JAVAGURU INTRODUCTION TO JAVA

LESSON 3

COMMON MISTAKES AND PITFALLS OVERVIEW

OVERVIEW

1. Missing terminator sign ';'
2. Incorrect spelling
    2.1. Class name
    2.2. Package name
    2.3. Variable name
3. Code placement outside of the body
4. Missing quotes or misplacement

(1) MISSING TERMINATOR SIGN: THE CODE

```
public class ForgotSemicolonAgain {

    public static void main(String[] args) {
        System.out.println("Oops.. I did it again")
    }

}
```

(1) MISSING TERMINATOR SIGN: THE FIX

```
public class ForgotSemicolonAgain {

    public static void main(String[] args) {
        System.out.println("Oops.. I did it again");
    }

}
```

(2.1) BAD CLASS SPELLING: THE CODE

```
public class sizeMatters {

    public static void main(String[] args) {
        system.out.println("Sorry, it does");
    }

}
```

(2.1) BAD CLASS SPELLING: THE FIX

```
public class SizeMatters {

    public static void main(String[] args) {
        System.out.println("Sorry, it does");
    }

}
```

(2.2) BAD PACKAGE SPELLING: THE CODE

```
package lv.javaguru.lessons.HOMEWork;
```

(2.2) BAD PACKAGE SPELLING: THE CODE

```
package lv.javaguru.lessons.homework;
```

(3) CODE PLACEMENT OUTSIDE OF THE BODY: THE CODE

```
public class AttentionPlease {

    System.out.println("Hide and seek");
    public static void main(String[] args) {
    }

}
```

(3) CODE PLACEMENT OUTSIDE OF THE BODY: THE FIX

```
public class AttentionPlease {

    public static void main(String[] args) {
        System.out.println("Hide and seek");
    }

}
```

(4) QUOTES MISPLACEMENT: THE CODE

```
public class NoSleepNoFocus {

    public static void main(String[] args) {
        System.out.println(I wanna coffee);
        System.out.println("So bad);
    }

}
```

(4) QUOTES MISPLACEMENT: THE FIX

```
public class NoSleepNoFocus {

    public static void main(String[] args) {
        System.out.println("I wanna coffee");
        System.out.println("So bad");
    }

}
```

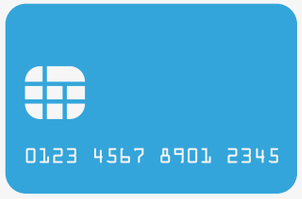
OBJECT ORIENTED
PROGRAMMING IN DEPTH

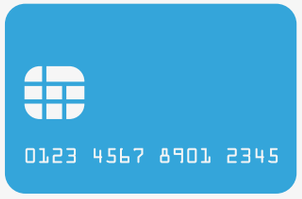
PRIMARY CONCEPTS: CLASS AND OBJECT

- Class describes template (blueprint) of something with state and behaviour
- Object is concrete instance of that class with set state

EXAMPLE: BANK CARD (STATE)

Class		Object
A. Bank Name	→	A. Citadele Banka
B. Payments Processor	→	B. Master Card
C. Name on Card	→	C. John Doe
D. Card Number	→	D. 5224 9989 7556 2871
E. Expiration Date	→	E. 12/2022
F. Security Code	→	F. 218

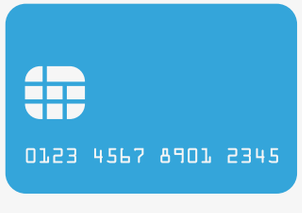




EXAMPLE: BANK CARD (BEHAVIOUR)

Class

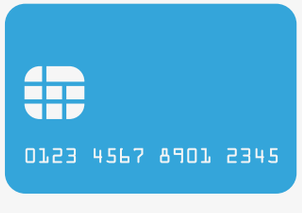
A. Get balance
B. Deposit funds
C. Withdraw funds



→

Object

A. Get balance
B. Deposit funds
C. Withdraw funds



CLASS DECLARATION IN JAVA: SYNTAX

```
class ClassName {

    type variable1;
    type variable2;
    ...
    type variableN;

    method1() {}
    method2() {}
    ...
    methodN() {}

}
```

CLASS DECLARATION IN JAVA: EXAMPLE BREAKDOWN

Special keyword

Class name

```
class Dog {

    String name;
    int age;
    boolean hungry;

    void bark() {

    }

    void sleep() {

    }

    void eat() {

    }

}
```

Class variables (fields) defining state

Class methods defining behaviour

OBJECT INSTANTIATION IN JAVA: SYNTAX

- Object instantiation without assignment

```
new Class();
```

- Object instantiation with assignment

```
Class var = new Class();
```

OBJECT INSTANTIATION IN JAVA: SYNTAX

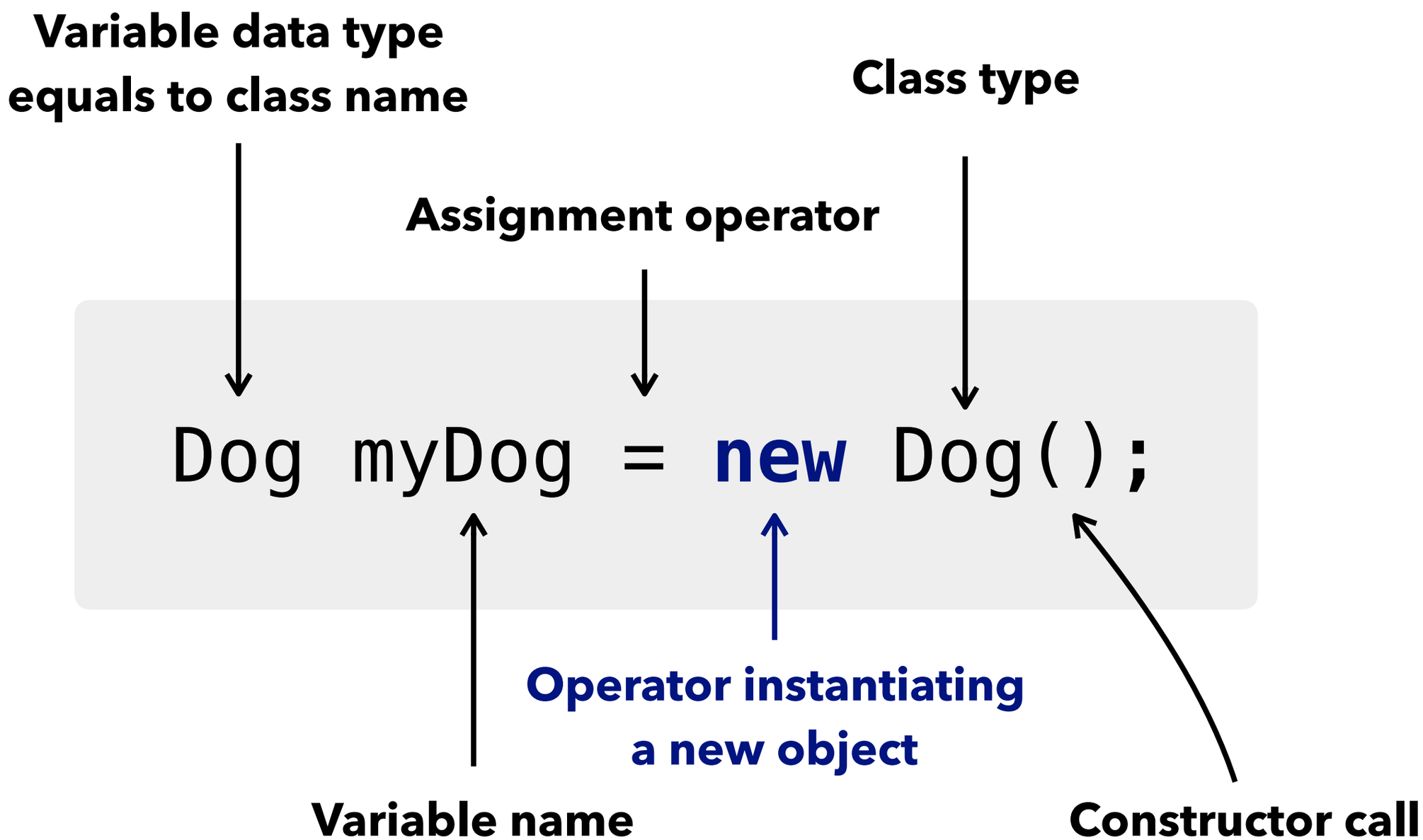
- Object instantiation without assignment

```
new Dog();
```

- Object instantiation with assignment

```
Dog myDog = new Dog();
```

OBJECT INSTANTIATION BREAKDOWN



THREE-STEP PROCESS OF OBJECT CREATION

1. Declaration - object variable declaration of a class type
2. Instantiation - the process of creating an object with new operator
3. Initialisation - the process of object construction by setting its initial state

CONSTRUCTORS

- Every class has a constructor
- If explicit constructor(s) is not specified in code, Java Compiler will generate default constructor implicitly
- Each time a new object is created, at least one constructor will be invoked
- Each defined constructor must have unique signature (i.e. ordered number and type of arguments)

CONSTRUCTOR DECLARATION IN JAVA: EXAMPLE BREAKDOWN

Explicit default constructor without arguments

```
public class Dog {

    private String name;

    public Dog() {

    }

    public Dog(String name) {
        this.name = name;
    }

}
```

Explicit constructor with argument and initialisation

MEMORY OVERVIEW

MEMORY TYPES

- Java Heap Memory
  - Created objects are stored in the heap space
  - Lives from the start till the end of application execution
  - Objects stored in heap are globally accessible
- Java Stack Memory
  - Contains local primitive variables and reference variables to objects in heap space
  - Lives only within method execution, short-lived
  - Bound to the current execution thread

METHODS OVERVIEW

METHOD DEFINITION

- Java method is a collection of statements that are grouped together to perform an operation
  - Invoking *System.out.println()* method actually executes several statements in order to display a message on the console
- Describes behaviour of class or actions that object can perform
- Method either produces output or not

METHOD DECLARATION IN JAVA: SYNTAX

Defines the access type of the method

Defines method name

```
modifier returnType methodName (arg1, arg2, ..., argN) {

    //body

}
```

Specifies return type for methods producing output

List of parameters, it is type, order and number

METHOD DECLARATION IN JAVA: WITH RETURN EXAMPLE

Can be accessed from everywhere

Result type

Method name

Method accepts two integer parameters

```
public int sum(int a, int b) {
    int result = a + b;
    return result;
}
```

Operation with parameters

Keyword with variable that returns value back to the invoker

METHOD DECLARATION IN JAVA: WITHOUT RETURN EXAMPLE

Method name

Keyword meaning no return value

Method accepts two integer parameters

```
public void printSum(int a, int b) {
    int result = a + b;
    System.out.println("Sum of numbers is" + result);
}
```

Operations with parameters

A BIT MORE ABOUT RETURNING RESULT

- After completion method returns to the code that invoked it
- Whether method returns value or not is declared in method signature
  - When type is void - return statement is unnecessary, however can be stated
  - Other type - return statement is necessary

ACCESSING AND CHANGING OBJECT STATE: GETTERS & SETTERS

- In OOP another party should not be able to access object state directly
- To keep things safe, one can
  - Retrieve object state via get methods (getters)
  - Change object state via set methods (setters)

GETTERS & SETTERS DECLARATION

```
public class Person {

    private String name;
    private int age;

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public int getAge() {
        return age;
    }

    public void setAge(int age) {
        this.age = age;
    }
}
```

Getters

Setters

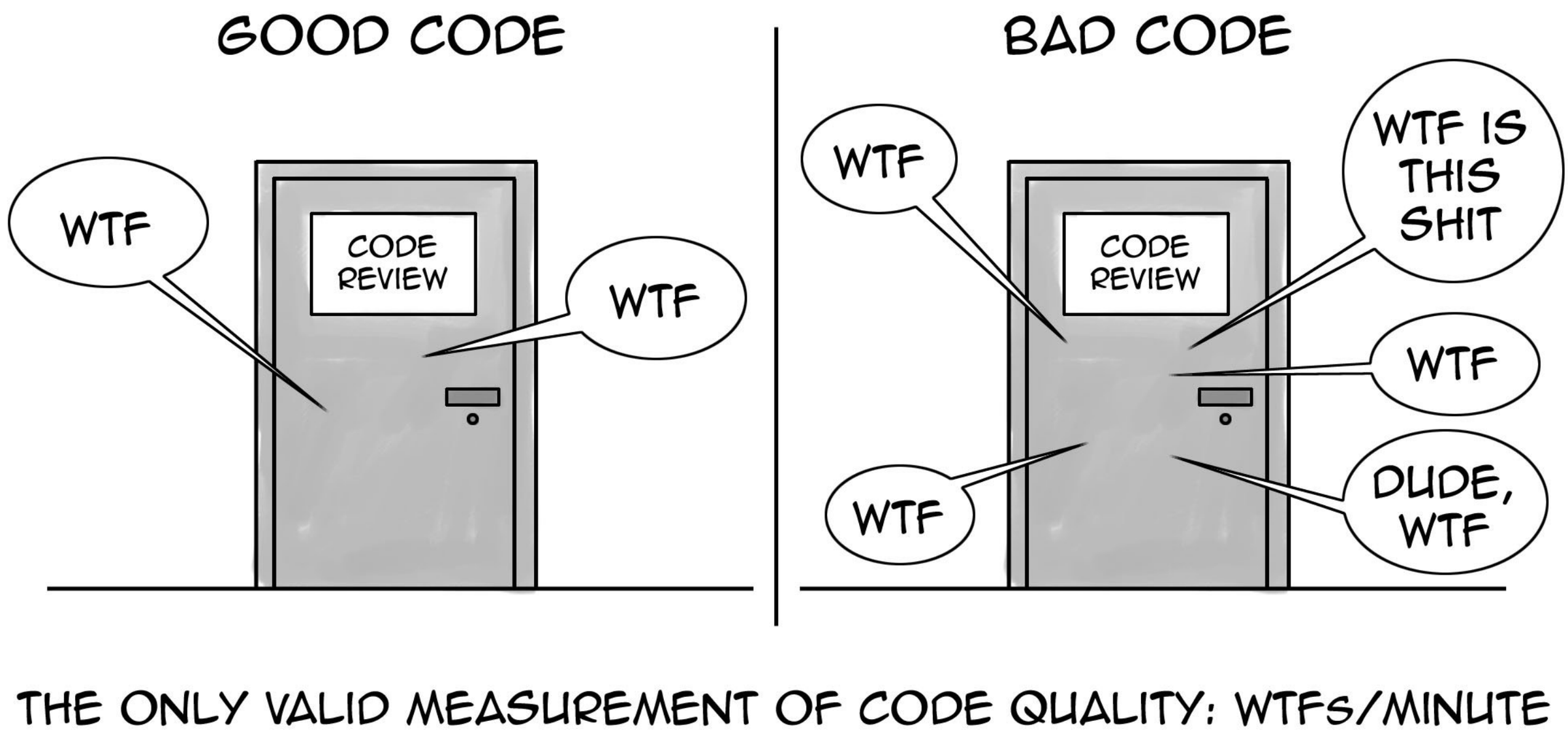
GETTERS & SETTERS USAGE

```java
public class PersonTest {

    public static void main(String[] args) {

        Person person = new Person();
        person.setName("John Doe");
        person.setAge(32);

        String personName = person.getName();
        int personAge = person.getAge();

        System.out.println("His name is " + personName);
        System.out.println("He is " + personAge + " years old");

    }

}
```

CLEAN CODE PRACTICES

ANY FOOL CAN WRITE CODE THAT COMPUTER UNDERSTAND. GOOD PROGRAMMERS WRITE CODE THAT HUMANS CAN UNDERSTAND.

Martin Fowler


GOOD CODE
WTF
CODE REVIEW
WTF
BAD CODE
WTF
CODE REVIEW
WTF IS THIS SHIT
WTF
WTF
DUDE, WTF
THE ONLY VALID MEASUREMENT OF CODE QUALITY: WTFs/MINUTE

BAD CODE AND GOOD CODE

Bad

```java
public class Cat {

    private String n;

    public String getN() {
        return n;
    }

    public void setN(String n) {
        this.n = n;
    }

    public void v() {
        System.out.println("Meow");
    }

}
```

Good

```java
public class Cat {

    private String name;

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public void voice() {
        System.out.println("Meow");
    }

}
```

REFERENCES

- https://docs.oracle.com/javase/tutorial/java/javaOO/methods.html
- https://www.tutorialspoint.com/java/java_methods.htm