Նախագծման Ձևանմուշներ: Builder

Հրաչյա Թանդիլյան

2020

Builder

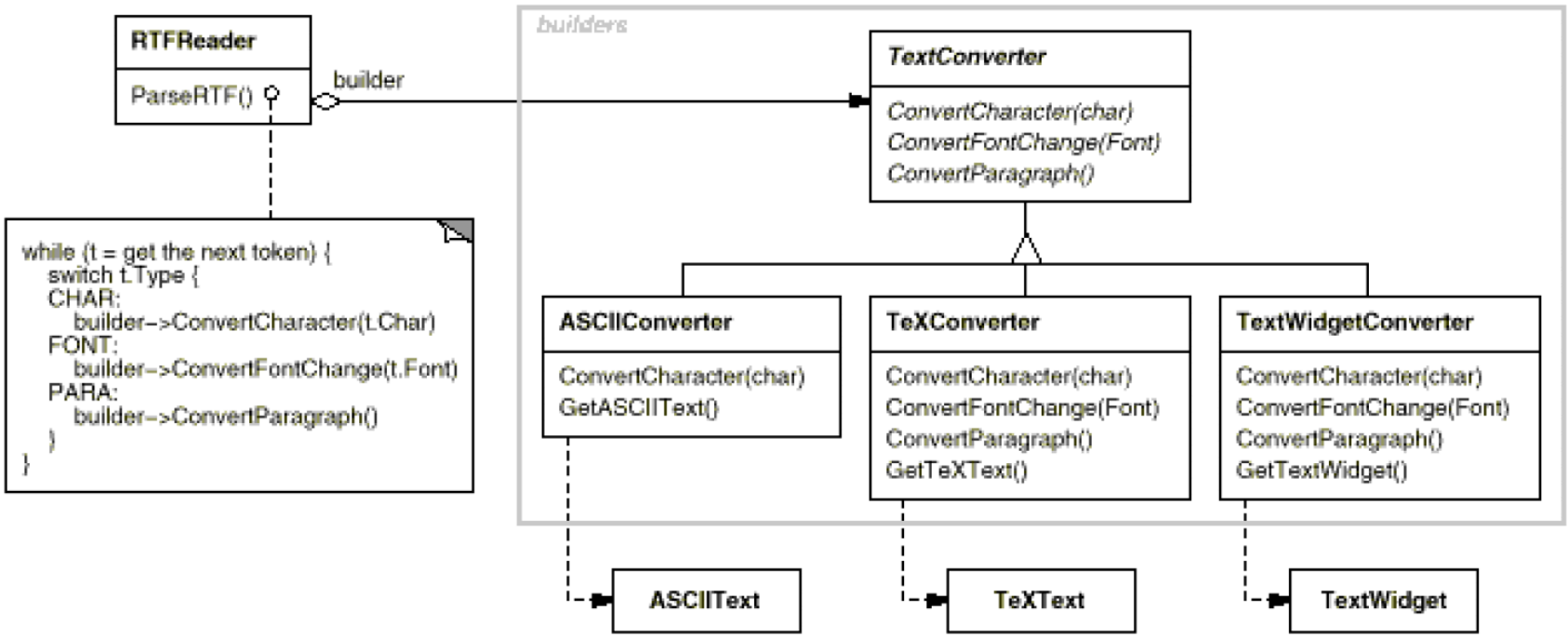
Նպատակը

Առանձնացնում է կոմպլեքս օբյեկտի կառուցումը նրա ներկայացումից այնպես, որ միևնույն կառուցման պրոցեսը հնարավոր լինի կիրառել տարբեր օբյեկտներ ստեղծելու համար:

Նաև հայտնի է որպես

- Այլ լայնորեն կիրառվող անուններ չկան:

Մոտիվացիան


RTFReader
ParseRTF()
builder
while (t = get the next token) {
switch t.Type {
CHAR:
builder->ConvertCharacter(t.Char)
FONT:
builder->ConvertFontChange(t.Font)
PARA:
builder->ConvertParagraph()
}
}
builders
TextConverter
ConvertCharacter(char)
ConvertFontChange(Font)
ConvertParagraph()
ASCIIConverter
ConvertCharacter(char)
GetASCIIText()
TeXConverter
ConvertCharacter(char)
ConvertFontChange(Font)
ConvertParagraph()
GetTeXText()
TextWidgetConverter
ConvertCharacter(char)
ConvertFontChange(Font)
ConvertParagraph()
GetTextWidget()
ASCIIText
TeXText
TextWidget

Կիրառելիությունը

Այս Ն.Ձ. պետք է օգտագործել երբ.

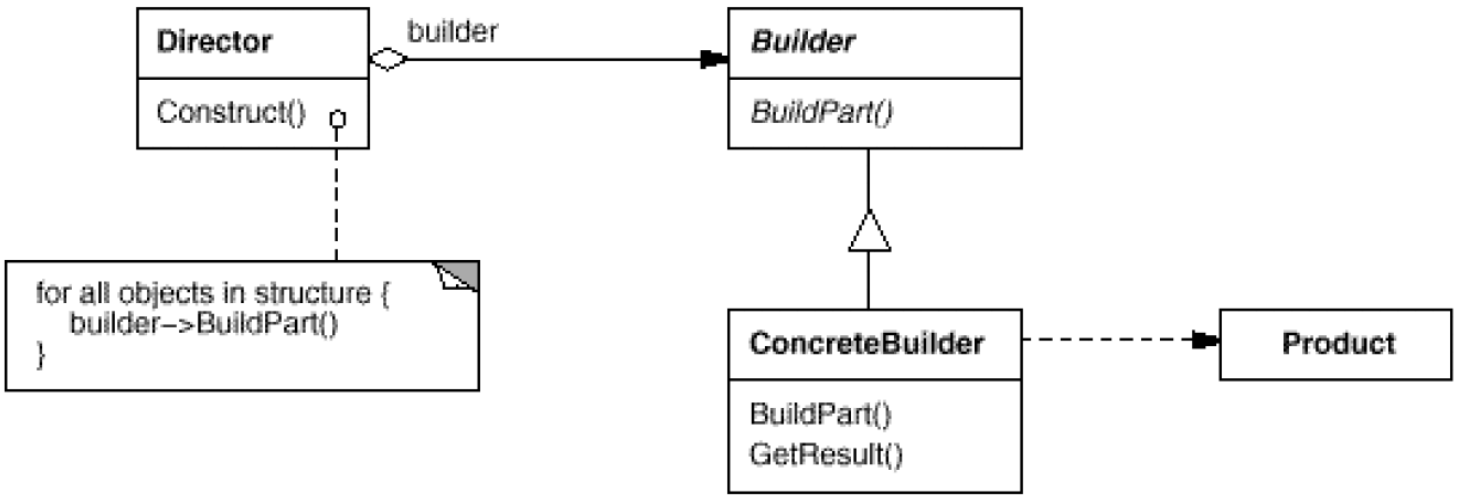
- Կոմպլեքս օբյեկտ ստեղծելու ալգորիթմը պետք է անկախ լինի կառուցվելիք օբյեկտի մասերից, դրանց միավորման ձևից:

Կիրառելիությունը

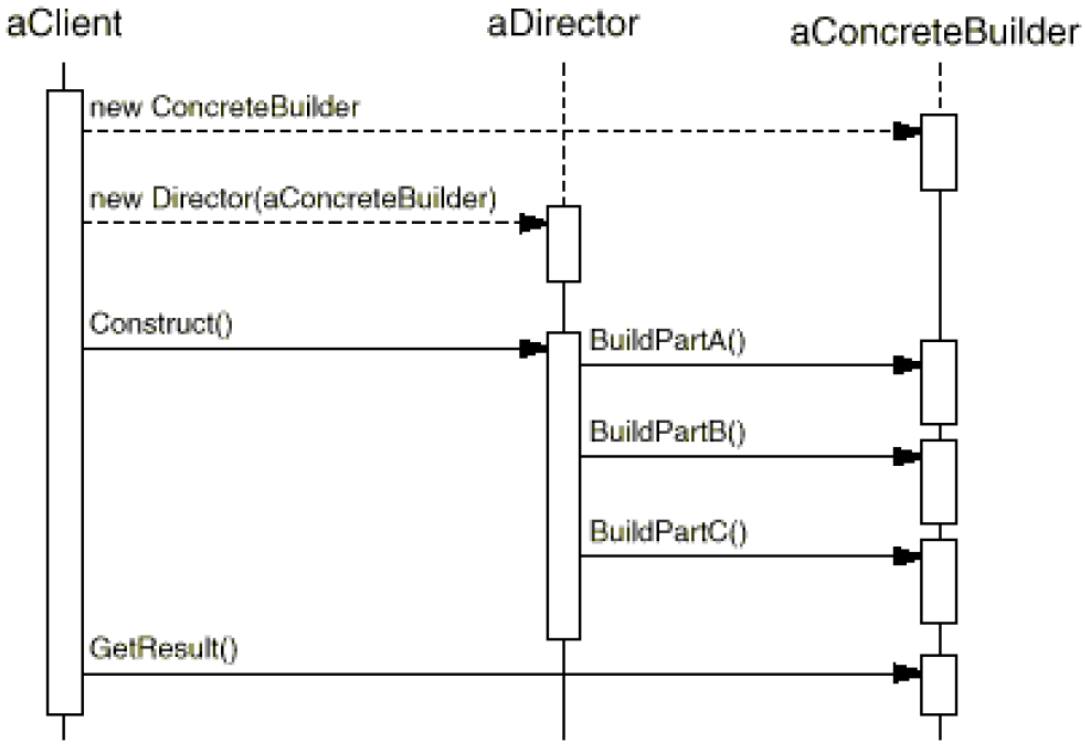
Այս Ն.Ձ. պետք է օգտագործել երբ.

- Ա Կոմպլեքս օբյեկտ ստեղծելու ալգորիթմը պետք է անկախ լինի կառուցվելիք օբյեկտի մասերից, դրանց միավորման ձևից:
- Բ Կառուցման պրոցեսը պետք է թույլ տա կառուցվող օբյեկտի տարբեր ներկայացումներ:

Կառուցվածքը



Փոխհամագործակցությունը



Հետևանքները

Այս Ն.Ձ. ունի հետևյալ առավելություններն ու թերությունները.

- Թույլ է տալիս հեշտորեն փոփոխել օբյեկտների ներքին ներկայացումը:

Հետևանքները

Այս Ն.Ձ. ունի հետևյալ առավելություններն ու թերությունները.

Ա Թույլ է տալիս հեշտորեն փոփոխել օբյեկտների ներքին ներկայացումը:

Բ Առանձնացնում է օբյեկտի կառուցման և ներկայացման կոդերը:

Հետևանքները

Այս Ն.Ձ. ունի հետևյալ առավելություններն ու թերությունները.

Ա Թույլ է տալիս հեշտորեն փոփոխել օբյեկտների ներքին ներկայացումը:

Բ Առանձնացնում է օբյեկտի կառուցման և ներկայացման կոդերը:

Գ Տալիս է կառուցման պրոցեսի ավելի լավ ղեկավարման հնարավորություն:

Իրականացումը

- Կառուցման և միավորման ինտերֆեյս:

Իրականացումը

Ա Կառուցման և միավորման ինտերֆեյս:

Բ Ինչու ստեղծվելիք օբյեկտների համար աբստրակտ բազային դաս չի կիրառվում:

Իրականացումը

Ա. Կառուցման և միավորման ինտերֆեյս:

Բ. Ինչու ստեղծվելիք օբյեկտների համար աբստրակտ բազային դաս չի կիրառվում:

Գ. Դատարկ մեթոդներ աբստրակտ մեթոդների փոխարեն:

Օրինակ

```
class MazeBuilder {

public:
    virtual void BuildMaze() { }

    virtual void BuildRoom(int room) { }

    virtual void BuildDoor(int roomFrom, int roomTo) { }

    virtual Maze* GetMaze() { return 0; }

protected:
    MazeBuilder();
};
```

Օրինակ

```
Maze* MazeGame::CreateMaze(MazeBuilder& builder) {
    builder.BuildMaze();
    builder.BuildRoom(1);
    builder.BuildRoom(2);
    builder.BuildDoor(1, 2);
    return builder.GetMaze();
}
```

Օրինակ

```
Maze* MazeGame::CreateMaze(MazeFactory& factory) {
    Maze* aMaze = factory.MakeMaze();

    Room* r1 = factory.MakeRoom(1);
    Room* r2 = factory.MakeRoom(2);
    aMaze->AddRoom(r1); aMaze->AddRoom(r2);

    Door* aDoor = factory.MakeDoor(r1, r2);
    r1->SetSide(East, aDoor); r2->SetSide(West, aDoor);

    r1->SetSide(North, factory.MakeWall());
    r1->SetSide(South, factory.MakeWall());
    r1->SetSide(West, factory.MakeWall());
    r2->SetSide(North, factory.MakeWall());
    r2->SetSide(East, factory.MakeWall());
    r2->SetSide(South, factory.MakeWall());

    return aMaze;
}
```

Օրինակ

```
class StandardMazeBuilder : public MazeBuilder {

public:
    StandardMazeBuilder() { currentMaze_= NULL; }

    virtual void BuildMaze() { currentMaze_= new Maze; }

    virtual void BuildRoom(int);

    virtual void BuildDoor(int, int);

    virtual Maze* GetMaze() { return currentMaze_; }

private:
    Direction CommonWall(Room*, Room*);
    Maze* currentMaze_;
};
```

Օրինակ

```
void StandardMazeBuilder::BuildRoom(int n) {
    if (currentMaze_->RoomNo(n)) return;

    Room* room = new Room(n);
    currentMaze_->AddRoom(room);

    room->SetSide(North, new Wall); room->SetSide(South, new Wall);
    room->SetSide(East, new Wall); room->SetSide(West, new Wall);
}

void StandardMazeBuilder::BuildDoor(int n1, int n2) {
    Room* r1 = currentMaze_->RoomNo(n1);
    Room* r2 = currentMaze_->RoomNo(n2);

    Door* d = new Door(r1, r2);

    r1->SetSide(CommonWall(r1,r2), d);
    r2->SetSide(CommonWall(r2,r1), d);
}
```

Օրինակ

```
Maze* maze;
MazeGame game;
StandardMazeBuilder builder;
game.CreateMaze(builder);
maze = builder.GetMaze();
```

```
Maze* maze;
MazeGame game;
StandardMazeFactory factory;
maze = game.CreateMaze(factory);
```

Օրինակ

```
Maze* MazeGame::CreateComplexMaze(MazeBuilder& builder) {
    builder.BuildRoom(1);
    builder.BuildRoom(2);
    // ...
    builder.BuildRoom(1001);
    builder.BuildDoor(i1, i2);
    // ...
    return builder.GetMaze();
}
```

Օրինակ

```
class CountingMazeBuilder : public MazeBuilder {

public:
    CountingMazeBuilder(): doors_(0), rooms_(0) {}

    virtual void BuildMaze() { doors_ = rooms_ = 0;}

    virtual void BuildRoom(int) { ++rooms_;}

    virtual void BuildDoor(int, int) { ++doors_;}

    virtual void AddWall(int, Direction){}

    void GetCounts(int& doors, int& rooms) const {
        doors = doors_;  rooms = rooms_;
    }

private:
    int doors_; int rooms_;
};
```

Առնչվող Նախագծման Ձևանմուշները

- Composite