Универзитет „Св. Кирил и Методиј“ - Скопје
ФАКУЛТЕТ ЗА ИНФОРМАТИЧКИ НАУКИ И КОМПЈУТЕРСКО ИНЖЕНЕРСТВО

Аудиториски вежби X*

Наследување и полиморфизам

Објектно ориентирано програмирање

Интервју за работа

Дали сакате да работите во некоја од следните компании?

Прашање на интервју

The classic "eval" interview question

- Прв пат поставено на интервју во Амазон
- Прашањето е доста обемно и содржи многу важни вештини со кои треба да ги поседува еден квалитетен софтверски инженер:
  - **ООП дизајн**
  - **рекурзија**
  - бинарни дрва
  - **полиморфизам**
  - ...

Прашањето
1/2

Во одреден момент, кандидатот конечно сфаќа дека секоја аритметичка операција може да се претстави како бинарно дрво, ако препоставиме дека ги користиме само основните бинарни оператори како $+, -, *, /$. Листовите се броеви, а сите внатрешни јазли се оператори. Евалуацијата на изразот значи изминување на ова дрво.
Не го разбирате ова? Не е важно...
Ова е сеуште интересен проблем.

Прашањето
2/2

Првиот дел од прашањето е како од стринг кој претставува некаков аритметички израз (пр. “5 + (3 * 2)”) го трансформираме во дрво за изразот.
Вториот дел е: да речеме дека ова е проект за двајца луѓе и вашиот партнер е задолжен за трансформацијата на изразот во дрво, а на вас останува лесниот дел. Треба да напишете соодветни класи кои ќе ги употреби вашиот партнер за да ја заврши неговата задача.

Одговорот

Најдоброто решение користи полиморфизам!

Да се обидеме да одговориме на следните прашања?

- Кои се заедничките карактеристики или однесување на различни изрази?
- Кои се разликите?
- Кои методи/полиња треба да се наследат, а кои да се имплементираат посебно?

Expression abstract class (interface)

Решение 1/4

```
class Expression {
public:
    virtual double eval() = 0;

    virtual void print() = 0;

    virtual ~Expression() {
    }
};
```

BinaryExpression abstract class

Решение 2/4

```
class BinaryExpression : public Expression {
protected:
    Expression *left;
    Expression *right;
public:
    BinaryExpression(Expression *left, Expression *right) {
        this->left = left;
        this->right = right;
    }

    ~BinaryExpression() {
        delete left;
        delete right;
    }
};
```

ValueNode class

Решение 3/4

```
class ValueNode : public Expression {
private:
    double value;
public:
    ValueNode(double value) {
        this->value = value;
    }

    double eval() {
        return value;
    }

    void print() {
        cout << value;
    }
};
```

AddExpression class

Решение 4/4

```
class AddExpression : public BinaryExpression {
public:
    AddExpression(Expression *left, Expression *right) : BinaryExpression(left,
         right) {
    }

    double eval() {
        return left->eval() + right->eval();
    }

    void print() {
        left->print();
        cout << " + ";
        right->print();
    }
};
```

Тест

```
int main() {
    Expression *addition = new AddExpression(
        new ValueNode(5),
        new MultiplyExpression(new ValueNode(3), new ValueNode(2))
    );
    addition->print();
    cout << " = " << addition->eval() << endl;
    delete addition;
    return 0;
}
```

Материјали и прашања

Предавања, аудиториски вежби, соопштенија
courses.finki.ukim.mk

Изворен код на сите примери и задачи
https://github.com/tdelev/OOP/tree/master/latex/src

Прашања и дискусија
forum.finki.ukim.mk