոնենտի ինտերֆեյսի մաքսիմիզացում:

Դ. Զավակ կոմպոնենտների ղեկավարման գործողությունների նկարագրում:

Ե. Անհրաժեշտ է արդյոք զավակ կոմպոնենտների ցուցակը սահմանել Component դասում:

Իրականացումը

- Ձավակների կարգավորումը:
- Քեշավորում արագության բարելավման համար:
- Ով պետք է ջնջի կոմպոնենտները:
- Որն է կոմպոնենտները պահելու համար լավագույն տվյալների կառուցվածքը:

Օրինակ

```
class Equipment {

public:
    virtual ~Equipment();
    const char* Name() { return name; }
    virtual Watt Power();
    virtual Currency NetPrice();

    virtual void Add(Equipment*);
    virtual void Remove(Equipment*);
    virtual Iterator* CreateIterator();

protected:
    Equipment(const char*);

private:
    const char* name;
};
```

Օրինակ

```
class FloppyDisk : public Equipment {

public:
    FloppyDisk(const char*);
    virtual ~FloppyDisk();
    virtual Watt Power();
    virtual Currency NetPrice();
    virtual Currency DiscountPrice();
};
```

Օրինակ

```
class CompositeEquipment : public Equipment {

public:
    virtual ~CompositeEquipment();
    virtual Watt Power();

    virtual Currency NetPrice() {
        std::auto_ptr<Iterator> it(CreateIterator());
        Currency total = 0;

        for (it->First(); !it->IsDone(); it->Next()) {
            total += it->CurrentItem()->NetPrice();
        }
        return total;
    }

    virtual void Add(Equipment*);
    virtual void Remove(Equipment*);
    virtual Iterator* CreateIterator();

protected:
    CompositeEquipment(const char*);

private:
    List equipment;
};
```

Օրինակ

```
class Chassis : public CompositeEquipment {

public:
    Chassis(const char*);
    virtual ~Chassis();
    virtual Watt Power();
    virtual Currency NetPrice();
    virtual Currency DiscountPrice();
};
```

Օրինակ

```
Cabinet* cabinet = new Cabinet("PC Cabinet");
Chassis* chassis = new Chassis("PC Chassis");
cabinet->Add(chassis);

Bus* bus = new Bus("MCA Bus");
bus->Add(new Card("16Mbs Token Ring"));
chassis->Add(bus);

chassis->Add(new FloppyDisk("3.5in Floppy"));

cout << "The net price is " << chassis->NetPrice() << endl;
```

Առնչվող Նախագծման Ձևանմուշները

- Chain of Responsibility
- Decorator
- Flyweight
- Iterator
- Visitor