



# Принципы Java

8-4

Основные принципы и методы отладки



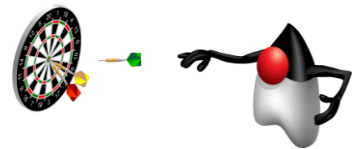
**ORACLE**  
Academy

© 2019, Корпорация Oracle и аффилированные с ней лица. Все права защищены.

# Цели

В данном уроке рассматриваются следующие темы:

- Тестирование и отладка программы Java
- Определение трех типов ошибок
- Применение методов отладки
  - Операторы `print`
  - Отладчик NetBeans
- Советы и методы отладки



# Темы

- Тестирование и отладка программы Java
- Три типа ошибок
- Методы отладки Операторы `print`
- Методы отладки Отладчик NetBeans
- Советы по отладке



**ORACLE**  
Academy

JFo 8-4  
Основные принципы и  
методы отладки

© 2019, Корпорация Oracle и аффилированные с ней лица. Все права защищены.

4

# Тестирование программы Java

Ричи написал программу Java, чтобы найти максимальное значение среди трех целых чисел:

```
public static void main(String[] args) {  
    int num1 = 3, num2 = 3, num3 = 3;  
    int max = 0;  
    if (num1 > num2 && num1 > num3) {  
        max = num1;  
    }  
    if (num2 > num1 && num2 > num3) {  
        max = num2;  
    }  
    if (num3 > num1 && num3 > num2) {  
        max = num3;  
    }  
    System.out.println(" Максимум трех чисел составляет " + max);  
}
```

# Тестирование программы Java

- Ричи **проверил** программу на множестве наборов данных, например  $\langle 3, 5, 9 \rangle$ ,  $\langle 12, 1, 6 \rangle$  и  $\langle 2, 7, 4 \rangle$ .
- Программа работала для всех данных.
- Тем не менее, ему сказали, что программа не работает; он не мог понять почему.



# Упражнение 1

- Импортируйте и откройте проект `DebuggingEx`.
- Просмотрите `MaxIntegers.java`.
- Можете ли вы определить, что Ричи пропустил во время своей проверки?

## Определение ошибки

- Программа выдает сбой при проверке с помощью **дублирующихся значений**, например  $\langle 3,3,3 \rangle$  и  $\langle 7,2,7 \rangle$ , и отображает нулевой результат.
- Вы определили ошибку.
- Следующим этапом является исправление ошибки.



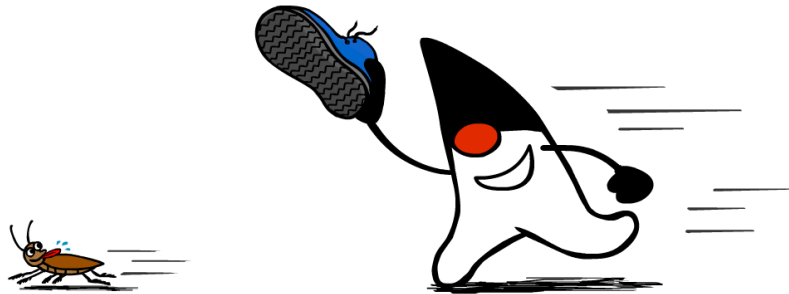
# Исправление ошибки

Измените программу и проверьте ее на множестве наборов данных, включая дублирующиеся значения.

```
public static void main(String[] args) {  
    int num1 = 3, num2 = 3, num3 = 3;  
    int max = 0;  
    if (num1 > max) {  
        max = num1;  
    }  
    if (num2 > max) {  
        max = num2;  
    }  
    if (num3 > max) {  
        max = num3;  
    }  
    System.out.println(" Максимум трех чисел составляет " + max);  
}
```

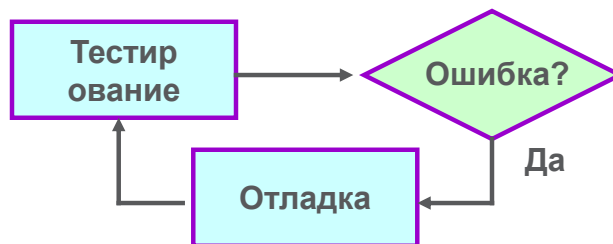
# Тестирование и отладка

Как видно из предыдущего примера, проверка и отладка являются важными действиями в процессе разработки ПО.



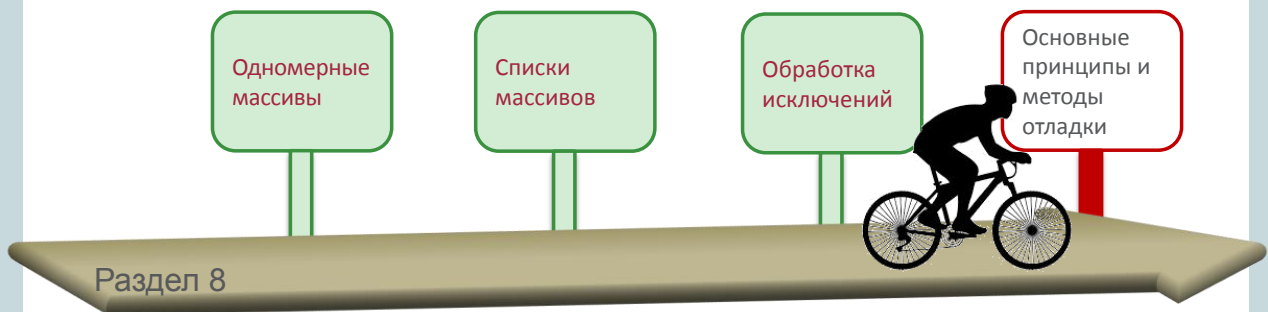
# Тестирование и отладка

- Проверка:
  - Для определения наличия ошибок в коде
- Отладка:
  - Для определения ошибки и ее исправления



# Темы

- Тестирование и отладка программы Java
- Три типа ошибок
- Методы отладки Операторы `print`
- Методы отладки Отладчик NetBeans
- Советы по отладке



**ORACLE**  
Academy

JFo 8-4  
Основные принципы и  
методы отладки

© 2019, Корпорация Oracle и аффилированные с ней лица. Все права защищены.

12

# Три типа ошибок

- Ошибки компиляции
- Логические ошибки
- Ошибки выполнения

# Ошибки компиляции

- Ошибка синтаксиса
- Простой тип ошибок для исправления
- Примеры:

– Пример 1: Отсутствие точки с запятой

```
int a = 5 // отсутствует точка с запятой
```

– Пример 2: Ошибки в выражении

```
x = ( 3 + 5; // отсутствует закрывающая скобка
```

```
y = 3 + * 5; // отсутствует аргумент между "+" и "*"
```

**ORACLE**

Academy

JFo 8-4  
Основные принципы и  
методы отладки

© 2019, Корпорация Oracle и аффилированные с ней лица. Все права защищены.

14

# Логические ошибки

- Программа выполняется, но выдает неверный результат.
- Трудно охарактеризовать, следовательно труднее исправить.
- Пример: Неинициализированная переменная

```
int i;
```

```
i++; // переменная i не инициализирована
```

**ORACLE**

Academy

JFo 8-4  
Основные принципы и  
методы отладки

© 2019, Корпорация Oracle и аффилированные с ней лица. Все права защищены.

15

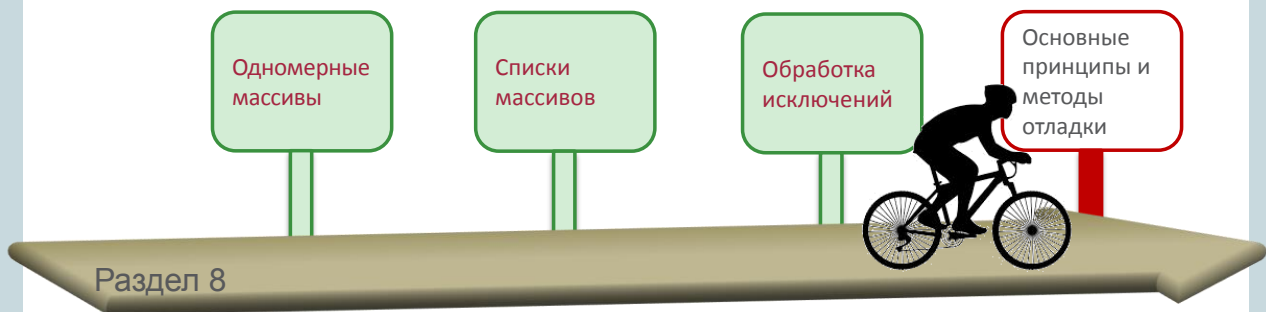
# Ошибки выполнения

- Эти ошибки происходят на этапе выполнения.
- Механизм обработки исключений Java не может захватить такие ошибки.
- Некоторые из основных исключений:
  - `ArrayIndexOutOfBoundsException`
  - `NullPointerException`
  - `ArithmeticException`



# Темы

- Тестирование и отладка программы Java
- Три типа ошибок
- Методы отладки Операторы `print`
- Методы отладки Отладчик NetBeans
- Советы по отладке



**ORACLE**  
Academy

JFo 8-4  
Основные принципы и  
методы отладки

© 2019, Корпорация Oracle и аффилированные с ней лица. Все права защищены.

17

# Методы отладки

Рассмотрим два метода отладки:

- С помощью операторов `print`
- С помощью отладчика NetBeans

# Операторы `print`: Преимущества

- Удобное добавление
- Предоставление информации
  - Какие методы были вызваны
  - Значение параметров
  - Порядок, в котором методы были вызваны
  - Значения локальных переменных и полей в стратегических точках

## Операторы `print`: Недостатки

- Не практично добавлять операторы `print` к каждому методу.
- Слишком много операторов `print` ведет к перегрузке информацией.
- Удаление операторов `print` является трудоемкой работой.

# Операторы `print`: Пример

- Рассмотрим следующий код Java:

```
int n = 10;
int sum = 10;
while (n > 1){
    sum = sum + n;
    n--;
}
System.out.println("Сумма целых чисел от 1 до 10 равна " + sum);
```

- При выполнении данной программы он работает неверно.
- Чтобы найти причину, можно отследить значение переменных `n` и `sum` с помощью вставки операторов `print`.

**ORACLE**

Academy

JFo 8-4  
Основные принципы и  
методы отладки

© 2019, Корпорация Oracle и аффилированные с ней лица. Все права защищены.

21

# Измененная программа с дополнительными операторами `print` для отладки

```
int n = 10;
int sum = 10;
while (n > 1) {
    System.out.println("В начале цикла: n = " + n);
    System.out.println("В начале цикла: sum="+ sum);

    System.out.println("В конце цикла: n = " + n);
    System.out.println("В конце цикла: sum = " + sum);
    System.out.println("Сумма целых чисел от 1 до 10 равна " + sum);
}
```

**ORACLE®**

Academy

JFo 8-4  
Основные принципы и  
методы отладки

© 2019, Корпорация Oracle и аффилированные с ней лица. Все права защищены.

22

## Вывод

- Далее представлены четыре первые строки результата после **первой итерации** цикла:

В начале цикла: `n = 10`

В начале цикла: `sum = 10`

В конце цикла: `n = 9`

В конце цикла: `sum = 20`

- Можно увидеть, что что-то не так.

Для переменной `sum` было установлено значение 20. Так как она была инициализирована для значения 10, для нее установлено значение  $10 + 10$ , которое не является верным при необходимости добавления чисел от 1 до 10.

**ORACLE**

Academy

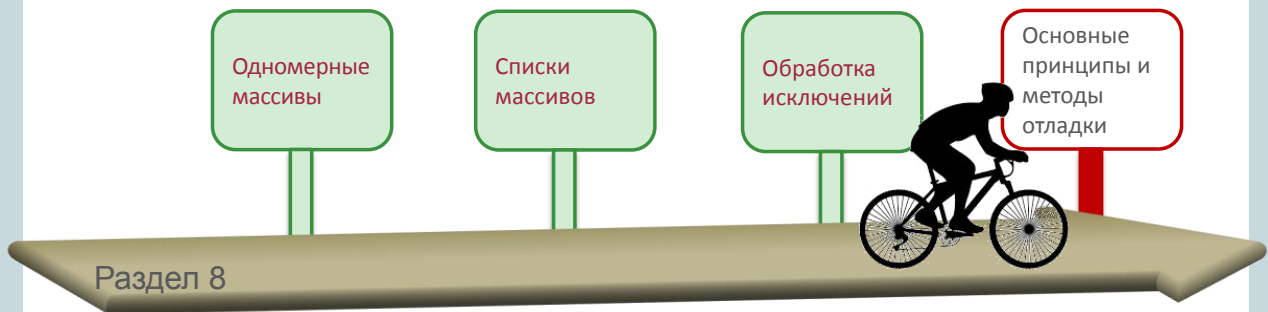
JFo 8-4  
Основные принципы и  
методы отладки

© 2019, Корпорация Oracle и аффилированные с ней лица. Все права защищены.

23

# Темы

- Тестирование и отладка программы Java
- Три типа ошибок
- Методы отладки Операторы `print`
- Методы отладки Отладчик NetBeans
- Советы по отладке



**ORACLE**  
Academy

JFo 8-4  
Основные принципы и  
методы отладки

© 2019, Корпорация Oracle и аффилированные с ней лица. Все права защищены.

24



# Отладчик NetBeans

- Вы уже использовали среду отладки NetBeans на основе графического интерфейса.
- Были использованы следующие функции отладчика:
  - Установка точек останова
  - Трассировка через программу по каждой строке за раз
- Воспользуемся другой важной функцией для просмотра содержимого переменных.

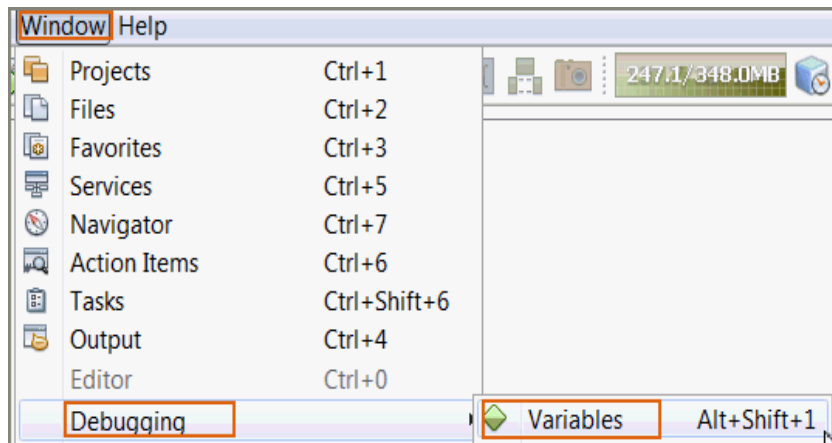
# Окно переменных

- При достижении установленной точки останова можно воспользоваться окном переменных, чтобы увидеть значение переменных в данный момент.
- Можно найти значения переменных без ввода множества операторов `print` в программу.

# Доступ к окну переменных


Чтобы отобразить окно переменных, в главном меню NetBeans выполните следующее:

Нажмите "Window > Debugging > Variables".




## Упражнение 2: сценарий



- Предположим, что у вас есть машина и необходимо поехать на заправочную станцию. У вас есть следующие (  )
  - Текущее расположение машины:  $x1$  и  $y1$
  - Расположение заправочной станции:  $x2$  и  $y2$
  - Скорость машины
- Необходимо вычислить время, которое потребуется машине, чтобы добраться от текущего местоположения ( $x1, y1$ ) до заправочной станции ( $x2, y2$ ).
- Программа Java для вычисления времени с помощью формулы  $\text{time} = \text{distance} / \text{speed}$  доступна в проекте `ComputeTime`.

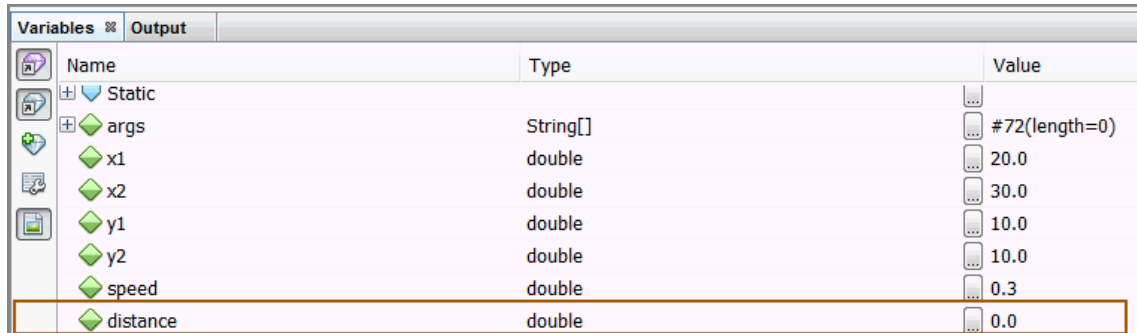


## Упражнение 2

- Импортируйте и откройте проект `DebuggingEx`.
- Просмотрите `ComputeTime.java`.
- Запустите программу с отладчиком NetBeans для отладки этой программы:
  - Установите точку останова в методе `getDistance`.
  - Нажмите **Ввод** для перехода к следующей строке .
  - Просмотрите значения переменных `x1`, `x2`, `y1`, `y2`, `speed`, `distance` и `time`.
- Можно ли выявить ошибку?

# Просмотрите значение `distance`

- В предыдущем упражнении с помощью функций отладки NetBeans была выявлена ошибка.



Name	Type	Value
Static		
args	String[]	#72(length=0)
x1	double	20.0
x2	double	30.0
y1	double	10.0
y2	double	10.0
speed	double	0.3
distance	double	0.0

- Как можно увидеть `distance` имеет значение 0,0, формула для вычисления расстояния была неверной, что привело к возврату неправильного значения для переменной `distance`.

# Определение потенциальной ошибки

```
public static void main(String[] args) {  
  
    double x1 = 20;  
    double x2 = 30;  
    double y1 = 10;  
    double y2 = 10;  
    double speed = 0.3;  
    double distance = getDistance(x1, x2, y1, y2);  
    double time = distance / speed;  
    System.out.println("Time taken to reach the gas station is " + time);  
  
}  
  
static double getDistance(double x1, double x2, double y1, double y2) {  
    return Math.sqrt((x1 - x1) * (x1 - x2) + (y1 - y2) * (y1 - y2));  
}  
}
```

Потенциальная ошибка

# Исправление ошибки

- Так как ошибка была определена, можно изменить расположение точки останова на то, в которой вызывается метод `getDistance()`.
- Это устраняет необходимость повторного просмотра кода.
- Поэтому изменим код и перезапустим отладчик с использованием новой точки останова, чтобы узнать получаемый результат.



# Повторный запуск отладчика

```
public static void main(String[] args) {  
    double x1 = 20;  
    double x2 = 30;  
    double y1 = 10;  
    double y2 = 10;  
    double speed = 0.3;  
    double distance = getDistance(x1, x2, y1, y2);  
    double time = distance / speed;  
    System.out.println("Time taken to reach the gas station is " + time);  
}  
  
static double getDistance(double x1, double x2, double y1, double y2) {  
    return Math.sqrt((x1 - x2) * (x1 - x2) + (y1 - y2) * (y1 - y2));  
}
```

Новая точка останова



Измененный код



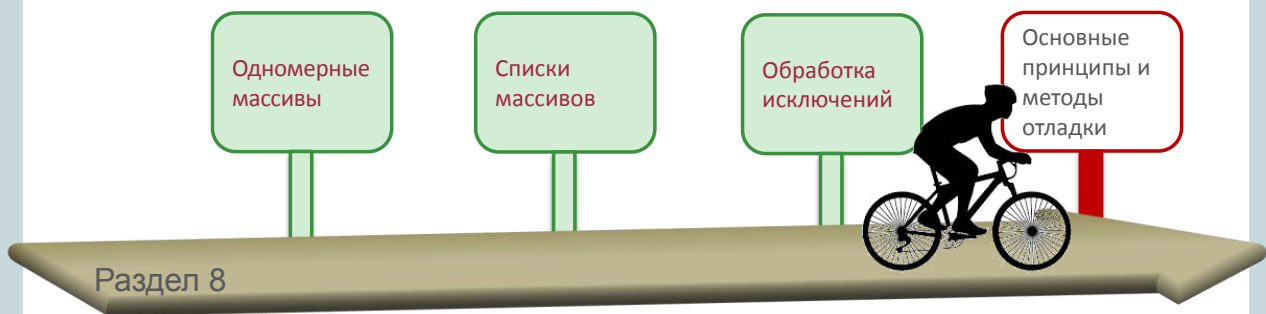
# Просмотр переменных

Variables		Output
Name	Type	Value
args	String[]	#/2(length=0)
x1	double	20.0
x2	double	30.0
y1	double	10.0
y2	double	10.0
speed	double	0.3
distance	double	10.0
time	double	33.333333333333336

- Ошибка исправлена!
- Переменная `distance` теперь сообщает о значении 10,0, а переменная `time` о значении 33,33.

# Темы

- Тестирование и отладка программы Java
- Три типа ошибок
- Методы отладки Операторы `print`
- Методы отладки Отладчик NetBeans
- Советы по отладке



**ORACLE**  
Academy

JFo 8-4  
Основные принципы и  
методы отладки

© 2019, Корпорация Oracle и аффилированные с ней лица. Все права защищены.

35

# Один по отношению к двум равен оператору

Сравнение операторов назначения (=) и сравнения (==)

## 1. Оператор сравнения

`if ( x = 0 )` вместо `if (x == 0)`

Ищите в операторах `if`, `for` и `while`.

## 2. Оператор назначения

`int x == 1;` вместо `int x = 1;`

# Неверно размещенная точка с запятой

Проверьте точку с запятой после оператора `if` или операторов цикла `for/while`.

```
if (x == 0) ; {  
    <операторы>  
}
```

instead of

```
if(x == 0) {  
    <операторы>  
}
```

```
while(<выражение boolean>) ; {  
    <операторы>  
}
```

instead of

```
while(<выражение boolean>) {  
    <операторы>  
}
```

# Вызов методов с помощью неверных аргументов

- Типы параметров вызова метода должны соответствовать типам параметров определения метода.

- Например:

- Указанное определение метода:

```
void methodName (int x, char y) { }
```

- Вызовите следующий метод:

```
methodName (a, b)
```



**a должно представлять int,  
а b должно представлять char.**

**ORACLE**

Academy

JFo 8-4  
Основные принципы и  
методы отладки

© 2019, Корпорация Oracle и аффилированные с ней лица. Все права защищены.

38

# Граничные условия

- Важно проверять **граничные условия**.
- Обоснование данного тестирования заключается в том, что ошибки могут происходить рядом с граничными значениями переменной ввода.
- Например, граничное условие для:
  - Данные ввода (проверка с помощью сравнения допустимости относительно недопустимости)
  - Циклы (начало и завершение циклов)

# Проверка граничных условий для циклов

- Это позволяет выполнять тестирование граничных случаев, таких как "меньше" и "больше" для точной проверки условий итераций циклов.
- Например, рассмотрим данный цикл:

```
if ( num >= 50 && num <= 100 ) {  
  //выполнить что-нибудь  
}
```

- Чтобы проверить граничные условия, необходимо выполнить проверку с помощью числовых значений, близких к 50 и 100, т.е. 49, 50, 51, 99, 100 и 101.





## Упражнение 3

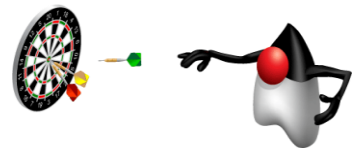
- Импортируйте и откройте проект `DebuggingEx`.
- Просмотрите `BoundaryTesting.java`.
- Проверьте ввод с помощью выполнения программы со следующими граничными тестовыми значениями для года и месяца:

Год	Месяц
1582	2
1583	0
1583	13
1583	1
1583	12

# Сводка

В этом уроке вы узнали следующее:

- Тестирование и отладка программы Java
- Определение трех типов ошибок
- Применение методов отладки
  - Операторы `print`
  - Отладчик NetBeans
- Советы и методы отладки



**ORACLE®**  
Academy

JFo 8-4  
Основные принципы и  
методы отладки

© 2019, Корпорация Oracle и аффилированные с ней лица. Все права защищены.

42

