2. Объекты

Программирование на Java

Автор курса: Федор Лаврентьев

Лектор: Виктор Яковлев

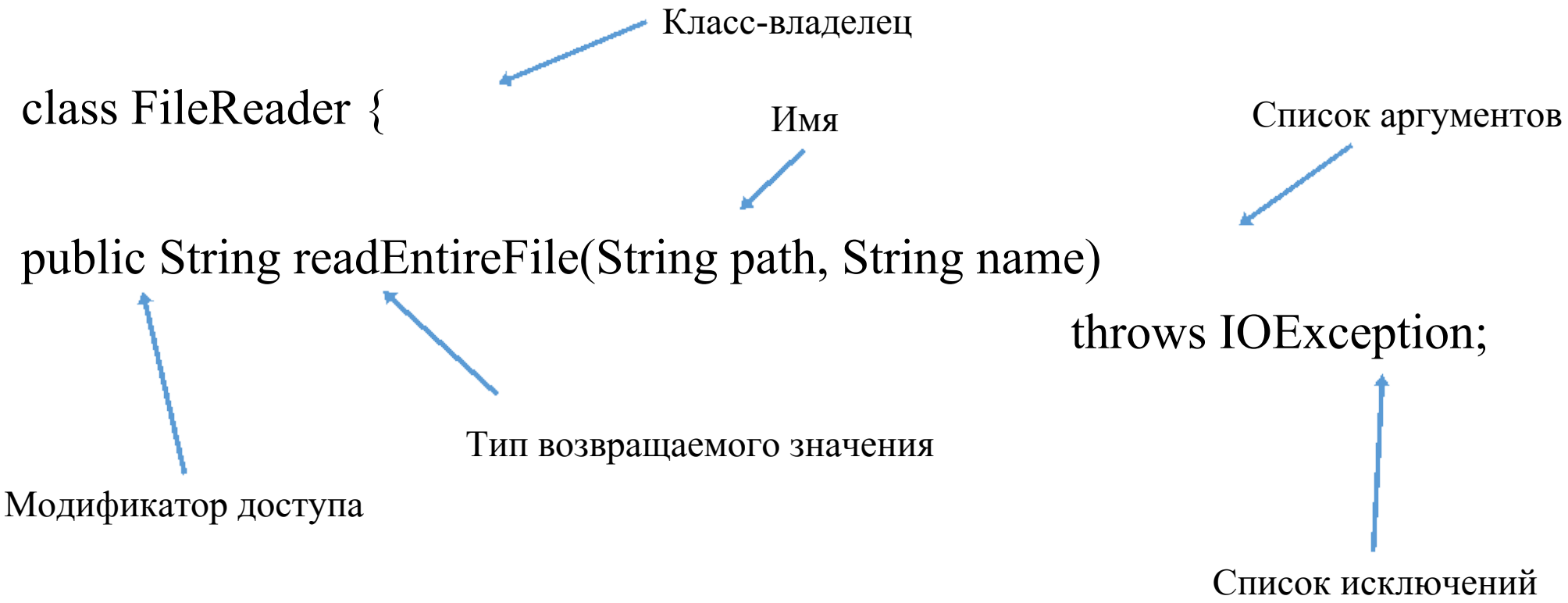
МФТИ, 2016-2017

Методы

Методы класса Object

- equals(), hashCode()
- clone()
- toString()
- finalize()
- wait(), notify(), notifyAll()
- getClass()

Декларация метода



Сигнатура метода

```
class FileReader {

public String readEntireFile(String path, String name)
                                                    throws IOException;
```

- Сигнатура = имя + список аргументов

Перегрузка метода

```
public String do() { ... }

public String do(String s) { ... }

public int do(Object obj) { ... }

public int do(int i) { ... }

public long do(int i) { ... }
```

Перегрузка метода

```
public String do() { ... }

public String do(String s) { ... }

public int do(Object obj) { ... }

public int do(int i) { ... }

public long do(int i) { ... } // WRONG!
```

Перегрузка метода - пример

```
public String do(String s) { ... }

public String do(int i) {
  return do(Integer.toString(i));
}

public String do() {
  return do(“Just do it”);
}
```

Инициализация объекта

Конструктор объекта

```
public class Animal {
  private String name;

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

  public Animal() {
    this.name = “Untitled animal”;
  }
}

Animal a = new Animal(“Monkey”);
```

Конструктор объекта

```
public class Animal {
  private String name;

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

  public Animal() {
    this(“Untitled animal”);
  }
}

Animal a = new Animal(“Monkey”);
```

Конструктор по умолчанию

```
public class Dog {
  String name;
  String breed;
  int age;

}
```

```
Dog dog = new Dog();
System.out.println(dog.name);
dog.name = “Pooker”;
```

Конструктор по умолчанию

```
public class Dog {
  String name = null;
  String breed = null;
  int age = 0;

  public Dog() { /* do nothing */ }
}

Dog dog = new Dog();
System.out.println(dog.name); // null
dog.name = “Pooker”;
```

Значения по умолчанию

```
public class Dog {
  String name = ”Nameless”;
  String breed = “Mongrel”;
  int age = 1;
}
```

```
Dog dog = new Dog();
System.out.println(dog.name); // ”Nameless”
System.out.println(dog.age);  // 1
```

Final поля

```
public class Dog {
  final String name;
  final String breed;
}
```

```
Dog dog = new Dog();
dog.name = “Pooker”;
dog.breed = “Bloodhound”;
```

Final поля

```
public class Dog {
  final String name;
  final String breed;
}                          // WRONG!
```

```
Dog dog = new Dog();       // WRONG!
dog.name = “Pooker”;       // WRONG!
dog.breed = “Bloodhound”; // WRONG!
```

Final поля

```
public class Dog {
  final String name;
  final String breed;

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

```
Dog dog = new Dog(“Pooker”, “Bloodhound”);
```

Final поля

```
pulic class Dog {
  final String name;
  final String breed;

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

  public Dog(String name) {
    this(name, “Mongrel”);
  }

  public Dog() {
    this(“Nameless”);
  }
}
```

Final объекты

```
class IntWrapper {
  int i;
  IntWrapper(int i) { this.i = i };
}

class FinalIntWrapper {
  final IntWrapper wrapper = new IntWrapper(1);
  ...
  wrapper = new IntWrapper(2);

}
```

Final объекты

```
class IntWrapper {
  int i;
  IntWrapper(int i) { this.i = i };
}

class FinalIntWrapper {
  final IntWrapper wrapper = new IntWrapper(1);
  ...
  wrapper = new IntWrapper(2); // WRONG

}
```

Final объекты

```
class IntWrapper {
  int i;
  IntWrapper(int i) { this.i = i };
}

class FinalIntWrapper {
  final IntWrapper wrapper = new IntWrapper(1);
  ...
  wrapper = new IntWrapper(2); // WRONG
  wrapper.i = 3;
}
```

Final объекты

```
class IntWrapper {
  int i;
  IntWrapper(int i) { this.i = i };
}

class FinalIntWrapper {
  final IntWrapper wrapper = new IntWrapper(1);
  ...
  wrapper = new IntWrapper(2); // WRONG
  wrapper.i = 3;                    // but right o_O
}
```

Статические поля и методы

```
class Integer {
  static final int MAX_VALUE = 2147483647;

  static final String VALUE_NAME = “int32”;

  static String toString(int i) { ... };
}

class Foo { ...
  String s = Integer.VALUE_NAME + “ ” +
      Integer.toString(Integer.MAX_VALUE));
}
```

Ловушка – порядок инициализации

```
class Utils {
  static final String NAME = buildName(“a”);

  static final String PREFIX = “utils_”;

  static String buildName(String s) {
    return PREFIX + s;
  }
}
```

Ловушка – порядок инициализации

```
class Utils {
  static final String NAME = buildName(“a”); // 1

  static final String PREFIX = “utils_”;     // 5

  static String buildName(String s) {        // 2
    return PREFIX + s;                       // 3
  }                                          // 4
}
```

Ловушка – порядок инициализации

```
class Utils {
  static final String NAME = buildName(“a”); // 1

  static final String PREFIX = “utils_”;     // 5

  static String buildName(String s) {        // 2
    return PREFIX + s; // PREFIX = null!     // 3
  }                                          // 4
}
```

Ловушка – цикл инициализации классов

```
class Foo {
  static Bar BAR = new Bar();
  static String NAME = “n”;
}

class Bar {
  static String s = Nja.default();
}

class Nja {
  static String default() { return Foo.NAME }
}
```

Ловушка – цикл инициализации классов

```
class Foo {
  static Bar BAR = new Bar();                 // 1
  static String NAME = “n”;                   // 4
}

class Bar {
  static String s = Nja.default();            // 2
}

class Nja {
  static String default() { return Foo.NAME } // 3
}
```

Статические блоки инициализации

```
class NamedPerson {
  static final String ABC;

  static {
    StringBuilder sb = new StringBuilder();
    for (char c = ‘a’; c <= ‘z’; ++c) {
      sb.append(c);
    }
    ABC = sb.toString();
  }
}
```

Статические блоки инициализации не нужны

```
class NamedPerson {
  static final String ABC = buildAbc();

  static String buildAbc() {
    StringBuilder sb = new StringBuilder();
    for (char c = ‘a’; c <= ‘z’; ++c) {
      sb.append(c);
    }
    return sb.toString();
  }
}
```

Удаление объекта

Область видимости переменной

```
public String swarmBark() {
  StringBuilder sb = new StringBuilder();
  Dog pooker = new Dog(“pooker”);

  sb.append(pooker.bark());

  for (int i = 0; i < 10; ++i) {
    Dog replier = new Dog(“Dog “ + i);
    sb.append(replier.bark());
  }

  return sb.toString();
}
```

Деструкторы в Java

- Деструкторов нет
- Реакции на область видимости нет
- Есть сборщик мусора

Сборка мусора

- Терпи, пока память есть
- Найди* все** недостижимые*** объекты
- Вызови для каждого object.finalize()
- Удали их из памяти

Освобождение ресурсов

```
public void printFile(String fileName) [...] {
  BufferedReader reader = new BufferedReader(
      new FileReader(fileName));

  String line;
  while ((line = reader.readLine()) != null) {
    System.out.println(line);
  }
  reader.close();
}
```

Освобождение ресурсов (плохой пример)

```
public void printFile(String fileName) [...] {
  BufferedReader reader = new BufferedReader(
      new FileReader(fileName));

  String line;
  while ((line = reader.readLine()) != null) {
    System.out.println(line);
  }
  reader.close(); // ALL WRONG
}
```

Исключения

Исключения

```
public int modulo(int a, int b) {


  int r = a % b;
  if (r > 0 && a < 0) {
    r -= n;
  }
}
```

Исключения

```
public int modulo(int a, int b) {


  // what if b == 0 ?


  int r = a % b;
  if (r > 0 && a < 0) {
    r -= n;
  }
}
```

Исключения

```
public int modulo(int a, int b) {

  if (b == 0) {
    throw new Exception(“Division by zero”);
  }

  int r = a % b;
  if (r > 0 && a < 0) {
    r -= n;
  }
}
```

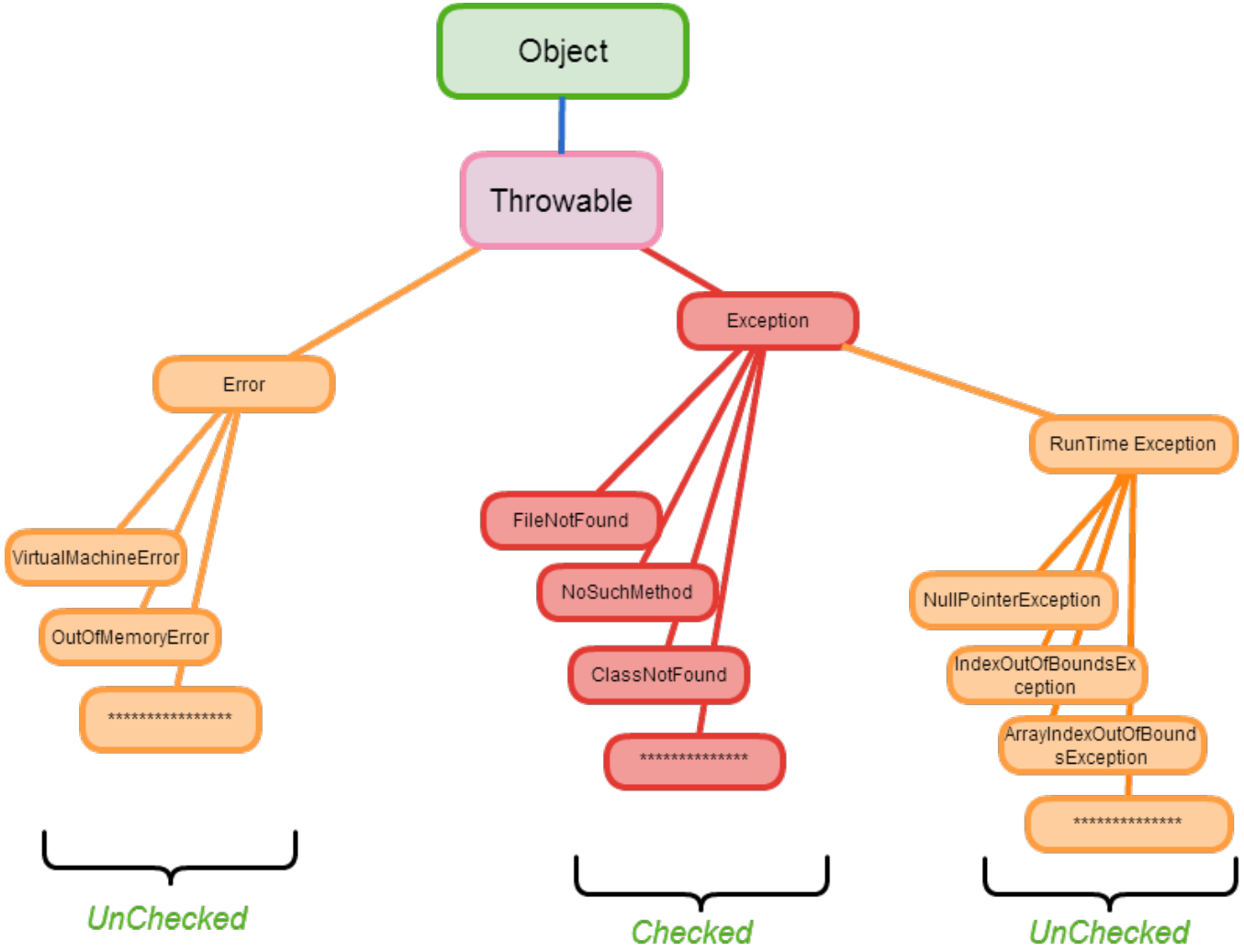
Исключения

```
public int modulo(int a, int b) {

  if (b == 0) {
    throw new DivisionByZeroException();
  }

  int r = a % b;
  if (r > 0 && a < 0) {
    r -= n;
  }
}
```

Исключения

```
public int modulo(int a, int b)
    throws DivisionByZeroException {
  if (b == 0) {
    throw new DivisionByZeroException();
  }

  int r = a % b;
  if (r > 0 && a < 0) {
    r -= n;
  }
}
```

Иерархия исключений



http://www.ufthelp.com/2014/11/exception-handling-in-java.html

Ловля исключений

```
void touch(String name) throws IOException {...}

void refreshFile(String name) {
  try {
    touch(name);
  } catch (FileNotFoundException e) {
    // It’s ok, do nothing
  } catch (EOFException|SyncFailedException e) {
    e.printStackTrace();
  } catch (IOException e) {
    throw new IllegalStateException(e);
  }
}
```

Finally

```
public void printFile(String fileName) [...] {
  BufferedReader reader = new BufferedReader(
      new FileReader(fileName));

  String line;
  while ((line = reader.readLine()) != null) {
    System.out.println(line);
  }
  reader.close();
}
```

Finally

```
public void printFile(String fileName) [...] {
  BufferedReader reader = null;
  try {
    reader = new BufferedReader(
        new FileReader(fileName));

    String line;
    while ((line = reader.readLine()) != null) {
      System.out.println(line);
    }
  } finally {
    reader.close();
  }
}
```

Finally

```
public void printFile(String fileName)
    throws IOException {
  BufferedReader reader = null;
  try {
    reader = new BufferedReader(
        new FileReader(fileName));

    String line;
    while ((line = reader.readLine()) != null) {
      System.out.println(line);
    }
  } finally {
    if (reader != null) reader.close();
  }
}
```

Try-with-resources

```
public void printFile(String fileName)
    throws IOException {
  try (BufferedReader reader =
      new BufferedReader(
          new FileReader(fileName))) {

    String line;
    while ((line = reader.readLine()) != null) {
      System.out.println(line);
    }
  }
}
```

Try-with-resources

```
try (AutoCloseable closeable = ...) {
  // Logic
} catch (Exception e) {
  // Handle exceptions
} finally {
  // Finish him
}
```

Finally и return

```
public String  kickStudentsBrain(String s) {
  try {
    return s.toLowerCase();
  } finally {
    throw new IllegalStateException();
  }
}
```

Finally и return

```
public String smashStudentsBrain(String s) {
  try {
    return s.toLowerCase();
  } finally {
    return s.toUpperCase();
  }
}
```