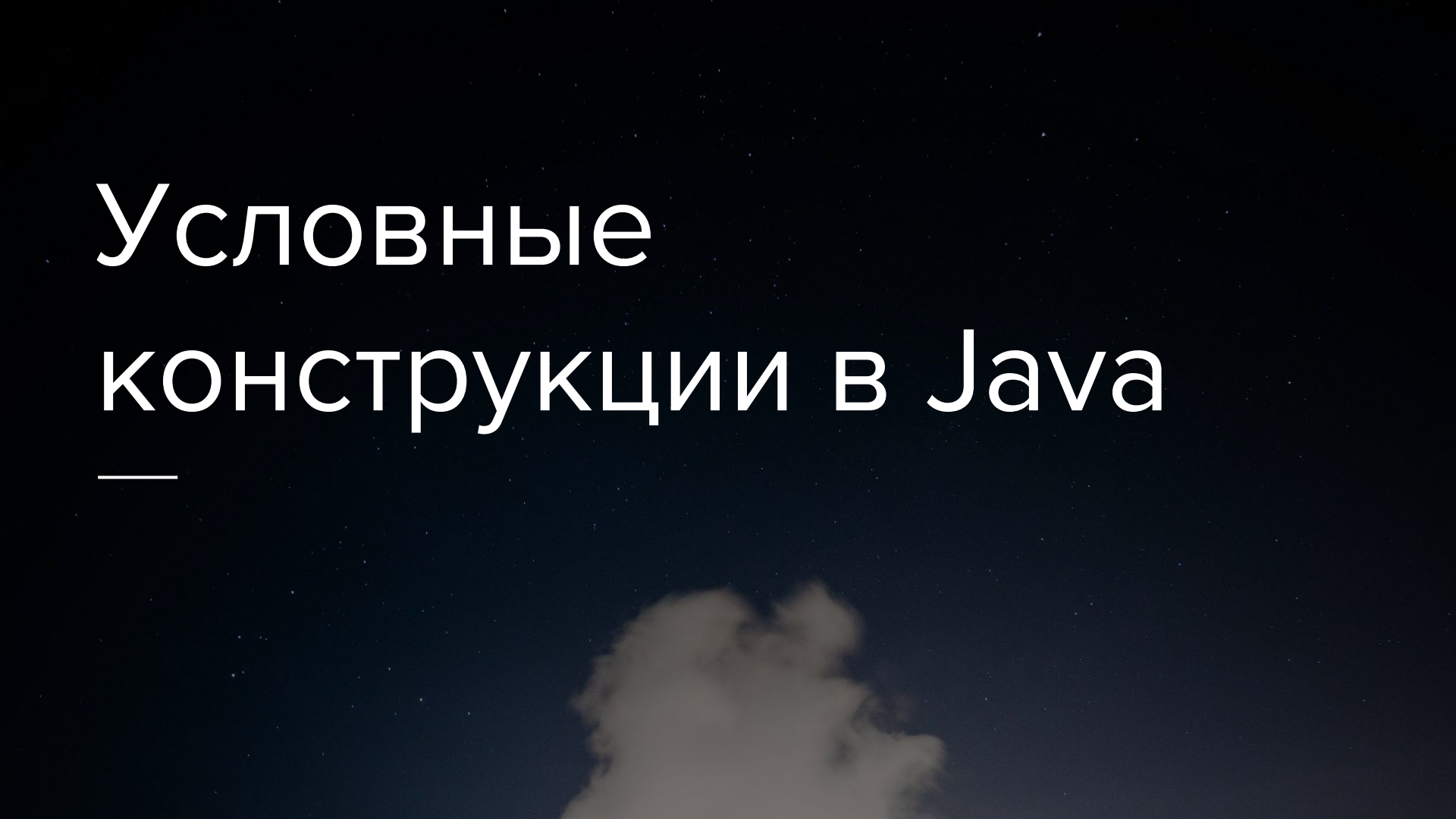
Условные конструкции в Java



Условные операторы используются, когда в зависимости от условия необходимо выполнить разные действия.

Оператор if

если указанное **условие** истинно (true), выполняет **код** в фигурных скобках,

Синтаксис:

```
if (условие) {
    код, который выполнится, если условие истинно
}
```

Например,

```
if (answer == 13) { // если значение  переменной answer будет равно 13
    // в консоль будет выведено: Ответ верный
    System.out.println("Ответ верный");
}
```

Необязательный блок else

выполняется, если **условие** ложно (false)

Синтаксис:

```
if (условие) {
    код, который выполнится, если условие истинно
} else {
    код, который выполнится, если условие ложно
}
```

Например,

```
if (answer == 13) { // если значение  переменной answer будет равно 13
    // в консоль будет выведено: Ответ правильный
    System.out.println("Ответ правильный");
} else { // если значение  переменной answer не будет равно 13
    // в консоль будет выведено: Ошибка в ответе
    System.out.println("Ошибка в ответе");
}
```

Несколько условий else if

используются, если необходимо добавить новые варианты условий.

Каждое новое условие будет проверяться только,
если предыдущие условия ложны

Синтаксис:

```
if (условие 1) {
    код, который выполнится, если условие истинно
} else if (условие 2){
    код, который выполнится, если условие 1 ложно и условие 2 истинно
} else {
    код, который выполнится, если условие 1 ложно и условие 2 ложно
}
```

Несколько условий else if (пример)

Например,

```
if (answer == 13) { // если значение  переменной answer будет равно 13
        // в консоль будет выведено: Ответ правильный
        System.out.println(“Ответ правильный”);
} else if (answer < 13) { // если значение  переменной answer будет меньше 13
        // в консоль будет выведено: Попробуйте число больше
        System.out.println("Попробуйте число больше");
} else {  //если все условия будут ложными
        // в консоль будет выведено: Попробуйте число меньше
        System.out.println("Попробуйте число меньше");
}
```

Конструкция switch

может выполнять проверку только равенства
(оператор if может вычислять результат булева выражения).

Две **константы** case в операторе switch не могут иметь одинаковые значения

Синтаксис,

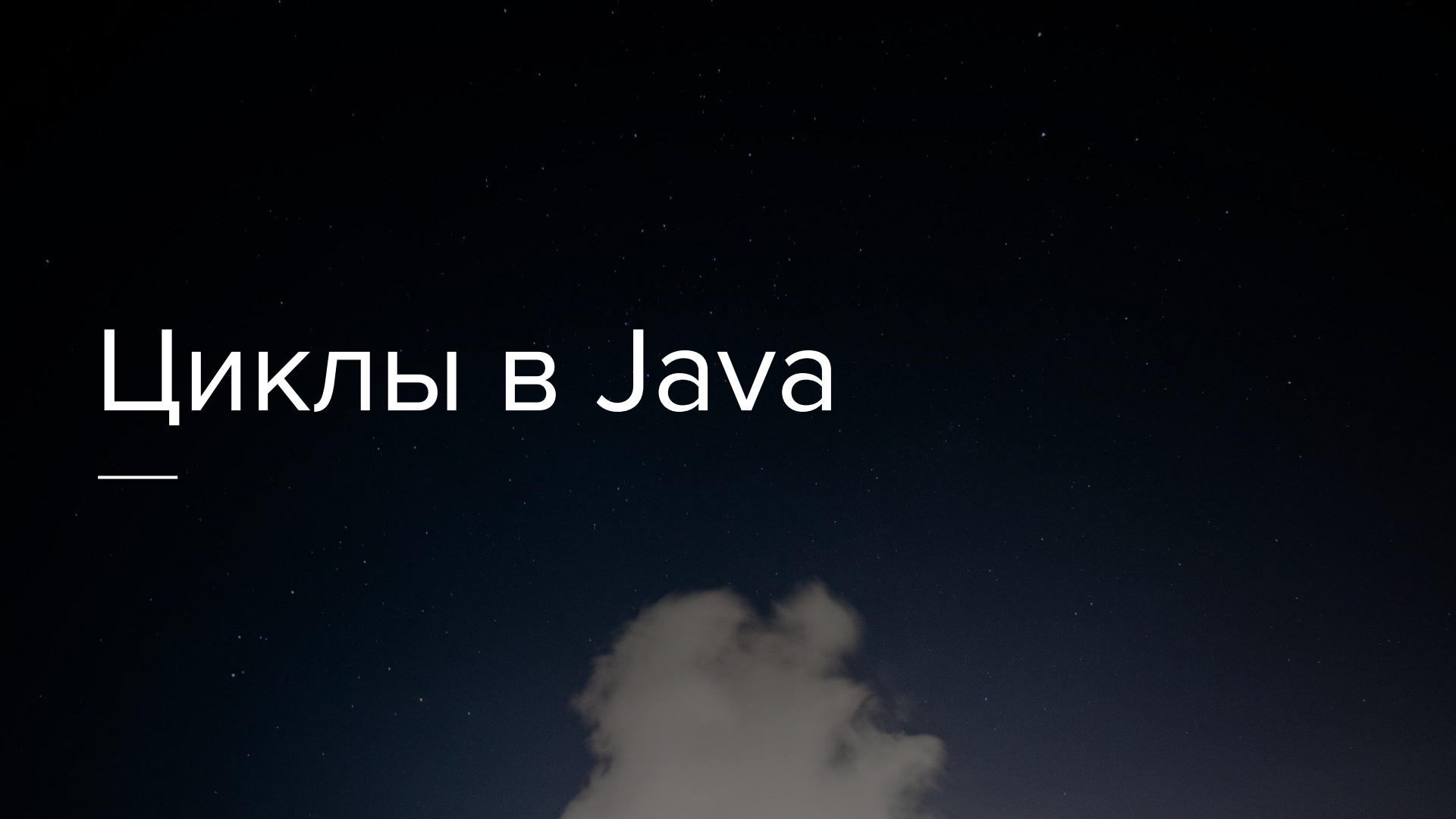
```
switch(выражение) {
        case значение1 : // if (выражение == значение1)
                ... код
                [break]
        case значение2 : // if (выражение == значение1)
                ... код
                [break]
        case значение3 : // if (выражение == значение3)
        case значение4 : // if (выражение == значение4)
                 ... код
                [break]
        ...
        default :
                ... код, который необходимо выполнить,
                        если ни один case не совпал
                [break]
}
```

Выражение проверяется на равенство 1му значению **значение1**, затем 2му **значение2** и так далее.
Если **соответствие установлено** – switch **начинает выполняться** от соответствующего case и далее, **до ближайшего break или до конца switch.**
Если **ни один case не совпал** – выполняется вариант **default** (если он описан).

Конструкция switch (пример)

```
String item = “какое-то значение”;
switch (item) {
  case "Oranges": // если item == "Oranges",
    // отработает данный case и в консоле мы увидим "Oranges - $0.59 a pound."
    System.out.println("Oranges - $0.59 a pound.");
    // так как break не указан следующий case отработает без проверки условия
    // в консоле мы увидим также "Apples - $0.32 a pound."
  case "Apples": // если item == "Apples",
    // отработает данный case и в консоле мы увидим "Apples - $0.32 a pound."
    System.out.println("Apples - $0.32 a pound.");
    break;
  case "Mangoes": // если item == "Mangoes" или / и
  case "Papayas": // если item == "Papayas"
    // в консоле мы увидим "Mangoes and papayas are $2.79 a pound."
    System.out.println("Mangoes and papayas are $2.79 a pound.");
    break;
  default: // если item не найдет совпадений,
    // в консоле мы увидим "Sorry, we are out of " + item + "."
    System.out.println("Sorry, we are out of " + item + ".");
}
```

Циклы в Java



Циклы позволяют выполнять однотипное действие несколько раз. Каждое повторение цикла называется **итерацией**.

Цикл с предусловием while

Условие будет проверяться перед каждым выполнением тела цикла, если оно истинно, тело цикла будет выполняться. Так будет происходить, пока условие истинно, когда условие станет ложным, программы выйдет из цикла.
Если условие изначально ложно, тела цикла не будет выполнено ни разу.
Если условие всегда истинно, цикл будет продолжаться бесконечно.

Синтаксис,

```
while(условие) { // проверка условия
        тело цикла  выполняется, если условие истинно
} // если условие ложно, программа выходит из цикла и продолжает работу
код после цикла;
```

Цикл с предусловием while (пример)

```
int count = 3;

while (count > 0) {

        System.out.println(count);
        count--;

}
```

Первая проверка условия: ***count*** = 3, значит условие истинно и тело цикла выполнится.

Итерация 1 (первое выполнение тела цикла):

1. в консоль будет выведено: 3
2. уменьшение значения ***count*** на 1, значит ***count*** будет равен 2

Вторая проверка условия: ***count*** = 2 (после уменьшения в теле цикла), значит условие истинно и тело цикла выполнится

Итерация 2 (второе выполнение тела цикла):

1. в консоль будет выведено: 2
2. уменьшение значения ***count*** на 1, значит ***count*** будет равен 1

Третья проверка условия: ***count*** = 1(после уменьшения в теле цикла), значит условие истинно и тело цикла выполнится

Итерация 3 (второе выполнение тела цикла):

1. в консоль будет выведено: 1
2. уменьшение значения ***count*** на 1, значит ***count*** будет равен 0

Четвертая проверка условия: ***count*** = 0 (после уменьшения в теле цикла), значит **условие ложно** и **тело цикла не будет выполнено**, программа выйдет из цикла.

Цикл с постусловием do...while

Сначала будет выполняться тело цикла, а затем проверяться условие, если оно истинно, тело цикла будет выполняться еще раз. Так будет происходить, пока условие истинно, когда условие станет ложным, программы выйдет из цикла.

Если условие изначально ложно, тела цикла выполняется один раз, т.к. условие проверяется после выполнения тела цикла.

Если условие всегда истинно, цикл будет продолжаться бесконечно.

Синтаксис,

```
do {
    тело цикла  выполнится первый раз в любом случае,
                далее будет выполняться, если условие истинно
} while (условие); // проверка условия, если условие ложно, программа выходит из
                   цикла и продолжает работу
код после цикла;
```

Цикл с постусловием do...while (пример)

```
int count = 2;
do { // тело цикла:
        System.out.println(count);  // вывод в консоль:  значение count
        count ; // уменьшение значения count на 1
} while (count > 0);  // проверка условия: условие истинно, пока значение count больше 0
  // условие станет ложным, когда значение count будет равно 0
```

Итерация 1 (первое повторение тела цикла):

1. в консоль будет выведено: 2
2. уменьшение значения ***count*** на 1, значит ***count*** будет равен 1

Первая проверка условия: ***count*** = 1, значит условие истинно и тело цикла выполнится.

Итерация 2 (второе повторение тела цикла):

1. в консоль будет выведено: 1
2. уменьшение значения ***count*** на 1, значит ***count*** будет равен 0

Вторая проверка условия: ***count*** = 0, значит условие ложно и тело цикла выполняться не будет, программа выйдет из цикла

Цикл for

состоит из выполнения трех операций

Синтаксис,

```
for ( [ начало-инициализация ]; [ условие ]; [ шаг ] ) {
    тело цикла  выполняется, если условие истинно
}
```

[ ] - квадратные скобки в описании используются для обозначения необязательных параметров

1. **начало-инициализация** - выполняется один раз, при заходе в цикл. Обычно это выражение инициализирует один или несколько счётчиков. Также используется для объявления переменных.
2. **условие** - проверяется при входе в цикл и далее перед каждой итерацией, если оно истинно, **тело цикла** выполняется, если ложно, программа выходит из цикла. Если **условие** пропущено, оно считается истинным.
3. **шаг** - выполняется после выполнения тела цикла на каждой итерации, но перед проверкой условия. Обычно, обновление счетчика (уменьшение/увеличение).

Цикл for (пример)

```
int count = 3;

for (int i = 0; i < count ; i++) {
        System.out.println(count);
        count--;
}
```

Начало цикла:
int i = 0 - инициализация счетчика

Первая проверка условия: i < ***count***,
значение i равно 0, а значение ***count*** равно 3,
получаем 0 < 3, значит условие истинно и тело цикла выполнится

Итерация 1 (первое выполнение тела цикла):

1. в консоль будет выведено: 3
2. уменьшение значения ***count*** на 1, значит ***count*** будет равен 2

Первое обновление счетчика (шаг):
i++ - значение i станет равно 1

Вторая проверка условия: i < ***count***,
значение i равно 1(после первого обновления счетчика), а ***count*** равно 2 (после уменьшения в теле цикла),
получаем 1 < 2,
значит условие истинно и тело цикла выполнится.

Итерация 2 (второе выполнение тела цикла):

1. в консоль будет выведено: 2
2. уменьшение значения ***count*** на 1, значит ***count*** будет равен 1

Второе обновление счетчика (шаг):
i++; значение i станет равно 2

Третья проверка условия: i < ***count***,
значение i равно 2(после второго обновления счетчика), а значение ***count*** равно 1 (после уменьшения в теле цикла),
получаем 2 < 1, значит условие ложно и тело не выполнится, программа выйдет из цикла

Директивы break и continue

Прерывание цикла break

позволяет выйти из цикла в любой момент и продолжить выполнение кода после цикла.

Синтаксис,

```
while(условие) { // проверка условия

    тело цикла  выполняется, если условие истинно

    if (условие2) { break; }  // если условие2 истинно, программа выходит
                                 из цикла благодаря директиве break
}
код после цикла;
```

Директивы break и continue

Следующая итерация continue

прекращает выполнение текущей итерации цикла.

Синтаксис и пример,

```
for (int i = 0; i < 10; i++) {

    if (i % 2 === 0) continue;   // если i - четное число,
                                 завершает текущую итерацию,
                                 цикл переходит к проверки  условия i < 10
    System.out.println(i);   // выводит значение i
}
```

Директивы break и continue (метки)

метки позволяют указать из какого цикла необходимо выйти (в случае break) или какой включающий цикл следует продолжить (в случае continue)

Например,

```
outer: for (инициализация; условия; обновление счетчиков){
    // цикл outer
        for (инициализация; условия; обновление счетчиков){
            // вложенный цикл
            if (условие) break outer; // выход не только из вложенного цикла,
                                      но из outer
        }
    }
```