Yandex Cloud

За кулисами эволюции Java на примере pattern matching

Олег Естехин
Yandex Cloud
@OlegEstekhin

О чём будет этот доклад

Удовлетворить собственное любопытство

- Как появляются новые фичи?
- Когда появляются новые фичи?
- Почему новые фичи обладают определённым поведением?

О чём будет этот доклад

Удовлетворить собственное любопытство

- Как происходит процесс дизайна языка?
- Какие аргументы используют дизайнеры языка?
- Какими принципами руководствуются дизайнеры языка?

Источники информации

Для сторонних наблюдателей

- JVM Language Summit
- Project Amber

JVM Language Summit

https://openjdk.org/projects/mlvm/jvmlangsummit/

Project Amber

https://openjdk.org/projects/amber/

Принципы

- Ментальная модель
- Рефакторинг
- Композиция

JEP 286: Local-Variable Type Inference

```
var value = "one";
```

JEP 361: Switch Expressions

```
var value = switch (input) {
    case 1  -> "one";
    case 2  -> "two";
    default -> "many";
};
```

```
var value = switch (input) {
    case 1:  yield "one";
    case 2:  yield "two";
    default: yield "many";
};
```

JEP 394: Pattern Matching for instanceof

```
if (input instanceof String str) {
}
```

JEP 395: Records

```
record Box(Object value) {}

record Point(int x, int y) {}
```

JEP 409: Sealed Classes

```
sealed interface Shape permits Rectangle {}

record Rectangle(Point p1, Point p2) implements Shape {}
```

JEP 440: Record Patterns

```
if (input instanceof Point(int x, int y)) {
}
```

JEP 440: Record Patterns

```
if (input instanceof Point(int x, int y)) {
}

if (input instanceof Box(Point point)) {
}

if (input instanceof Box(Point(int x, int y))) {
}
```

JEP 441: Pattern Matching for switch

```
switch (input) {
    case null                                    -> {}
    case Box(Point(int x, int y)) when x == y -> {}
    case Box(Point point)                        -> {}
    case Box box                                 -> {}
    default                                      -> {}
}
```

JEP 443: Unnamed Patterns and Variables

```
if (input instanceof Point(int x, _)) {
}

if (input instanceof Point(int x, int _)) {
}

try {
} catch (Exception _) {
}
```

Project Amber

Pattern Matching

- JEP 286: Local-Variable Type Inference
- JEP 361: Switch Expressions
- JEP 394: Pattern Matching for instanceof
- JEP 395: Records
- JEP 409: Sealed Classes
- JEP 440: Record Patterns
- JEP 441: Pattern Matching for switch
- JEP 443: Unnamed Patterns and Variables

Pattern Matching

Условное и безопасное извлечение данных

```
if (input instanceof String) {
    String str = (String) input;
} else if (input instanceof Point) {
    Point point = (Point) input;
    var x = point.x();
    var y = point.y();
}
```

Pattern Matching

Условное и безопасное извлечение данных

```
if (input instanceof String str) {
} else if (input instanceof Point point) {
    var x = point.x();
    var y = point.y();
}
```

Pattern Matching

Условное и безопасное извлечение данных

```
if (input instanceof String str) {
} else if (input instanceof Point(var x, var y)) {
}
```

Полный шаблон

- Совпадает со всеми возможными входными значениями
- Полнота шаблона зависит от вида шаблона И от контекста
- Спросить себя, «требует ли шаблон каких-либо проверок»

Type pattern – полный или частичный

```
assert (Number) input __matches Number value == true;

assert (Number) null  __matches Number value == true;
```

Псевдокод!

Type pattern – полный или частичный

```
assert (Object) input __matches Number value == false;

assert (Object) null  __matches Number value == false;
```

Псевдокод!

Record pattern – частичный

```
assert (Point) input __matches Point(var x, var y) == true;

assert (Point) null  __matches Point(var x, var y) == false;
```

Псевдокод!

Unnamed pattern – полный

```
assert (Object) input __matches _ == true;

assert (Object) null  __matches _ == true;
```

Псевдокод!

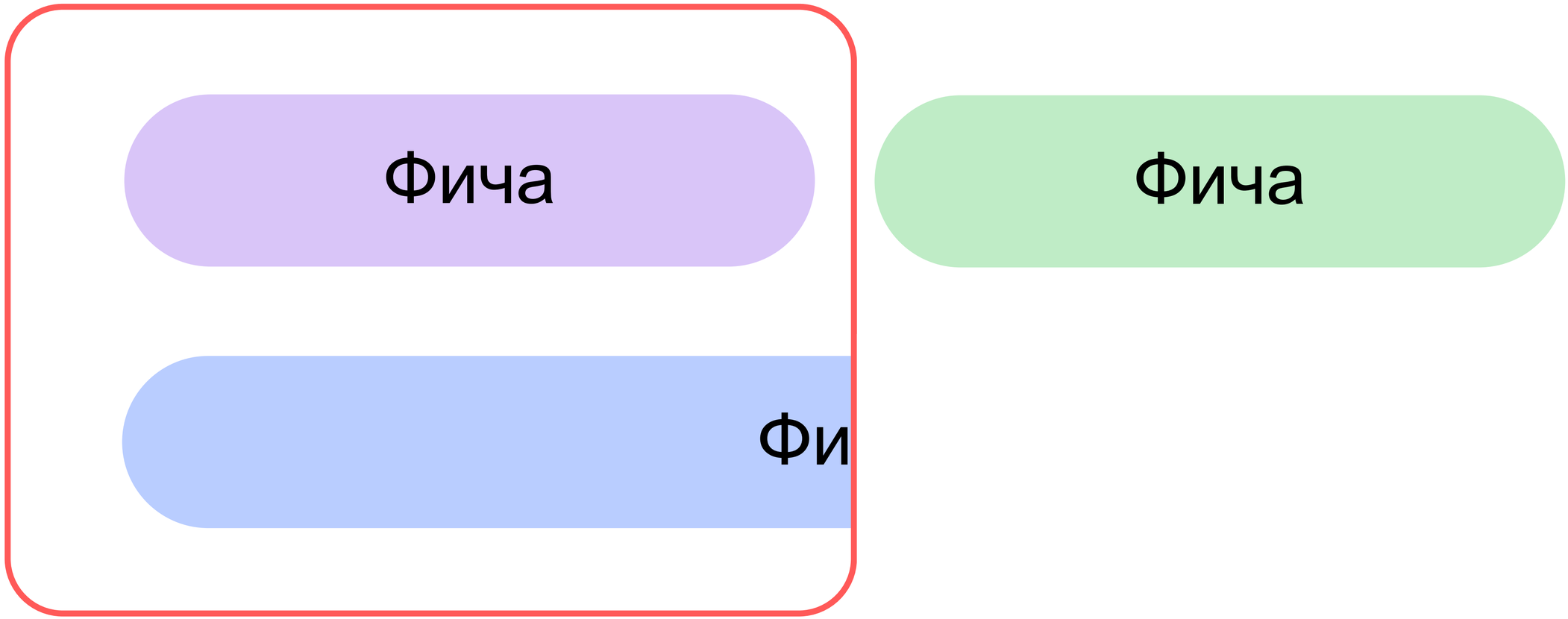
Принципы

- Ментальная модель
- Рефакторинг
- Композиция

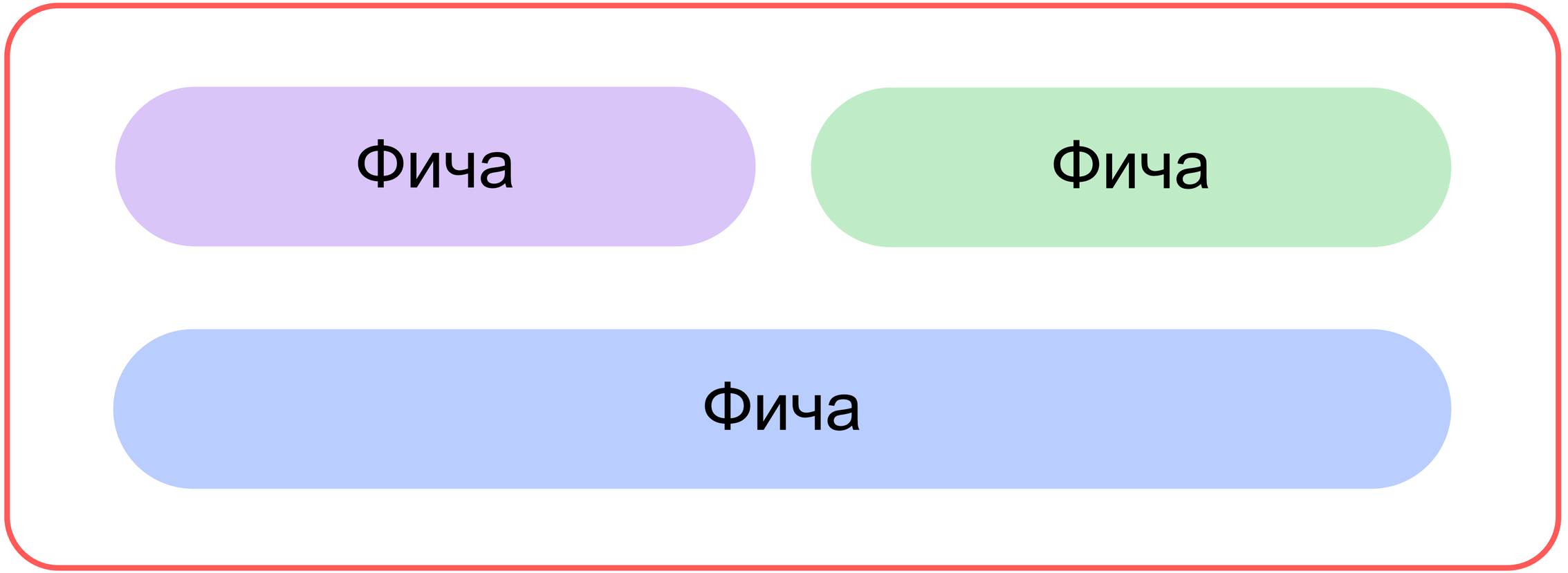
Ментальная модель

- «Внутренний» компилятор
- Который тоже надо апгрейдить

Thinking out of the box



Thinking out of the box



Ментальная модель и instanceof pattern

```
if (input instanceof String) {
    String str = (String) input;
}
```

```
if (input instanceof String str) {
}
```

Принципы

- Ментальная модель
- Рефакторинг
- Композиция

Рефакторинг

Изменение структуры кода без изменения его поведения

instanceof с типами

```
if (input instanceof String) {
    String str = (String) input;
} else if (input instanceof Point) {
    Point point = (Point) input;
    var x = point.x();
    var y = point.y();
} else {
}
```

instanceof с шаблонами

```
if (input instanceof String str) {
} else if (input instanceof Point(var x, var y)) {
} else {
}
```

switch с шаблонами

```
switch (input) {
    case String str           -> {}
    case Point(var x, var y) -> {}
    case null, default        -> {}
}
```

Принципы

- Ментальная модель
- Рефакторинг
- Композиция

The One True Source Of Expressiveness: Composition

Композиция instanceof с шаблонами

```
if (input instanceof Box(var value)
    && value instanceof Rectangle(var p1, var p2)
    && p1 instanceof Point(var x1, var y1)
    && p2 instanceof Point(var x2, var y2)
) {
}
```

instanceof с композицией шаблонов

```
if (input instanceof Box(Rectangle(Point(var x1, var y1), Point(var x2, var y2)))) {
}
```

Принципы

- Ментальная модель
- Рефакторинг
- Композиция

Принципы - применение

- Шаблоны с дополнительными условиями
- Обработка null

Принципы - применение

- Шаблоны с дополнительными условиями
- Обработка null

Шаблоны с дополнительными условиями

```
if (input instanceof Integer value && condition(value)) {
} else if (input instanceof Float value && condition(value)) {
} else if (input instanceof Number value) {
} else {
}
```

Шаблоны с дополнительными условиями

```
switch (input) {
    case Integer value when condition(value) -> {}
    case Float value when condition(value)   -> {}
    case Number value                        -> {}
    case null, default                       -> {}
}
```

Guarded pattern – частичный

```
assert (Number) input __matches Number value when condition(value) == false;

assert (Number) null  __matches Number value when condition(value) == false;
```

Псевдокод!

Guarded pattern – частичный

```
int obviousForProgrammer(Integer input) {
    return switch(input) {
        case null                          -> 0;
        case Integer value when value > 0  -> 1;
        case Integer value when value == 0 -> 0;
        case Integer value when value < 0  -> -1;
    }
}
```

Не компилируется!

Guarded pattern – частичный

```
int obviousForCompiler(Integer input) {
    return switch(input) {
        case null                              -> 0;
        case Integer value when value > 0  -> 1;
        case Integer value when value == 0 -> 0;
        case Integer value when value < 0  -> -1;
        case Integer value                     -> throw new AssertionError("never");
    }
}
```

Как реализовать guarded patterns?

- Через continue из тела бранча
- Через композицию шаблонов
- Через условие после ключевого слова

Условие в коде бранча

```
switch (input) {
    case Integer value -> {
        if (condition(value)) {
        } else {
            commonNumberCode();
        }
    }
    case Float value -> {
        if (!condition(value)) {
            commonNumberCode();
        } else {
        }
    }
    case Number value -> {
        commonNumberCode();
    }
    case null, default -> {}
}
```

Псевдокод!

continue в коде бранча

```
switch (input) {
    case Integer value -> {
        if (condition(value)) {
        } else {
            continue;
        }
    }
    case Float value -> {
        if (!condition(value)) {
            continue;
        }
    }
    case Number value -> {
        try {
            value.toString();
        } catch (Exception e) {
            continue;
        }
    }
    case null, default -> {}
}
```

Псевдокод!

Композиция шаблонов

```
if (input instanceof Integer value && condition(value)) {
}

switch (input) {
    case Integer value when condition(value) -> {}
}
```

Микроязык булевых выражений

```
booleanExpression1 && booleanExpression2
```

```
true && 42
true && "one"
```

Микроязык булевых выражений

```
(input instanceof pattern) && booleanExpression
```

Микроязык шаблонов

```
pattern1(pattern2)

pattern(42)
pattern("one")
```

Микроязык шаблонов

```
pattern1 && pattern2
```

~~pattern && true~~

Микроязык шаблонов

```
pattern1 && guard(booleanExpression)
```

Композиция шаблонов через оператор

```
if (input instanceof (Integer value && guard(condition(value))) {
} else if (input instanceof (Float value && guard(condition(value))) {
} else if (input instanceof Number value) {
} else {
}
```

Псевдокод!

Композиция шаблонов через оператор

```
switch (input) {
    case Integer value && guard(condition(value)) -> {}
    case Float value && guard(condition(value))   -> {}
    case Number value                             -> {}
    case null, default                            -> {}
}
```

Псевдокод!

Композиция шаблонов через оператор

```
switch (input) {
    case Integer value & guard(condition(value))       -> {}
    case Integer value && guard(condition(value))      -> {}
    case Integer value &&& guard(condition(value))     -> {}
    case Integer value and guard(condition(value))     -> {}

    case Integer value && true(condition(value))       -> {}
    case Integer value && false(condition(value))      -> {}

    case Box(Integer value && true(condition(value))) -> {}
}
```

Псевдокод!

Композиция шаблонов через оператор

```
switch (input) {
    case Integer value && condition(value) -> {}
}
```

Псевдокод?

if условие

```
switch (input) {
    case Integer value if condition(value) -> {}
    case Float value if condition(value)   -> {}
    case Number value                      -> {}
    case null, default                     -> {}
}
```

Псевдокод!

when условие

```
switch (input) {
    case Integer value when condition(value) -> {}
    case Float value when condition(value)   -> {}
    case Number value                        -> {}
    case null, default                       -> {}
}
```

Почему никто не любит новые ключевые слова?

```
record when(boolean when) {}
```

Почему никто не любит новые ключевые слова?

```
record when(boolean when) {}

boolean when(when when) {
    return when.when();
}
```

Почему никто не любит новые ключевые слова?

```
record when(boolean when) {}

boolean when(when when) {
    return when.when();
}

int pleaseDontDoThis(when input) {
    return switch (input) {
        case when(boolean when) when when -> 1;
        case when when when when(when)    -> 2;
    }
}
```

Не компилируется!

Почему никто не любит новые ключевые слова?

```
record when(boolean when) {}

boolean when(when when) {
    return when.when();
}

int pleaseDontDoThis(when input) {
    return switch (input) {
        case when(boolean when) when when -> 1;
        case when when when when(when)    -> 2;
        case when when                    -> 3;
    }
}
```

Шаблоны с дополнительными условиями

- Ментальная модель пока не готова к композиции шаблонов
- Без рефакторинга между instanceof и switch было бы тяжело
- when нельзя, но очень хочется, то можно

Принципы - применение

- Шаблоны с дополнительными условиями
- Обработка null

Обработка null

```
if ((Integer) null instanceof Integer) {
}

switch ((Integer) null) {
    case 0 -> {}
}
```

Обработка null

```
if ((Integer) null instanceof Integer i) {
}

switch ((Integer) null) {
    case Integer i -> {}
}
```

instanceof vs null vs полный шаблон

```
assert  (Number) null instanceof Integer == false;
assert (Integer) null instanceof Integer == false;

assert  (Number) null instanceof Integer value == false;
assert (Integer) null instanceof Integer value == true; // value == null
```

Java 16 instanceof vs null vs полный шаблон

```
assert  (Number) null instanceof Integer == false;
assert (Integer) null instanceof Integer == false;

assert  (Number) null instanceof Integer value == false;
assert (Integer) null instanceof Integer value == true; // value == null
```

switch vs null vs полный шаблон

```
switch ((Integer) null) {
    case Integer i -> {}
}
```

Java 1 – primitive switch vs null

```
void test(int input) {
    switch (input) {
        case 0 -> {}
    }
}
```

Java 5 – wrapper switch vs null

```
void test(Integer input) {
    switch (input) {
        case 0 -> {}
    }
}
```

```
void test(Integer input) {
    switch (input.intValue()) {
        case 0 -> {}
    }
}
```

Псевдокод!

Java 5 – enum switch vs null

```
void test(Color input) {
    switch (input) {
        case RED -> {}
    }
}
```

```
void test(Color input) {
    switch (input.ordinal()) {
        case 0 -> {}
    }
}
```

Псевдокод!

Java 7 – String switch vs null

```
void test(String input) {
    switch (input) {
        case "" -> {}
    }
}
```

```
void test(String input) {
    switch (input.hashCode()) {
        case 0 -> {
            if ("".equals(input)) {
            }
        }
    }
}
```

Псевдокод!

switch vs null

switch кидает NPE из-за случайного решения, сделанного 20 лет назад

Java 21 – switch vs null vs полный шаблон

```
switch ((Box) null) {
    case Box box -> {}
}
```

Java 21 – switch vs null vs вложенный шаблон

```
switch (new Box(null)) {
    case Box(Integer i) -> {}
    case Box(Number n)  -> {}
    case Box(Object o)  -> {}
}
```

```
switch (new Box(null)) {
    case Box(var obj) -> switch (obj) {
        case Integer i -> {}
        case Number n  -> {}
        case Object o  -> {}
    }
}
```

Что делать с null в switch?

- Отправлять в полный шаблон
- Отправлять в константный null шаблон
- Отправлять в default
- Отправлять в case null
- Кидать NPE

Java 21 – switch vs null – полный шаблон

```
Number input = null;
switch (input) {
    case Integer value -> {}
    case Number value  -> {}
}
```

```
Number input = null;
if (input instanceof Integer value) {
} else if (input instanceof Number value) {
}
```

Java 21 – switch vs null – ~~полный шаблон~~

```
Rectangle input = new Rectangle(…, …);
switch (input) {
    case Rectangle(Point p1, Point p2) -> {}
    case Rectangle(Point p1, null)     -> {}
    case Rectangle(null, Point p2)     -> {}
    case Rectangle(null, null)         -> {}
}
```

Java 21 – switch vs null – default

```
Number input = null;
switch (input) {
    case Number value -> {}
    default           -> {}
}
```

```
Number input = null;
if (input instanceof Number value) {
} else {
}
```

Java 21 – switch vs null – case null

```
Number input = null;
switch (input) {
    case null          -> {}
    case Number value -> {}
}
```

```
Number input = null;
if (input == null) {
} else if (input instanceof Number value) {
}
```

Java 21 – switch vs null – case null

```
Number input = null;
switch (input) {
    case Number value  -> {}
    case null, default -> {}
}
```

```
Number input = null;
if (input instanceof Number value) {
} else {
}
```

Java 21 – switch vs null – NPE

Let the nulls flow (until they really can’t)

- Не надо накладывать искусственные ограничения на домен
- Если в домене нет null, то и хорошо
- Если в домене есть null, то код должен это учитывать

Композиция vs null

- Конструкции языка обладают собственным поведением
- Шаблоны обладают собственным поведением

instanceof vs null

- Если исходное значение null, то вернуть false
- Иначе проверить, совпадает ли с «аргументом»

Псевдокод!

Java 21 instanceof vs null vs полный шаблон

```
assert  (Number) null instanceof Integer == false;
assert (Integer) null instanceof Integer == false;

assert  (Number) null instanceof Integer value == false;
assert (Integer) null instanceof Integer value == false;
```

switch vs null

- Если исходное значение null, и есть case null, то выполнить его, иначе кинуть NPE
- Иначе, если есть совпадающий case, то выполнить его
- Иначе, если есть default, то выполнить его
- Иначе выйти из switch-оператора или кинуть ME из switch-выражения

Псевдокод!

Java 21 switch vs null

```
Color input = …;
switch (input) {
    case null -> {}
    case RED  -> {}
}
```

```
String input = …;
switch (input) {
    case null -> {}
    case ""   -> {}
}
```

pattern vs null

- Полный шаблон совпадает с null
- Шаблон в контексте конструкции языка может не видеть null на входе, если таково поведение этой конструкции
- Вложенный шаблон работает в контексте шаблонов, и может видеть null на входе

Псевдокод!

Java 21 полный шаблон vs null

```
assert (Box) null    instanceof Box box                    == false;

assert new Box(null) instanceof Box box                    == true;

assert new Box(null) instanceof Box(Object value)          == true;

assert new Box(null) instanceof Box(Point(var x, var y)) == false;
```

Работать с null – сплошное удовольствие!

- Убрали случайные ограничения, реабилитировали instanceof для полных шаблонов и switch для «старых» типов
- Предоставили механизм для обработки null на любом уровне вложенности
- Реализовали универсальные фичи, а не одноразовые решения для конкретных комбинаций

Принципы - применение

- Шаблоны с дополнительными условиями
- Обработка null

Java 21+

Что, может быть, будет дальше

- Старые и новые конструкции с поддержкой шаблонов
- Новые шаблоны
- Новые конструкции*

Старые конструкции с поддержкой шаблонов

```
for (Point(var x, var y) : points) {
}

try {
} catch (Exception(Exception cause)) {
}
```

Псевдокод!

Новые конструкции с поддержкой шаблонов

```
let Point p = point;

let Point(var x, var y) = point;

let Point(var x, var y) = point
    when x > 0
    else {
        x = 0;
        y = 0;
    };
```

Псевдокод!

Новые шаблоны

```
class Parent {
    deconstructor Parent(int x) {
        x = this.x;
    }
}
```

Псевдокод!

Новые шаблоны

```
class Parent {
    deconstructor Parent(int x) {
        x = this.x;
    }
}
```

```
class Child extends Parent {
    deconstructor Child(int x, int y) {
        let Parent(int xx) = this;
        x = xx;
        y = this.y;
    }
}
```

Псевдокод!

Новые шаблоны

```
int input = 0;
if (input instanceof byte value) {
} else if (input instanceof int value) {
}
```

```
Point input = new Point(0, 0);
switch (input) {
    case Point(byte x, byte y) -> {}
    case Point(int x, int y)   -> {}
}
```

Псевдокод!

Новые конструкции*

```
if (point instanceof Point(int x, int y)) {
    x = 2 * x;
    y = 2 * y;
    point = new Point(x, y);
}
```

Новые конструкции*

```
point = switch (point) {
    case Point(int x, int y) -> new Point(
        2 * x,
        2 * y
    );
};
```

Новые конструкции*

```
point = point with {
    x = 2 * x;
    y = 2 * y;
};
```

Псевдокод!

Когда мы начнём всем этим пользоваться

- Переходим на record и sealed классы
- Ждём объявляемые шаблоны
- Ждём поддержку шаблонов в proto/xml/json библиотеках

О чём был этот доклад

- О ментальной модели, которую необходимо постоянно развивать
- О рефакторинге, который является одновременно и стимулом и ограничителем для дизайна языка
- О композиции, которая позволяет нам создавать сложное поведение из простых блоков

О чём я не успел рассказать

- О доминантности case блоков в switch с шаблонами
- О полноте/exhaustiveness switch-выражения
- О попытках поддержать полноту в switch-операторе
- Об остатке/remainder, MatchException, и изменениях, нарушающих обратную совместимость
- О взаимодействии с generic типами
- О поддержке остальных примитивных типов в switch
- Об объявляемых шаблонах/declared patterns
- О шаблонах и литералах для коллекций
- О видимости шаблонных переменных

Спасибо!

Олег Естехин
Yandex Cloud
@OlegEstekhin