

طراحی و پیاده سازی

نرم افزار ساخت نمودار کلاس و تبدیل و اضافه آن به کد زبان سی

استاد راهنما

دکتر محمدرضا رزازی

استاد داور

دکتر امیر کلباسی

دانشجو

امیررضا شیرمست

دلایل برنامه نویسی شیء گرا در زبان سی

دلایل استفاده از زبان سی

- ۱- ممکن نبودن استفاده از کامپایلری دیگر
- ۲- ادامه توسعه بر روی کدهای به زبان سی

دلایل استفاده از الگوی برنامه نویسی شیء گرا

- ۱- مدولار بودن برای عیب یابی آسان تر
- ۲- قابلیت استفاده مجدد از کد
- ۳- انعطاف پذیری
- ۴- امنیت

کارهای پیشین

OOP in C

- ۱- کتاب برنامه نویسی شیء گرا در زبان سی
- ۲- کدهای متعدد که از تکنیک های برنامه نویسی شیء گرا استفاده کرده اند.

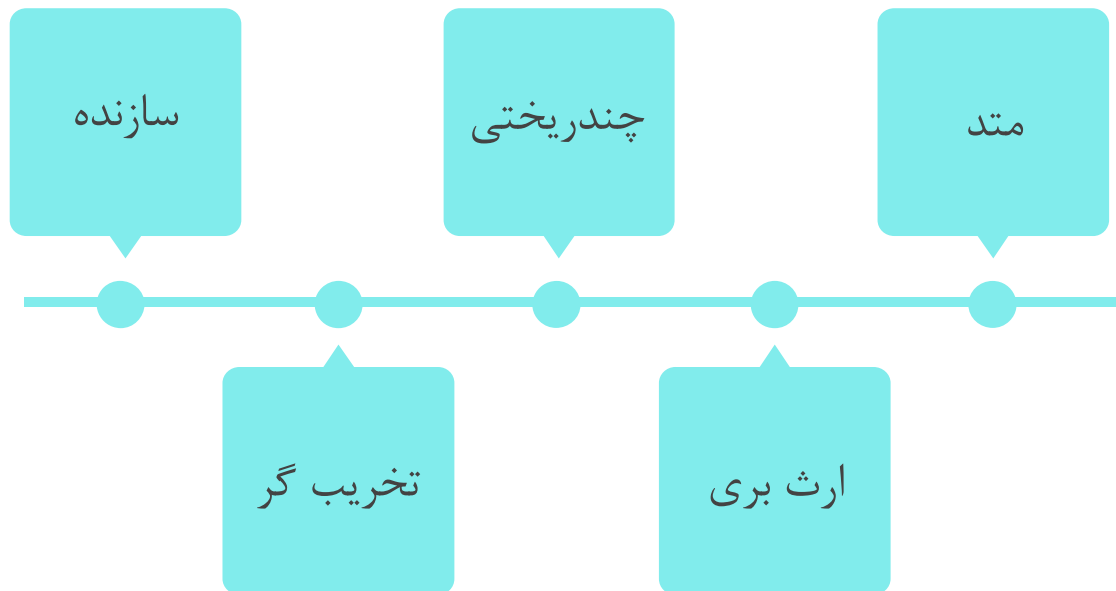
Transcompiler

- 1 - cfront
- 2- Comeau C/C++

UML ↔ Code

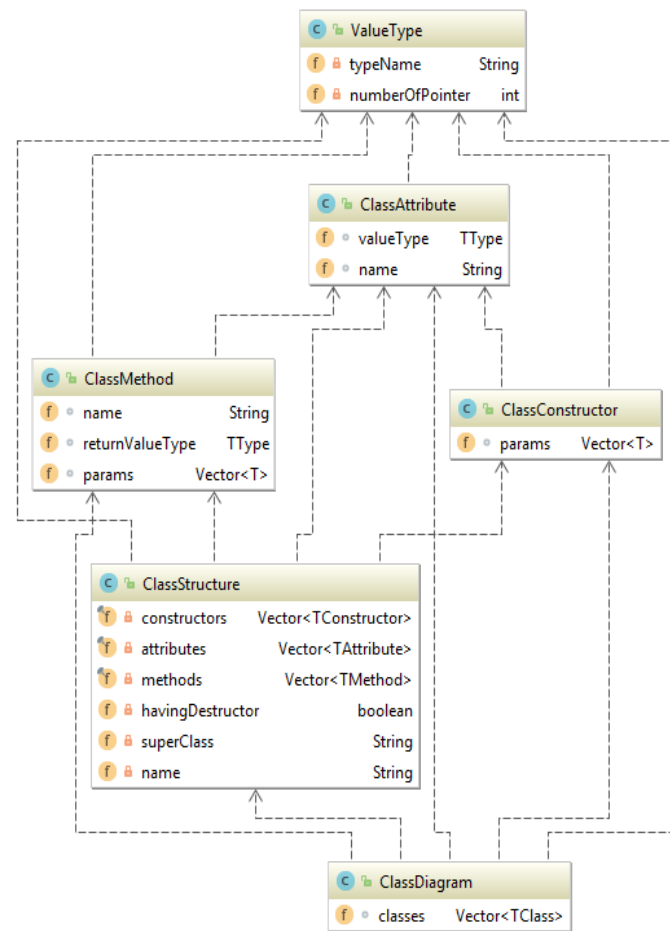
- ۱- نرم افزارهای کمک به طراحی نرم افزار
- ۳- UML2C plugins

قابلیت های پیاده سازی شده



ساختمان داده ها

- 1- ValueType
- 2- ClassAttribute
- 3- ClassConstructor
- 4- ClassMethod
- 5- ClassStructure
- 6- ClassDiagram



ورودی برنامه

XML

```
-<ClassDiagram>
+<Class></Class>
+<Class></Class>
+<Class></Class>
-<Class>
  <name>Student</name>
  <super>Object</super>
  <destructor>>true</destructor>
  -<attributes>
    -<Attribute>
      <name>student_name</name>
    -<Type>
      <typename>class FixedSizeString</typename>
      <number>0</number>
    </Type>
  </Attribute>
  +<Attribute></Attribute>
</attributes>
```

```
-<constructors>
  -<Constructor>
    -<params>
      -<Attribute>
        <name>number_of_professors</name>
      -<Type>
        <typename>int</typename>
        <number>0</number>
      </Type>
    </Attribute>
  </params>
</Constructor>
+<Constructor></Constructor>
</constructors>
-<methods>
  +<Method></Method>
```

```
-<Method>
  <name>set_professor</name>
  -<Type>
    <typename>void</typename>
    <number>0</number>
  </Type>
  -<params>
    -<Attribute>
      <name>number</name>
    -<Type>
      <typename>int</typename>
      <number>0</number>
    </Type>
  </Attribute>
  -<Attribute>
    <name>professor_name</name>
  -<Type>
    <typename>class FixedSizeString</typename>
    <number>0</number>
  </Type>
  </Attribute>
</params>
</Method>
</methods>
</Class>
</ClassDiagram>
```

classes

String

Stream

add

delete

attributes

type name

class name:

Stream

☐ have destructor

☒ super class

super class name:

String

add

delete

constructors

(type name)

add

delete

methods

String* get_string

add

delete

method

int index

method name:

get_string

level of pointer:

1

value type:

String

add

delete

name:

index

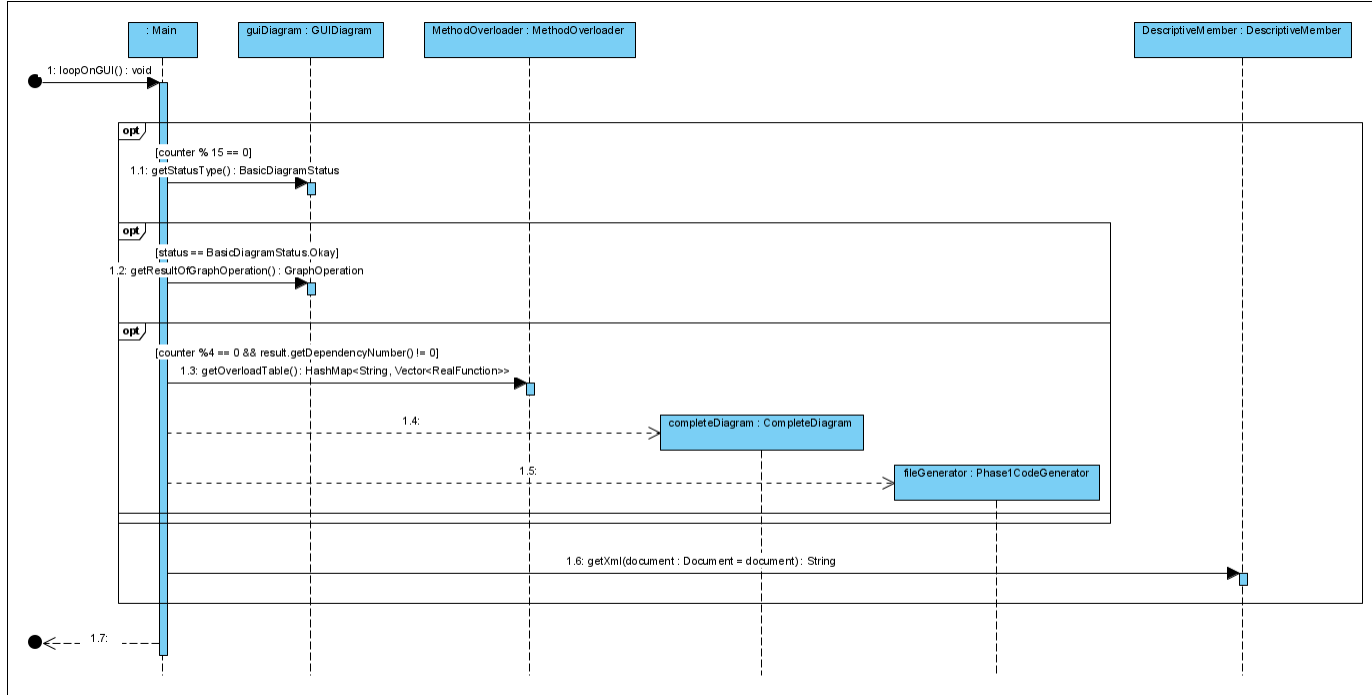
level of pointer:

0

value type:

int

رابط گرافیکی کاربر



بررسی ورودی

خطاهای پایه ای

Same Attribute And Method Name
Same Attribute And Class Name
Same Method And Class Name
Same Attribute Name
Same Class Name

Type Name Error
Attribute Name Error
Method Name Error
Class Name Error
Super Class Name Error

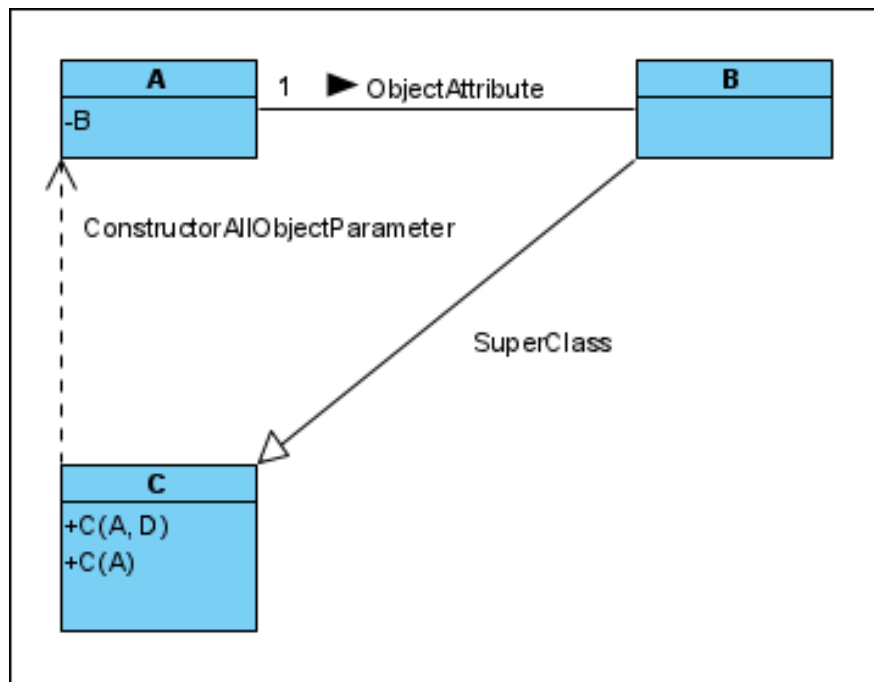
Negative Value Type Pointer
Non Exist Super Class
Same Constructor
Same Method Signature
Same Parameter Name

وابستگی کلاس ها

• وابستگی نوع

• وابستگی نوع ۱

• وابستگی نوع ۲



• یک حلقه با وابستگی نوع

وابستگی کلاس ها

وابستگی نوع ۲

Pointer Attribute
Method Object Type
Method Pointer Type
Method Object Parameter
Method Pointer Parameter
Constructor Object Parameter
Constructor Pointer Parameter

وابستگی نوع ۱

Constructor All Pointer Parameter

Constructor All Hybrid Parameter

وابستگی نوع ۰

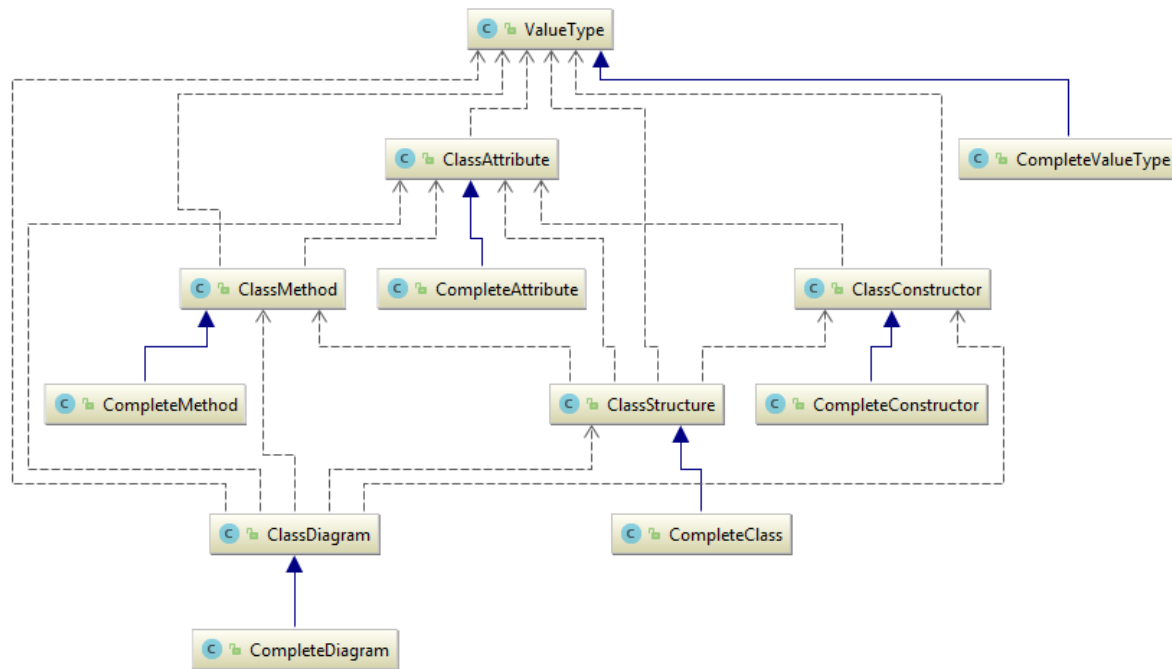
Class Super

Attribute Object

Parameter Object All Constructor

تولید کد در مرحله اول

اعمال ارث بری



ساختار کلاس ها

```
union DefinedClass
{
    union Parent1 unionParent1;
    union Parent2 unionParent2;
    union Parent3 unionParent3;
    .
    .
    .

    struct
    {
        union DefinedClass* this;
        Attribute1Type attribute1Name;
        Attribute2Type attribute2Name;
        Attribute3Type attribute3Name;
        .
        .
        .
    };
};
```

```
union Cube
{
    union Number unionNumber;
    union Square unionSquare;
    struct
    {
        union Cube* this;
        int number;
    };
};
```


ساختار متدها

```
class FixedSizeString get_professor(union Student* this, int number);  
void set_professor(union Student* this, int number, class FixedSizeString professor_name);  
  
void constructorStudent(union Student* this, int number_of_professors);  
union Student* newStudent(int number_of_professors);  
void constructorStudent(union Student* this, class FixedSizeString the_name, int number_of_professors);  
union Student* newStudent(class FixedSizeString the_name, int number_of_professors);  
  
void destructorStudent(union Student* this);  
void delete(union Student* this);
```

پیاده سازی متد ها

```
void Student::set_professor(int number, class FixedSizeString professor_name)
{
    // TODO: code here
}

Student::~~Student()
{
    // TODO: code here
}

char* get_chars(union FixedSizeString* this)
{
    return get_chars(&(this->unionString));
}

union FixedSizeString* newFixedSizeString()
{
    union FixedSizeString* this = (union FixedSizeString*) malloc(sizeof(union FixedSizeString));
    constructorFixedSizeString(this);
    return this;
}
```

دستکاری نام متد ها

```
class FixedSizeString get_professor_0lebQZOyj (union Student* this, int number);  
void set_professor_XJ7DnU162 (union Student* this, int number, class FixedSizeString professor_name);  
  
void constructorStudent_EvYa3CERu (union Student* this, int number_of_professors);  
union Student* newStudent_EvYa3CERu (int number_of_professors);  
void constructorStudent_baf15zIog (union Student* this, class FixedSizeString the_name, int number_of_professors);  
union Student* newStudent_baf15zIog (class FixedSizeString the_name, int number_of_professors);  
  
void destructorStudent (union Student* this);  
void delete_keyword_Od7UHBQEk (union Student* this);
```

سربار گذاری متد ها

```
#define CHOOSE __builtin_choose_expr
#define IFTYPE(X, T) __builtin_types_compatible_p(typeof(X), T)

#define SELECT_1(X1, ...) X1
#define SELECT_2(X1, X2, ...) X2
#define SELECT_3(X1, X2, X3, ...) X3
#define SELECT_4(X1, X2, X3, X4, ...) X4

#define model1(...) \
CHOOSE(IFTYPE(SELECT_1(__VA_ARGS__), union Benz*), model_BUb2bJfj3, \
model_0ZgXXBEWz) \
(__VA_ARGS__)
#define model2(...) \
model_C9PoYoVFF \
(__VA_ARGS__)

#define info1(...) \
CHOOSE(IFTYPE(SELECT_1(__VA_ARGS__), union Benz*), info_0klPTXPmp, \
info_lH0778d69) \
(__VA_ARGS__)

#define SELECT_N(X, _1, _2, _3, _4, N, ...) N

#define model(...) SELECT_N(X, ##__VA_ARGS__, model4, model3, model2, model1, model0)(__VA_ARGS__)
#define info(...) SELECT_N(X, ##__VA_ARGS__, info4, info3, info2, info1, info0)(__VA_ARGS__)
```

تولید کد در مرحله دوم

ترنسکامپایل کد به زبان سی

```
void Student::set_professor_G42ARckI6(int number, class FixedSizeString professor_name)
{
    this->professors[number] = professor_name;
}

Student::~~Student()
{
    printf("%s\n", this->student_name.get_chars());
}

void set_professor_G42ARckI6(union Student* this_keyword, int number, union FixedSizeString professor_name)
{
    this_keyword->this_keyword = this_keyword;
    this_keyword->professors[number] = professor_name;
}

void destructorStudent(union Student* this_keyword)
{
    this_keyword->this_keyword = this_keyword;
    printf("%s\n", get_chars(&(this_keyword->student_name)));
}

}
```

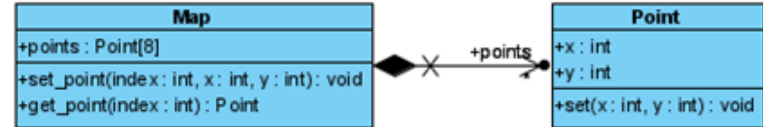
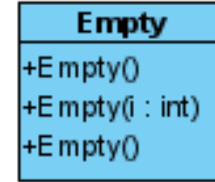
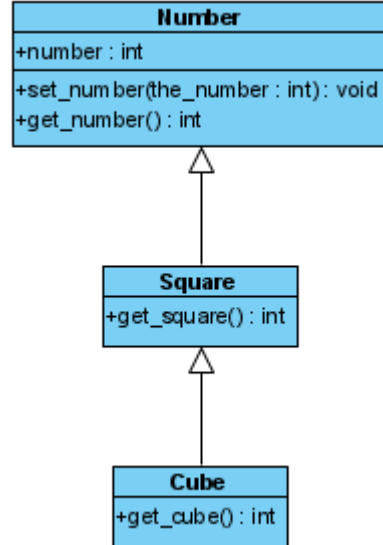
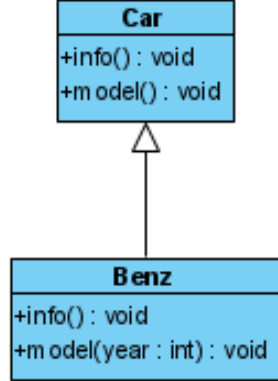
ترنسکامپایل کد به زبان سی

```
void tm(class FixedSizeString pro)
{
    class FixedSizeString m(30);
    m.all = "21";
    pro.set_chars(m.all, 3);
}
```

```
void tm(union FixedSizeString* pro)
{
    union FixedSizeString m = *newFixedSizeString(30);
    m.all = "21";
    set_chars((pro), m.all, 3);
    delete_keyword(&m);
}
```

ارزیابی

آزمون ها



خسته نباشید