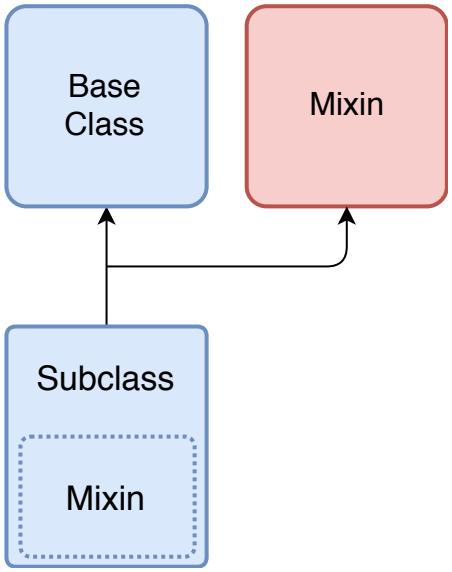
FORGOTTEN TRAITS

Михаил Розумянский

ПРИМЕСЬ (MIXIN)



TRAIT

- Отсутствие состояния
- Ограничение базового класса
- Переопределение методов базового класса
- Вызов методов базового класса
- Модификаторы доступа

ОБЪЯВЛЕНИЕ

```
trait CollectionWithComparator<E : Comparable<E>> :
    AbstractCollection<E> {

  protected val comparator: Comparator<E>

  override fun contains(element: E): Boolean {
    return any { comparator.compare(it, element) == 0 }
  }
}
```

ИСПОЛЬЗОВАНИЕ

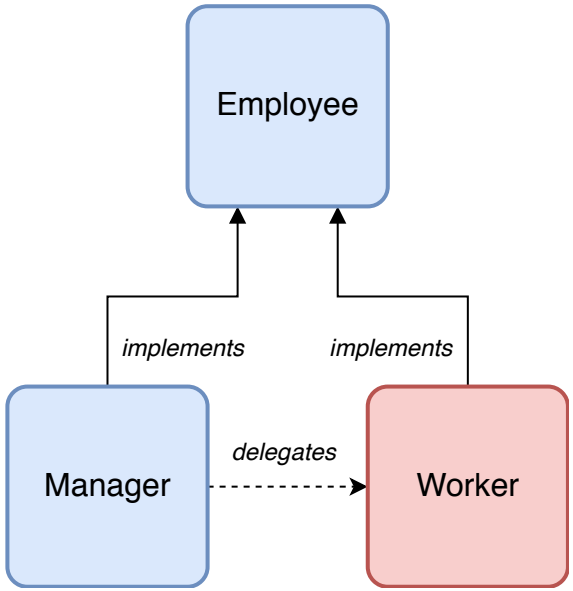
```
class CaseInsensitiveStringCollection :
    AbstractCollection<String>(),
    CollectionWithComparator<String> {

  override val comparator = String.CASE_INSENSITIVE_ORDER

  // ...
}
```

ЧЕМ ЗАМЕНИТЬ?

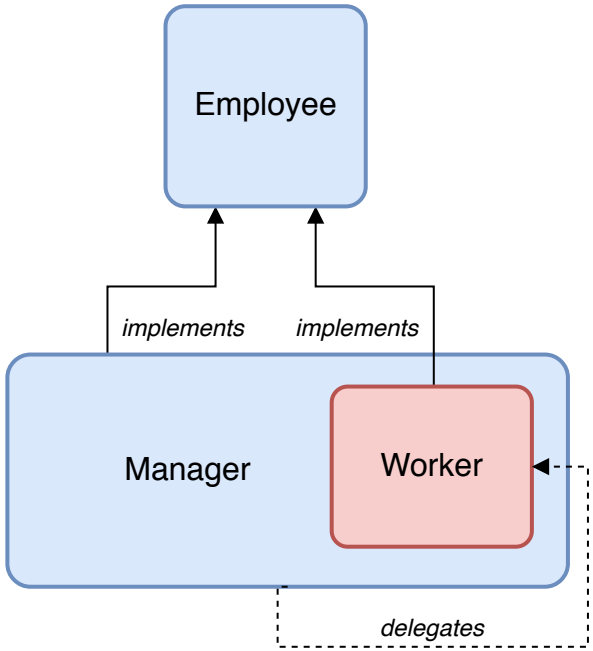
НИЧЕМ

А НА САМОМ ДЕЛЕ?

ЗАДАЧА


Employee
implements
implements
Manager
delegates
Worker

ДЕЛЕГАТ


Employee
implements
implements
Manager
Worker
delegates

СОТРУДНИК

```
interface Employee {
  fun performTask(task: String)
}
```

```
abstract class AbstractEmployee(val name: String) : Employee {
  fun say(text: String, vararg arguments: String) {
    Output.write(name, text, *arguments)
  }
}
```

JAVA WAY

WORKER

```
public class Worker extends AbstractEmployee {
  public Worker(final String name) {
    super(name);
  }

  @Override
  public void performTask(final String task) {
    say("I had to %s and I've just done it.", task);
  }
}
```

MANAGER

```
public class Manager extends AbstractEmployee {
  private final Employee delegate = new Worker("Dilbert");

  public Manager(final String name) { super(name); }

  @Override
  public void performTask(@NotNull final String task) {
    delegate.performTask(task);
  }

  public void manageTask(@NotNull final String task) {
    say("Hey, %s.", task);
    performTask(task);
    say("It's just a piece of cake to %s.", task);
  }
}
```

MAIN

```
public class Demo {
  public static void main(final String[] args) {
    final Manager manager = new Manager("Boss");
    for (final String task : args) {
      manager.performTask(task);
    }
  }
}
```

Boss
Hey, prepare a presentation.

Dilbert
I had to prepare a presentation and I've just done it.

Boss
It's just a piece of cake to prepare a presentation.

KOTLIN WAY

WORKER

```
class Worker(name: String) : AbstractEmployee(name) {
  override fun performTask(task: String) {
    say("I had to $task and I've just done it.")
  }
}
```

MANAGER

```
class Manager(
    name: String
) : AbstractEmployee(name),
    Employee by Worker("Dilbert") {
  fun manageTask(task: String) {
    say("Hey, $task.")
    performTask(task)
    say("It's just a piece of cake to $task.")
  }
}
```

ЧТО ВНУТРИ?

```
public final class Manager extends AbstractEmployee {
  private final Worker $$delegate_0 = new Worker("Dilbert");

  public Manager(@NotNull String name) { super(name); }

  public void performTask(@NotNull String task) {
    $$delegate_0.performTask(task);
  }

  public final void manageTask(@NotNull String task) {
    say("Hey, " + task + '.');
    performTask(task);
    say("It's just a piece of cake to " + task + '.');
  }
}
```

ДОСТУП К ДЕЛЕГАТУ

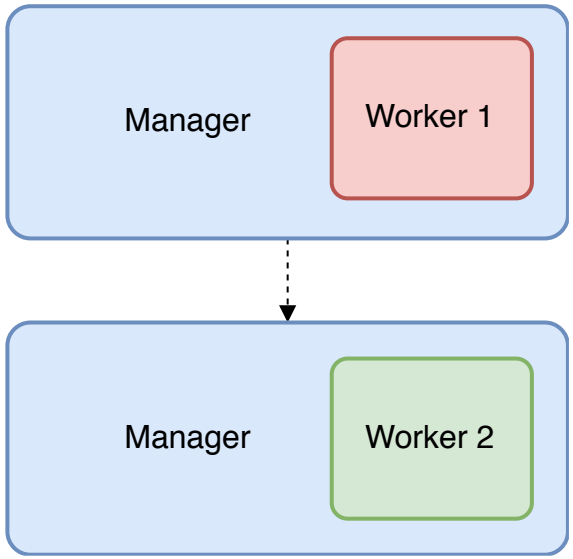
НОВЫЙ MANAGER

```
class Manager private constructor(
    name: String,
    private val delegate: Employee
) : AbstractEmployee(name), Employee by delegate {
  constructor(name: String) : this(name, Worker("Dilbert"))

  override fun performTask(task: String) {
    say("Hey, $task.")
    delegate.performTask(task)
    say("It's just a piece of cake to $task.")
  }
}
```

МОГЛО БЫ БЫТЬ И ЛУЧШЕ

- KT-83 — Distinguish syntactically when delegating to a property
- KT-18427 — Provide a way to reference a delegate in class body

ИЗМЕНЕНИЕ ДЕЛЕГАТА



НАИВНЫЙ ПОДХОД

```
class Manager private constructor(
    name: String,
    private var delegate: Employee
) : AbstractEmployee(name), Employee by delegate {
  constructor(name: String) : this(name, Worker(Names.next()))

  fun manageTask(task: String) {
    say("Hey, $task.")
    performTask(task)
    say("It's just a piece of cake to $task.")
    fireAndHire()
  }

  private fun fireAndHire() {
    say("Seems I need to fire this guy and hire someone else.")
    delegate = Worker(Names.next())
  }
}
```

Boss
Hey, prepare a presentation.

Dilbert
I had to prepare a presentation and I've just done it.

Boss
It's just a piece of cake to prepare a presentation.
Seems I need to fire this guy and hire someone else.
Hey, write a speech.

Dilbert
I had to write a speech and I've just done it.

Boss
It's just a piece of cake to write a speech.
Seems I need to fire this guy and hire someone else.

НО ПОЧЕМУ?

ДЕКОМПИЛИРУЕМ

```
public final class Manager extends AbstractEmployee {
  private Employee delegate;
  private final Employee $$delegate_0;

  private Manager(String name, Employee delegate) {
    super(name);
    $$delegate_0 = delegate;
    delegate = delegate;
  }

  public Manager(@NotNull String name) {
    this(name, new Worker(Names.INSTANCE.next());
  }

  public void performTask(@NotNull String task) {
    $$delegate_0.performTask(task);
  }

  private final void fireAndHire() {
    say("Seems I need to fire this guy and hire someone else.");
    delegate = new Worker(Names.INSTANCE.next());
  }
}
```

ОКАЗЫВАЕТСЯ

Для делегата всегда создаётся поле

- KT-5870 — Delegation to var field

И ЧТО ЖЕ ДЕЛАТЬ?

ОЧЕРЕДНОЙ MANAGER

```
class Manager(name: String) : AbstractEmployee(name) {
  private var delegate = Worker(Names.next())

  override fun performTask(task: String) {
    say("Hey, $task.")
    delegate.performTask(task)
    say("It's just a piece of cake to $task.")
    fireAndHire()
  }

  private fun fireAndHire() {
    say("Seems I need to fire this guy and hire someone else.")
    delegate = Worker(Names.next())
  }
}
```

ПРЯМО КАК В JAVA 😢

```
public class Manager extends AbstractEmployee {
  private final Employee delegate = new Worker(Names.INSTANCE.next());

  public Manager(final String name) { super(name); }

  @Override public void performTask(task: String) {
    say("Hey, %s.", task);
    delegate.performTask(task);
    say("It's just a piece of cake to %s.", task);
    fireAndHire();
  }

  private void fireAndHire() {
    say("Seems I need to fire this guy and hire someone else.");
    delegate = Worker(Names.INSTANCE.next());
  }
}
```

ИНТЕРФЕЙСЫ С ЧАСТИЧНОЙ РЕАЛИЗАЦИЕЙ

```
interface DelegatingEmployee : Employee {
  val delegate: Employee

  override fun performTask(task: String) {
    delegate.performTask(task)
  }
}
```

КАК РАБОТАЮТ?

```
public interface DelegatingEmployee extends Employee {
  @NotNull Employee getDelegate();

  void performTask(@NotNull String var1);

  final class DefaultImpls {
    public static void performTask(
        @NotNull DelegatingEmployee self, String task) {
      self.getDelegate().performTask(task);
    }
  }
}
```

ВТОРАЯ ПОПЫТКА

```
class Manager(
    name: String
) : AbstractEmployee(name), DelegatingEmployee {
  override var delegate: Employee = Worker(Names.next())

  override fun performTask(task: String) {
    say("Hey, $task.")
    super.performTask(task)
    say("It's just a piece of cake to $task.")
    fireAndHire()
  }

  private fun fireAndHire() {
    say("Seems I need to fire this guy and hire someone else.")
    delegate = Worker(Names.next())
  }
}
```

Boss
Hey, prepare a presentation.

Dilbert
I had to prepare a presentation and I've just done it.

Boss
It's just a piece of cake to prepare a presentation.
Seems I need to fire this guy and hire someone else.
Hey, write a speech.

Wally
I had to write a speech and I've just done it.

Boss
It's just a piece of cake to write a speech.
Seems I need to fire this guy and hire someone else.

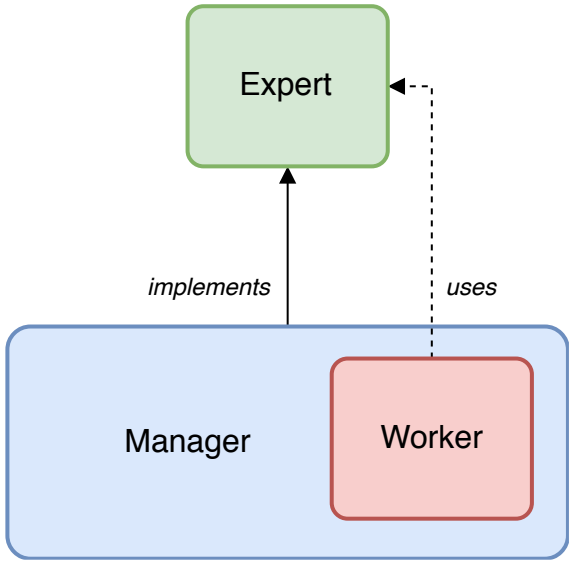
ПРЯМО КАК В JAVA 8 😢

```
public class Manager extends AbstractEmployee, DelegatingEmployee {
  private final String name;
  private Employee delegate = new Worker(Names.INSTANCE.next());

  public Manager(final String name) { super(name); }

  @Override public Employee getDeleagate() { return delegate; }

  @Override public void performTask(final String task) {
    say("Hey, %s.", task);
    super.performTask(task);
    say("It's just a piece of cake to %s.", task);
    fireAndHire();
  }

  private void fireAndHire() {
    say("Seems I need to fire this guy and hire someone else.");
    delegate = Worker(Names.INSTANCE.next());
  }
}
```

ТОЛЬКО НЕМНОГО ХУЖЕ 😭

- KT-4779 — Generate default methods for implementations in interfaces
- KT-19415 — Introduce JvmDefault annotation

ДЕЛЕГАТ ДЕЛЕГАТА



ПОЯВЛЯЕТСЯ ЭКСПЕРТ

```
interface Expert {
  fun answer(question: String)
}
```

РАБОТНИК СПРАШИВАЕТ ЭКСПЕРТА

```
class Worker(
    name: String,
    private val expert: Expert
) : AbstractEmployee(name) {
  override fun performTask(task: String) {
    ask("What is the theme?")
    ask("When is the deadline?")
    say("I had to $task and I've just done it.")
  }

  private fun ask(question: String) {
    say(question)
    expert.answer(question)
  }
}
```

ОЧЕВИДНОЕ РЕШЕНИЕ

```
class Manager(
    name: String
) : AbstractEmployee(name),
    Expert,
    Employee by Worker("Dilbert", this) {
  override fun performTask(task: String) {
    say("Hey, $task.")
    super.performTask(task)
    say("It's just a piece of cake to $task.")
  }

  override fun answer(question: String) {
    say(MotivationalQuotes.random())
  }
}
```

НЕ РАБОТАЕТ

Error: 'this' is not defined in this context

- KT-13603 — Allow passing this to a class delegate

ИСПРАВЛЯЕМ

```
class Manager(
    name: String
) : AbstractEmployee(name), DelegatingEmployee, Expert {
  override val delegate: Employee = Worker("Dilbert", this)

  override fun performTask(task: String) {
    say("Hey, $task.")
    super.performTask(task)
    say("It's just a piece of cake to $task.")
  }

  override fun answer(question: String) {
    say(MotivationalQuotes.random())
  }
}
```

Boss
Hey, prepare a presentation.

Dilbert
What is the theme?

Boss
To win in the marketplace you must first win in the workplace.

Dilbert
When is the deadline?

Boss
If it scares you, it might be a good thing to try.

Dilbert
I had to prepare a presentation and I've just done it.

Boss
It's just a piece of cake to prepare a presentation.

ОПЯТЬ КАК В JAVA 8 😢

```
public class Manager extends AbstractEmployee implements DelegatingEmployee, Expert {
  private final String name;
  private final Employee delegate = new Worker("Dilbert", this);

  public Manager(final String name) { super(name); }

  @Override public Employee getDelegate() { return delegate; }

  @Override public void performTask(final String task) {
    say("Hey, %s.", task);
    super.performTask(task);
    say("It's just a piece of cake to %s.", task);
  }

  @Override public void answer(final String question) {
    say(MotivationalQuotes.INSTANCE.random());
  }
}
```

В РЕЗУЛЬТАТЕ

- Наличие ~~Отсутствие~~ состояния
- Ограничение ~~базового класса~~ использования
- ~~Переопределение методов базового класса~~
- Вызов методов ~~базового класса~~ контейнера
- Модификаторы доступа

СПАСИБО ЗА ВНИМАНИЕ!