



Java

Требования к лабораторным

- Соблюдаем код конвеншн
 1. Имен переменных – `exampleVar`
 2. Имена констант – `EXAMPLE_CONST`
 3. Имена методов – `exampleMethod()`
 4. Все «магические строки» и «магические числа» выносим в константы
 5. Методы не больше 25 строк
 6. Имена переменных четко отражают их назначение (исключения – стандартный нейминг в циклах `for`)
- Лучше не использовать то, что мы не проходили

В предыдущих сериях

- ООП – основная парадигма Java (но не единственная)
- Основные принципы ооп – полиморфизм, наследование, инкапсуляция
- Объект – сущность, имеет состояние и поведение
- Класс – шаблон, по которому создается объект
- Свойство – параметр класса, у объекта имеет конкретное значение
- Метод – функция, описывается в классе, принадлежит объекту, может использовать его состояние
- Конструктор класса – специальный метод, который «собирает объект по чертежу»

Глава 4.1

Статика в Java

Статика в Java

- Статическая переменная – переменная, значение которых будет одинаковое для всех экземпляров класса
- Чаще всего статические переменные используют с модификатором `final` (константа)
- Статический метод – в некотором роде «процедура» из процедурного программирования
- Статические методы имеют доступ только к статическим полям

Статика в Java

```
public abstract class Vehicle implements Movable, PassengerTransport {  
  
    private static final int MAX_PASSENGERS_COUNT = 3;  
    private static int VEHICLES_CREATED = 0;  
  
    public static int getVehiclesCreated() {  
        return VEHICLES_CREATED;  
    }  
  
    protected String brand;  
    protected Integer passengersCount;  
  
    public Vehicle(String brand) {  
        this.brand = brand;  
        this.passengersCount = 0;  
        VEHICLES_CREATED++;  
    }  
  
    public abstract void move(String destination);  
  
    public void sitIn(Integer passengersCount) {  
        if (this.passengersCount + passengersCount <= MAX_PASSENGERS_COUNT) {  
            this.passengersCount += passengersCount;  
            System.out.println("Теперь у нас " + passengersCount + " пассажиров");  
            return;  
        }  
        System.out.println("Невозможно посадить больше пассажиров");  
    }  
}
```

Final – константа
(нельзя изменить)

Изменяемая
статическая
переменная

Статический
метод

Статическое
состояние доступно
экземпляру

```
public static void main(String[] args) {  
    Car bmwCar = new Car( brand: "BMW", engineCapacity: 200);  
    Car toyotaCar = new Car( brand: "Toyota", engineCapacity: 80);  
    Plane plane = new Plane( brand: "Ty134");  
  
    plane.sitIn( passengersCount: 1);  
    plane.sitIn( passengersCount: 4);  
    System.out.println(String.format("Всего транспортных средств: %s", Vehicle.getVehiclesCreated()));  
}
```

Вызов статика через
имя класса

```
C:\Users\kondo\.jdk\corretto-15.0.2\bin\j  
Теперь у нас 1 пассажиров  
Невозможно посадить больше пассажиров  
Всего транспортных средств: 3  
  
Process finished with exit code 0
```

Вопросы и ответы



Глава 4.2

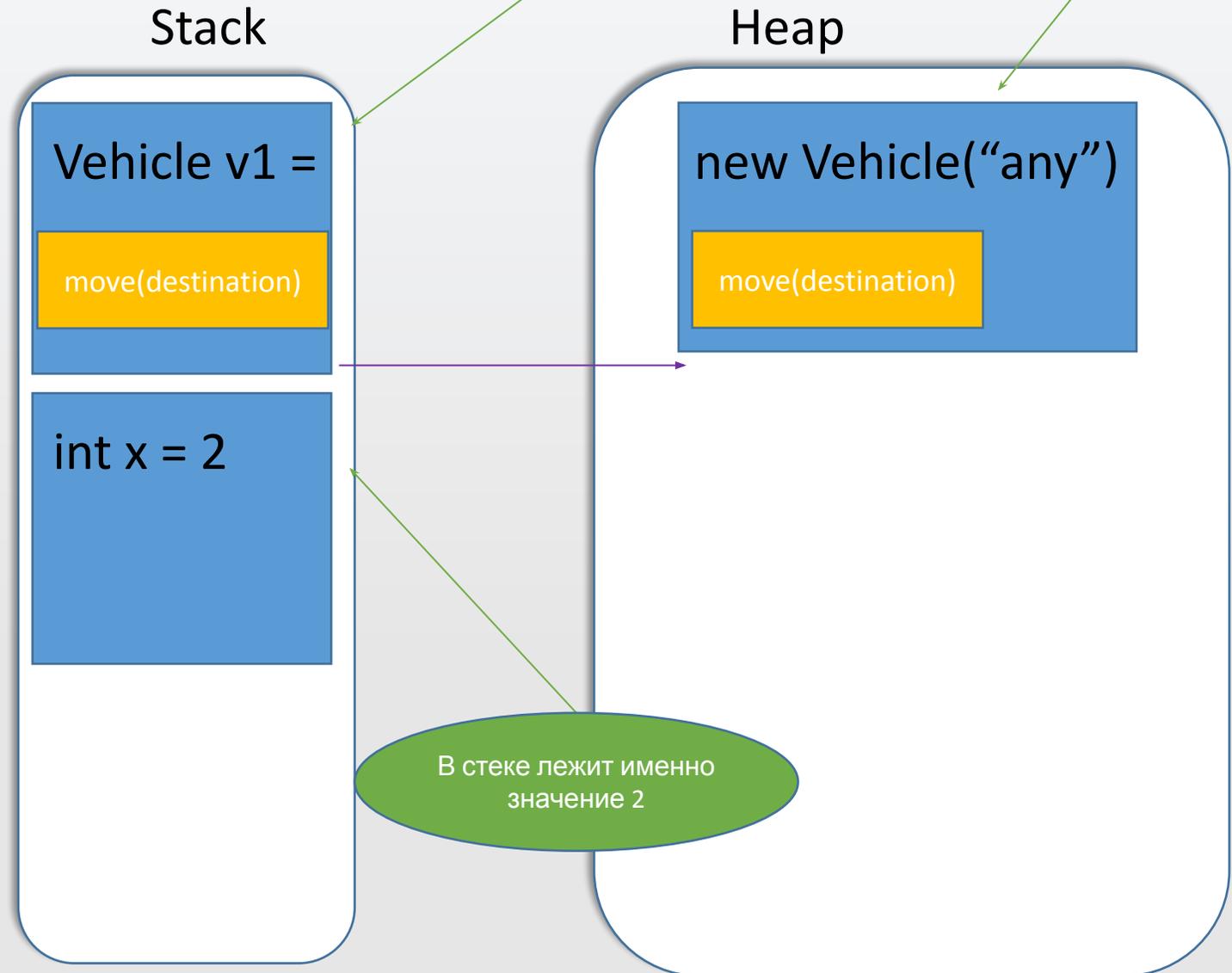
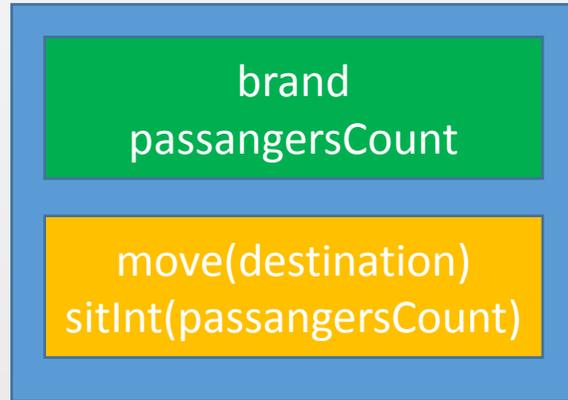
Передача по ссылке и по значению

Передача по ссылке и по значению

- В Java поведение при передаче внутрь методов разное у примитивов и объектов (ссылочных типов)
- Вспомним как устроена память в Java
- Stack – место для хранения примитивов и ссылок
- Heap (Куча) – хранит сами объекты

Память

Class Vehicle



Передача по ссылке и по значению

- Если в метод в качестве параметра передать примитив – передастся его **ЗНАЧЕНИЕ**
- Если в метод передать объект – передастся его **АДРЕС**
- Говорят, что в Java примитивы передаются по **ЗНАЧЕНИЮ** а ссылочные типы(объекты) по **ЗНАЧЕНИЮ ССЫЛКИ**

Передача примитивов

Мы передали в метод значение 3.

```
public static void main(String[] args) {  
    int x = 3;  
    changePrimitiveValue(x);  
    System.out.println(String.format("x = %s", x));  
}  
  
public static void changePrimitiveValue(int val){  
    val = 42;  
    System.out.println(String.format("val = %s", val));  
}  
}
```

Оригинальная переменная не измениться

```
Main x  
C:\Users\kondo\.jdk\corretto-15.0.2\bin\java.exe "-j  
val = 42  
x = 3  
  
Process finished with exit code 0
```

Передача ссылочных типов

```
public static void main(String[] args) {  
    int x = 3;  
    changePrimitiveValue(x);  
    System.out.println(String.format("x = %s", x));  
    int[] array = {1,2,3};  
    changeArrayValue(array);  
    System.out.println(String.format("array = %s", Arrays.toString(array)));  
}  
  
public static void changePrimitiveValue(int val){  
    val = 42;  
    System.out.println(String.format("val = %s", val));  
}  
  
public static void changeArrayValue(int[] val){  
    val[2] = 42;  
}
```

```
C:\Users\kondo\.jdk\corretto-15.0.2\bin  
val = 42  
x = 3  
array = [1, 2, 42]  
  
Process finished with exit code 0
```

Почему так?

```
1 ▶ public class Main {  
2  
3 ▶ public static void main(String[] args) {  
4     Integer x = 3;  
5     changeInteger(x);  
6     System.out.println(String.format("x = %s", x));  
7 }  
8  
9  
0 public static void changeInteger(Integer val){  
1     val = 42;  
2     System.out.println(String.format("val = %s", val));  
3 }  
4 }  
5
```

```
main *  
C:\Users\kondo\.jdk\corretto-15.0.2\bin  
val = 42  
x = 3  
  
Process finished with exit code 0
```



Неизменяемые классы

- (Неизменяемый) класс – класс, состояние которого нельзя изменить после его создания
- Как правило все поля неизменяемого класса имеют модификатор `final`
- Класс без состояние в каком то смысле тоже неизменяемый (хотя чаще говорят что он `Stateless`)
- Все обертки над примитивами в Java – `Immutable` (`Integer`, `Character` итд)
- Методы неизменяемого класса могут вернуть новый объект, но не могут изменить старый

Неизменяемые классы (Immutable)

```
public final class MyInteger {  
  
    private final int value;  
  
    public MyInteger(int value) {  
        this.value = value;  
    }  
  
    public MyInteger plus(int inc){  
        return new MyInteger( value: value + inc);  
    }  
}
```

Вопросы и ответы



Глава 4.3

Класс Object

Откуда взялись ЭТИ методы?

```
public static void main(String[] args) {
    Plane plane = new Plane(brand: "Ty134");
    plane.|
}
}
```

m	sitIn(Integer passengersCount)	void
m	move(String destination)	void
m	equals(Object obj)	boolean
<u>T</u>	arg	functionCall(expr)
m	hashCode()	int
m	toString()	String
m	getClass()	Class<? extends Plane>
m	notify()	void
m	notifyAll()	void
m	wait()	void
m	wait(long timeoutMillis)	void
m	wait(long timeoutMillis, int nanos)	void

Ctrl+Down and Ctrl+Up will move caret down and up in the editor [Next Tip](#)

Класс Object

- Вершина иерархии всех объектов
- На объекте основывается ООП в java
- Все методы, которые есть у объекта, будут у любого созданного класса.
- Методы класса Object сочетают в себе концепции объекта и концепции монитора(мьютекса из многопоточности)
- Пока мы не будем говорить про многопоточные методы `wait()`, `notify()`, `notifyAll()` и рефлексивный метод `getClass()`

Класс Object

- `toString` - превращает объект в строку
- `equals` – сравнивает объекты
- `hashCode` – возвращает хеш объекта

toString - превращает объект в строку

```
public class Plane extends Vehicle {  
  
    private Integer engineCount = 2;  
    private Integer wingsCount = 2;  
  
    public Plane(String brand) { super(brand); }  
  
    @Override  
    public void move(String destination) {  
        if (wingsCount == 2 && engineCount == 2) {  
            System.out.println("Я лечу в " + destination);  
        } else {  
            System.out.println("Что то не так с самолетом");  
        }  
    }  
  
    @Override  
    public String toString() {  
        return "Plane{" +  
            "engineCount=" + engineCount +  
            ", wingsCount=" + wingsCount +  
            ", brand='" + brand + '\'' +  
            ", passengersCount=" + passengersCount +  
            '}';  
    }  
}
```

Такую стандартную реализацию toString сделает IDEA, если переопределить его через ALT + INSERT

```
public class Main {  
  
    public static void main(String[] args) {  
        Plane plane = new Plane( brand: "ty134");  
        System.out.println(plane.toString());  
    }  
}
```

```
Main x  
C:\Users\kondo\.jdk\corretto-15.0.2\bin\java.exe "-javaagent:C:\Program F  
Plane{engineCount=2, wingsCount=2, brand='ty134', passengersCount=0}  
  
Process finished with exit code 0
```

equals - сравнение объектов в Java

```
public class Main {  
  
    public static void main(String[] args) {  
        Plane plane1 = new Plane( brand: "Ty134");  
        Plane plane2 = new Plane( brand: "Ty134");  
        System.out.println(plane1 == plane2);  
    }  
}
```

Почему false?



```
Main x  
C:\Users\kondo\.jdk\corretto-15.0.2\bin\java.exe "-ja  
false  
  
Process finished with exit code 0  
|
```

equals - сравнение объектов в Java

- Через “==” в Java сравниваются ссылки
- У plane1 и plane2 разные адреса в памяти (ссылки), поэтому и false
- Примитивы можно сравнивать на ==, т.к. они не ссылочные
- Для сравнение объектов “по смыслу” нужно переопределить метод equals
- По умолчанию equals определен как оператор сравнения “==”
- На переопределение equals накладывают ограничения

equals - сравнение объектов в Java

- Если `a.equals(b)`, то `b.equals(a)`
- `a.equals(a)`
- Если `a.equals(b)`, и `b.equals(c)`, то `a.equals(c)`
- Если объекты не изменялись, `equals` для них всегда возвращает один и тот же результат
- `a.equals(null)` возвращает `false`

hashCode - хеш функция для объектов в java

- Хеш функция (функция свертки) – любая функция для которой для одинаковых аргументов результат – одинаковое значение фиксированной длины
- Пример очень простой хеш функции – сложить все ‘буквы’ в строке, и взять остаток от деления на 10
- Для равных объектов хеш всегда равный
- Для разных объектов, хеш почти всегда разный, за исключением коллизий
- Часто используется для ‘ускорения’ операций

hashCode - хеш функция для объектов в java

- строка1 = "abc"
- строка2 = "abd"
- строка3 = "cba"
- Какие будут считаться для них hashCode?

hashCode - хеш функция для объектов в java

- Коллизия – ситуация, когда для разных объектов hashCode совпадает
- Возникает из-за того, что, результат хеш функции ограничен по длине (в Java – значениями int)
- По большому счету hashCode() = {return 42} тоже хеш функция, которая всегда будет выдавать коллизию

Связь equals и hashCode

- Методы equals и hashCode взаимосвязаны
- Как правило их переопределяют вместе
- Они должны зависеть от одних и тех же полей

Сгенерируем equals и hashCode через IDEA

```
public abstract class Vehicle implements Movable, PassengerTransport {  
  
    private static final int MAX_PASSENGERS_COUNT = 3;  
    private static int VEHICLES_CREATED = 0;  
  
    public static int getVehiclesCreated() {  
        return VEHICLES_CREATED;  
    }  
  
    protected String brand;  
    protected Integer passengersCount;  
  
    private final String serialNumber;  
  
    public Vehicle(String brand, String serialNumber) {  
        this.brand = brand;  
        this.serialNumber = serialNumber;  
        this.passengersCount = 0;  
        VEHICLES_CREATED++;  
    }  
  
    @Override  
    public boolean equals(Object o) {  
        if (this == o) return true;  
        if (o == null || getClass() != o.getClass()) return false;  
        Vehicle vehicle = (Vehicle) o;  
        return Objects.equals(serialNumber, vehicle.serialNumber);  
    }  
  
    @Override  
    public int hashCode() {  
        return Objects.hash(serialNumber);  
    }  
}
```

Добавили серийный номер

Тут главное, сравниваем серийные номера

Хеш зависит от серийного номера

Сгенерируем equals и hashCode через IDEA

```
public class Main {  
  
    public static void main(String[] args) {  
        Vehicle vehicle1 = new Plane( brand: "Ty134", serialNumber: "serial123");  
        Vehicle vehicle2 = new Plane( brand: "Ty134", serialNumber: "serial123");  
        Vehicle vehicle3 = new Plane( brand: "Ty134", serialNumber: "serial999");  
  
        System.out.println(vehicle1 == vehicle2);  
        System.out.println(vehicle2 == vehicle3);  
  
        System.out.println(vehicle1.equals(vehicle2));  
        System.out.println(vehicle2.equals(vehicle3));  
    }  
}
```

Равны только 1 и 2
транспорт на equals

```
Main x  
C:\Users\kondo\.jdk\corretto-15.0.2\  
false  
false  
true  
false
```

Boxing и Unboxing

```
public static void main(String[] args) {  
    int x = 3;  
    changeInteger(x);  
    Integer y = 4;  
    int z = y;  
}  
  
public static void changeInteger(Integer val){  
    val = 42;  
    System.out.println(String.format("val = %s", val));  
}
```

Boxing

Unboxing

Java автоматически
упаковывает/распаковывает примитив

Вопросы и ответы



Глава 4.4

Практика

Домашнее задание

- Создать иерархию из 1 предка и 3х наследников этого предка (пример, Музыкальные инструменты, Бытовая Техника, Животные). Использовать абстрактный класс
- В одном наследнике переопределить equals и hashCode. Продемонстрировать работу этих методов
- Во всех наследниках переопределить toString. Продемонстрировать полиморфизм.
- У предка должен быть хотя бы 1 публичный метод, продемонстрировать
- У одного из наследников должен быть приватный метод. Использовать этот метод внутри публичного метода (можно создать еще один публичный метод, можно переопределить существующий)