Динамическая диспетчеризация

Объектно-Ориентированное Программирование

Иван Трепаков

NSU

Полиморфизм

Полиморфизм — возможность функции с одним именем иметь разные реализации

Полиморфизм

Ad hoc полиморфизм

Ad hoc (букв. «к этому») — латинская фраза, означающая «для данного случая», «специально для этого». *Wikipedia*

Полиморфизм

Ad hoc полиморфизм

Ad hoc (букв. «к этому») — латинская фраза, означающая «для данного случая», «специально для этого». *Wikipedia*

- Выбор реализации делается в зависимости от количества и типов формальных параметров функции
  - Перегрузка функций в Java
  - Перегрузка операторов в C++

Ad hoc полиморфизм

Ad hoc (букв. «к этому») — латинская фраза, означающая «для данного случая», «специально для этого». *Wikipedia*

- Выбор реализации делается в зависимости от количества и типов формальных параметров функции
  - Перегрузка функций в Java
  - Перегрузка операторов в C++

```
var x = ...;
var y = ...;
var z = x + y; // ???
```

Полиморфизм

Ad hoc полиморфизм

Ad hoc (букв. «к этому») — латинская фраза, означающая «для данного случая», «специально для этого». *Wikipedia*

- Выбор реализации делается в зависимости от количества и типов формальных параметров функции
  - Перегрузка функций в Java
  - Перегрузка операторов в C++

```
var x = 1;     // int
var y = ...;
var z = x + y; // ???
```

Ad hoc полиморфизм

Ad hoc (букв. «к этому») — латинская фраза, означающая «для данного случая», «специально для этого». *Wikipedia*

- Выбор реализации делается в зависимости от количества и типов формальных параметров функции
  - Перегрузка функций в Java
  - Перегрузка операторов в C++

```
var x = 1;     // int
var y = 2;     // int
var z = x + y; // ???
```

Ad hoc полиморфизм

Ad hoc (букв. «к этому») — латинская фраза, означающая «для данного случая», «специально для этого». *Wikipedia*

- Выбор реализации делается в зависимости от количества и типов формальных параметров функции
  - Перегрузка функций в Java
  - Перегрузка операторов в C++

```
var x = 1;     // int
var y = 2;     // int
var z = x + y; // 3 (int)
```

Ad hoc полиморфизм

Ad hoc (букв. «к этому») — латинская фраза, означающая «для данного случая», «специально для этого». *Wikipedia*

- Выбор реализации делается в зависимости от количества и типов формальных параметров функции
  - Перегрузка функций в Java
  - Перегрузка операторов в C++

```
var x = 1;     // int
var y = 2.0;   // double
var z = x + y; // ???
```

Ad hoc полиморфизм

Ad hoc (букв. «к этому») — латинская фраза, означающая «для данного случая», «специально для этого». *Wikipedia*

- Выбор реализации делается в зависимости от количества и типов формальных параметров функции
  - Перегрузка функций в Java
  - Перегрузка операторов в C++

```
var x = 1;     // int
var y = 2.0;   // double
var z = x + y; // 3.0 (double)
```

Полиморфизм

Ad hoc полиморфизм

Ad hoc (букв. «к этому») — латинская фраза, означающая «для данного случая», «специально для этого». *Wikipedia*

- Выбор реализации делается в зависимости от количества и типов формальных параметров функции
  - Перегрузка функций в Java
  - Перегрузка операторов в C++

Ad hoc полиморфизм

Ad hoc (букв. «к этому») — латинская фраза, означающая «для данного случая», «специально для этого». *Wikipedia*

- Выбор реализации делается в зависимости от количества и типов формальных параметров функции
  - Перегрузка функций в Java
  - Перегрузка операторов в C++

```
String add(int x, int y) {
  return "ints: " + (x + y);
}
String add(String x, int y) {
  return "mixed: " + (x + y);
}
String add(String x, String y) {
  return "strings: " + (x + y);
}
```

Ad hoc полиморфизм

Ad hoc (букв. «к этому») — латинская фраза, означающая «для данного случая», «специально для этого». *Wikipedia*

- Выбор реализации делается в зависимости от количества и типов формальных параметров функции
  - Перегрузка функций в Java
  - Перегрузка операторов в C++

```
String add(int x, int y) {
  return "ints: " + (x + y);
}
String add(String x, int y) {
  return "mixed: " + (x + y);
}
String add(String x, String y) {
  return "strings: " + (x + y);
}
add(1, 2)
```

Ad hoc полиморфизм

Ad hoc (букв. «к этому») — латинская фраза, означающая «для данного случая», «специально для этого». *Wikipedia*

- Выбор реализации делается в зависимости от количества и типов формальных параметров функции
  - Перегрузка функций в Java
  - Перегрузка операторов в C++

```
String add(int x, int y) {
  return "ints: " + (x + y);
}
String add(String x, int y) {
  return "mixed: " + (x + y);
}
String add(String x, String y) {
  return "strings: " + (x + y);
}
add(1, 2) // ints: 3
```

Ad hoc полиморфизм

Ad hoc (букв. «к этому») — латинская фраза, означающая «для данного случая», «специально для этого». *Wikipedia*

- Выбор реализации делается в зависимости от количества и типов формальных параметров функции
  - Перегрузка функций в Java
  - Перегрузка операторов в C++

```
String add(int x, int y) {
  return "ints: " + (x + y);
}
String add(String x, int y) {
  return "mixed: " + (x + y);
}
String add(String x, String y) {
  return "strings: " + (x + y);
}
add(1, 2) // ints: 3
add("1", 2)
```

Ad hoc полиморфизм

Ad hoc (букв. «к этому») — латинская фраза, означающая «для данного случая», «специально для этого». *Wikipedia*

- Выбор реализации делается в зависимости от количества и типов формальных параметров функции
  - Перегрузка функций в Java
  - Перегрузка операторов в C++

```
String add(int x, int y) {
  return "ints: " + (x + y);
}
String add(String x, int y) {
  return "mixed: " + (x + y);
}
String add(String x, String y) {
  return "strings: " + (x + y);
}
add(1, 2) // ints: 3
add("1", 2) // mixed: 12
```

Ad hoc полиморфизм

Ad hoc (букв. «к этому») — латинская фраза, означающая «для данного случая», «специально для этого». *Wikipedia*

- Выбор реализации делается в зависимости от количества и типов формальных параметров функции
  - Перегрузка функций в Java
  - Перегрузка операторов в C++

```
String add(int x, int y) {
  return "ints: " + (x + y);
}
String add(String x, int y) {
  return "mixed: " + (x + y);
}
String add(String x, String y) {
  return "strings: " + (x + y);
}
add(1, 2) // ints: 3
add("1", 2) // mixed: 12
add("1", "2")
```

Ad hoc полиморфизм

Ad hoc (букв. «к этому») — латинская фраза, означающая «для данного случая», «специально для этого». *Wikipedia*

- Выбор реализации делается в зависимости от количества и типов формальных параметров функции
  - Перегрузка функций в Java
  - Перегрузка операторов в C++

```
String add(int x, int y) {
  return "ints: " + (x + y);
}
String add(String x, int y) {
  return "mixed: " + (x + y);
}
String add(String x, String y) {
  return "strings: " + (x + y);
}
add(1, 2) // ints: 3
add("1", 2) // mixed: 12
add("1", "2") // strings: 12
```

Параметрический полиморфизм

Параметрический полиморфизм

- Реализация функции использует *обобщенный* параметр
  - Generics в Java
  - Templates в C++
  - Type classes в Haskell
- Можно задавать дополнительные ограничения на обобщенный тип
- *Подробнее на следующей лекции*

Полиморфизм

Параметрический полиморфизм

- Реализация функции использует *обобщенный* параметр
  - Generics в Java
  - Templates в C++
  - Type classes в Haskell
- Можно задавать дополнительные ограничения на обобщенный тип
- *Подробнее на следующей лекции*

```
static <T, S> T foo(T x, S y) {
  return x;
}

static <T extends I> boolean bar(I x, I y) {
  return x.test(y);
}

interface I {
  boolean test(I x);
}
```

Полиморфизм подтипов

- Отношение *подтипа* `S <: T`
  Любое значение типа S является значением типа T

Полиморфизм подтипов

- Отношение *подтипа* `S <: T`
  Любое значение типа S является значением типа T
- В объектно-ориентированных языках отношение подтипа обычно совпадает с отношением наследования

Полиморфизм подтипов

- Отношение *подтипа* `S <: T`
  Любое значение типа S является значением типа T
- В объектно-ориентированных языках отношение подтипа обычно совпадает с отношением наследования
- Выбор реализации делается в зависимости от реального типа объекта *receiver* в момент исполнения вызова
  - *Виртуальные* методы
  - *Переопределение* методов

Полиморфизм подтипов

- Отношение *подтипа* `S <: T`
  Любое значение типа S является значением типа T
- В объектно-ориентированных языках отношение подтипа обычно совпадает с отношением наследования
- Выбор реализации делается в зависимости от реального типа объекта *receiver* в момент исполнения вызова
  - *Виртуальные* методы
  - *Переопределение* методов

```
class A {
  void foo() { println(”A.foo”); }
}
class B extends A {
  void foo() { println(”B.foo”); }
}
class C extends A { }
```

Полиморфизм подтипов

- Отношение *подтипа* `S <: T`
  Любое значение типа S является значением типа T
- В объектно-ориентированных языках отношение подтипа обычно совпадает с отношением наследования
- Выбор реализации делается в зависимости от реального типа объекта *receiver* в момент исполнения вызова
  - *Виртуальные* методы
  - *Переопределение* методов

```
class A {
  void foo() { println(”A.foo”); }
}
class B extends A {
  void foo() { println(”B.foo”); }
}
class C extends A { }

A x = ...;
x.foo();   // ???
```

Полиморфизм подтипов

- Отношение *подтипа* S <: T
  Любое значение типа S является значением типа T
- В объектно-ориентированных языках отношение подтипа обычно совпадает с отношением наследования
- Выбор реализации делается в зависимости от реального типа объекта *receiver* в момент исполнения вызова
  - *Виртуальные* методы
  - *Переопределение* методов

```
class A {
  void foo() { println(”A.foo”); }
}
class B extends A {
  void foo() { println(”B.foo”); }
}
class C extends A { }

A x = ...; // new A()
x.foo();   // ???
```

Полиморфизм

Полиморфизм подтипов

- Отношение *подтипа* `S <: T`
  Любое значение типа S является значением типа T
- В объектно-ориентированных языках отношение подтипа обычно совпадает с отношением наследования
- Выбор реализации делается в зависимости от реального типа объекта *receiver* в момент исполнения вызова
  - *Виртуальные* методы
  - *Переопределение* методов

```
class A {
  void foo() { println(”A.foo”); }
}
class B extends A {
  void foo() { println(”B.foo”); }
}
class C extends A { }

A x = ...; // new A()
x.foo();   // A.foo
```

Полиморфизм подтипов

- Отношение *подтипа* S <: T
  Любое значение типа S является значением типа T
- В объектно-ориентированных языках отношение подтипа обычно совпадает с отношением наследования
- Выбор реализации делается в зависимости от реального типа объекта *receiver* в момент исполнения вызова
  - *Виртуальные* методы
  - *Переопределение* методов

```
class A {
  void foo() { println(”A.foo”); }
}
class B extends A {
  void foo() { println(”B.foo”); }
}
class C extends A { }

A x = ...; // new B()
x.foo();   // ???
```

Полиморфизм подтипов

- Отношение *подтипа* S <: T
  Любое значение типа S является значением типа T
- В объектно-ориентированных языках отношение подтипа обычно совпадает с отношением наследования
- Выбор реализации делается в зависимости от реального типа объекта *receiver* в момент исполнения вызова
  - *Виртуальные* методы
  - *Переопределение* методов

```
class A {
  void foo() { println(”A.foo”); }
}
class B extends A {
  void foo() { println(”B.foo”); }
}
class C extends A { }

A x = ...; // new B()
x.foo();   // B.foo
```

Полиморфизм подтипов

- Отношение *подтипа* S <: T
  Любое значение типа S является значением типа T
- В объектно-ориентированных языках отношение подтипа обычно совпадает с отношением наследования
- Выбор реализации делается в зависимости от реального типа объекта *receiver* в момент исполнения вызова
  - *Виртуальные* методы
  - *Переопределение* методов

```
class A {
  void foo() { println(”A.foo”); }
}
class B extends A {
  void foo() { println(”B.foo”); }
}
class C extends A { }

A x = ...; // new C()
x.foo();   // ???
```

Полиморфизм подтипов

- Отношение *подтипа* S <: T
  Любое значение типа S является значением типа T
- В объектно-ориентированных языках отношение подтипа обычно совпадает с отношением наследования
- Выбор реализации делается в зависимости от реального типа объекта *receiver* в момент исполнения вызова
  - *Виртуальные* методы
  - *Переопределение* методов

```
class A {
  void foo() { println(”A.foo”); }
}
class B extends A {
  void foo() { println(”B.foo”); }
}
class C extends A { }

A x = ...; // new C()
x.foo();   // A.foo
```

Полиморфизм

Полиморфизм — возможность функции с одним именем иметь разные реализации

Полиморфизм

Полиморфизм — возможность функции с одним именем иметь разные реализации
Диспетчеризация — процесс выбора реализации полиморфной функции

Диспетчеризация

Диспетчеризация

Статическая диспетчеризация

- Реализация известна до исполнения
  - Прямой вызов (например статический)
  - Перегрузка
  - *Девиртуализация* в компиляторе
- Эффективно без поддержки среды исполнения

Диспетчеризация

Статическая диспетчеризация

- Реализация известна до исполнения
  - Прямой вызов (например статический)
  - Перегрузка
  - *Девиртуализация* в компиляторе
- Эффективно без поддержки среды исполнения

```
C.foo()  ⟶  call &C::foo()
```

Диспетчеризация

Статическая диспетчеризация

- Реализация известна до исполнения
  - Прямой вызов (например статический)
  - Перегрузка
  - *Девиртуализация* в компиляторе
- Эффективно без поддержки среды исполнения

Динамическая диспетчеризация

- Реализация известна только во время исполнения конкретного вызова
  - Виртуальный вызов
  - Интерфейсный вызов
- Требует поддержки среды исполнения

```
C.foo()  ⟶  call &C::foo()
```

Диспетчеризация

Статическая диспетчеризация

- Реализация известна до исполнения
  - Прямой вызов (например статический)
  - Перегрузка
  - *Девиртуализация* в компиляторе
- Эффективно без поддержки среды исполнения

Динамическая диспетчеризация

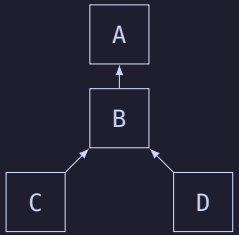
- Реализация известна только во время исполнения конкретного вызова
  - Виртуальный вызов
  - Интерфейсный вызов
- Требует поддержки среды исполнения

`C.foo()` ⟶ `call &C::foo()`

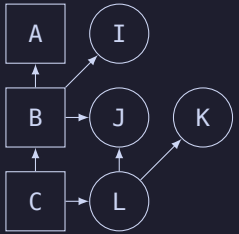
`x.foo()` ⟶ `???`

Наследование

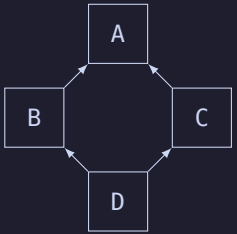
Одиночное



Интерфейсное



Множественное



simula

Java

OCaml

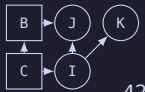
Eiffel

Диспетчеризация виртуальных методов

Диспетчеризация виртуальных методов

Таблица виртуальных методов

Virtual Method Table, VMT, vtable

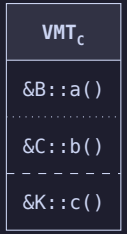


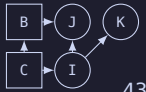
Диспетчеризация виртуальных методов

Таблица виртуальных методов

Virtual Method Table, VMT, vtable

- Массив с адресами реализаций виртуальных методов класса



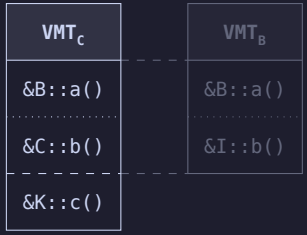


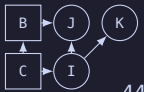
Диспетчеризация виртуальных методов

Таблица виртуальных методов

Virtual Method Table, VMT, vtable

- Массив с адресами реализаций виртуальных методов класса
- Таблицы подклассов расширяют таблицу суперкласса



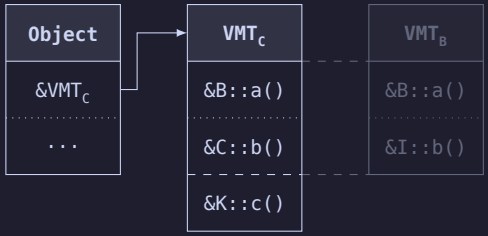


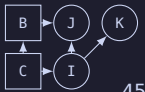
Диспетчеризация виртуальных методов

Таблица виртуальных методов

Virtual Method Table, VMT, vtable

- Массив с адресами реализаций виртуальных методов класса
- Таблицы подклассов расширяют таблицу суперкласса
- Таблица доступна напрямую из каждого объекта класса



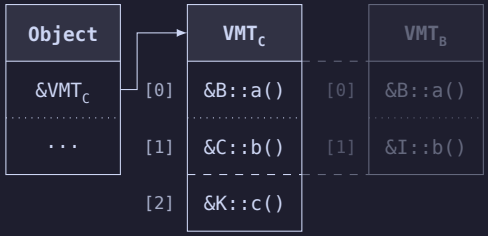


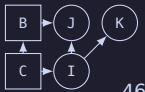
Диспетчеризация виртуальных методов

Таблица виртуальных методов

Virtual Method Table, VMT, vtable

- Массив с адресами реализаций виртуальных методов класса
- Таблицы подклассов расширяют таблицу суперкласса
- Таблица доступна напрямую из каждого объекта класса
- Для каждого виртуального метода можно определить *виртуальный номер*





Диспетчеризация виртуальных методов

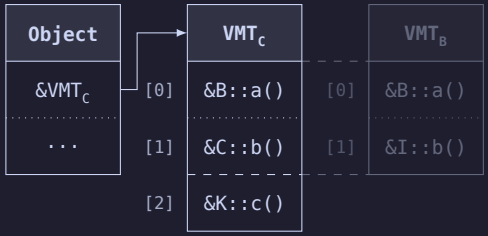
Таблица виртуальных методов

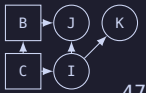
Virtual Method Table, VMT, vtable

- Массив с адресами реализаций виртуальных методов класса
- Таблицы подклассов расширяют таблицу суперкласса
- Таблица доступна напрямую из каждого объекта класса
- Для каждого виртуального метода можно определить *виртуальный номер*

Виртуальный вызов `x.b()`

```
// Формальный тип x - класс B
call x.vmt[vnum_B,b]
```

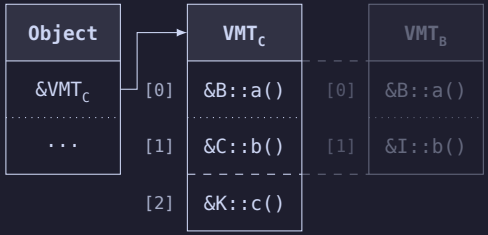


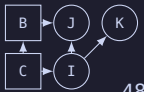


Диспетчеризация интерфейсных методов

Таблица интерфейсных методов

Interface Method Table, IMT, itable



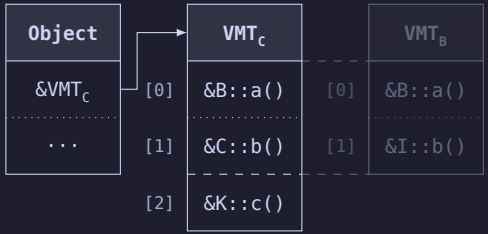


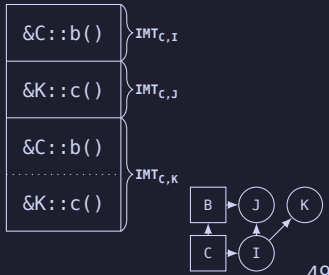
Диспетчеризация интерфейсных методов

Таблица интерфейсных методов

Interface Method Table, IMT, itable

- Массив с адресами реализаций методов интерфейса



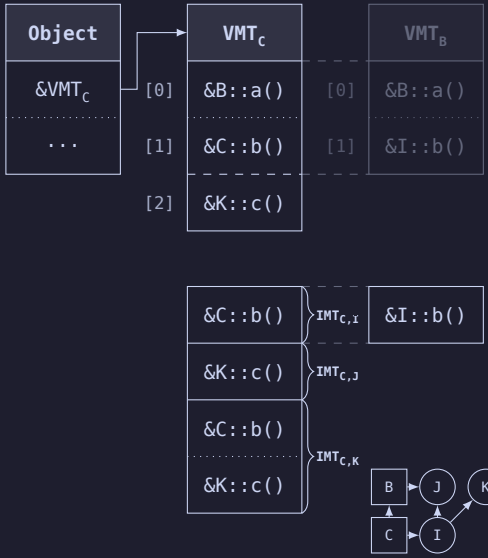


Диспетчеризация интерфейсных методов

Таблица интерфейсных методов

Interface Method Table, IMT, itable

- Массив с адресами реализаций методов интерфейса
- Раскладка фиксирована во всех реализующих классах

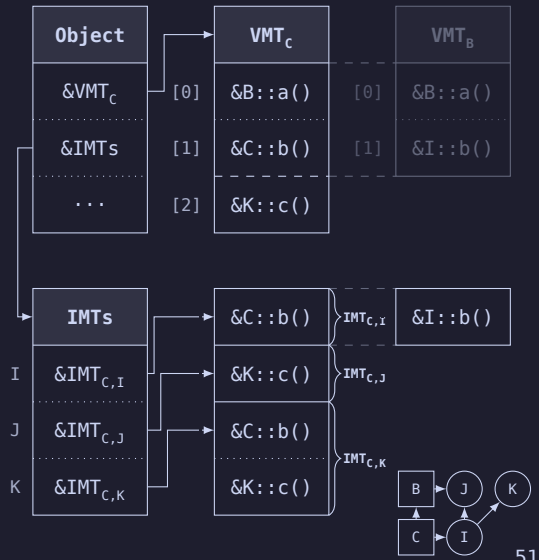


Диспетчеризация интерфейсных методов

Таблица интерфейсных методов

Interface Method Table, IMT, itable

- Массив с адресами реализаций методов интерфейса
- Раскладка фиксирована во всех реализующих классах
- Таблицы доступны из каждого объекта

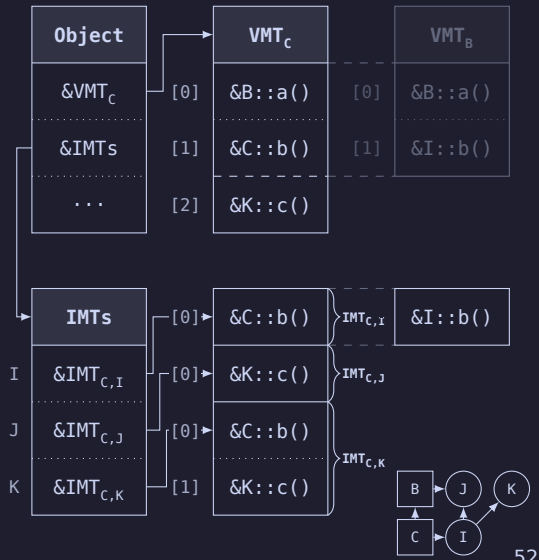


Диспетчеризация интерфейсных методов

Таблица интерфейсных методов

Interface Method Table, IMT, itable

- Массив с адресами реализаций методов интерфейса
- Раскладка фиксирована во всех реализующих классах
- Таблицы доступны из каждого объекта
- Для каждого интерфейсного метода можно определить *виртуальный номер*

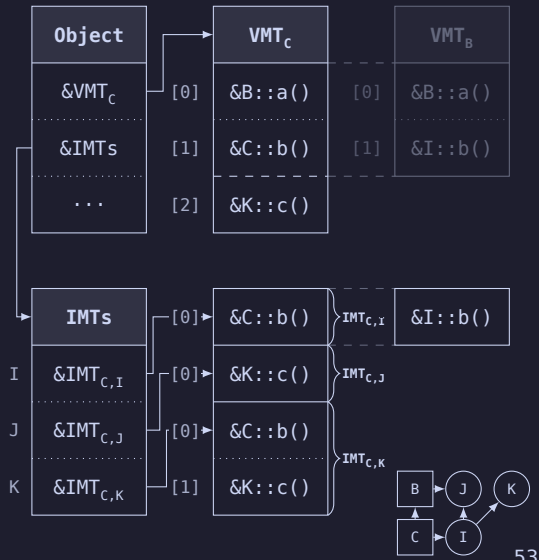


Диспетчеризация интерфейсных методов

Таблица интерфейсных методов

Interface Method Table, IMT, itable

- Массив с адресами реализаций методов интерфейса
- Раскладка фиксирована во всех реализующих классах
- Таблицы доступны из каждого объекта
- Для каждого интерфейсного метода можно определить *виртуальный номер*
- Необходим поиск нужной таблицы



Диспетчеризация интерфейсных методов

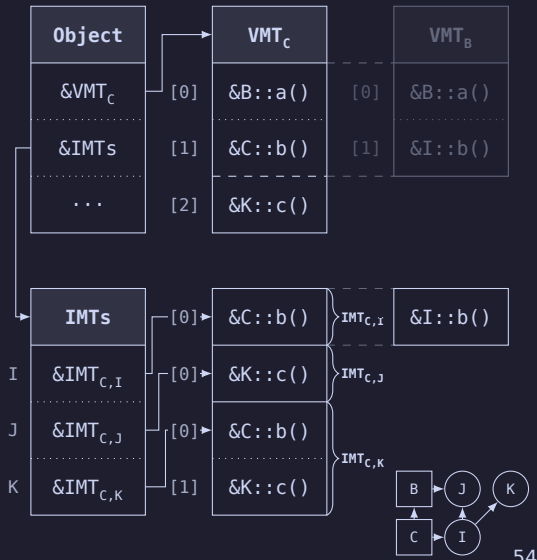
Таблица интерфейсных методов

Interface Method Table, IMT, itable

- Массив с адресами реализаций методов интерфейса
- Раскладка фиксирована во всех реализующих классах
- Таблицы доступны из каждого объекта
- Для каждого интерфейсного метода можно определить *виртуальный номер*
- Необходим поиск нужной таблицы

Интерфейсный вызов x.b()

```
// Формальный тип x - интерфейс I
imt_C,I = x.imts.find(&I)
call imt_C,I[vnum_I,b]
```



Диспетчеризация интерфейсных методов

Полиморфный инлайн кэш

Заключение

Q&A