

Contents

1	□ Лекция 1 — Основы синтаксиса Java + Настройка окружения и работа в IntelliJ IDEA	2
1.1	□ Основные темы	2
1.2	□ Настройка Java-окружения	2
1.2.1	Установка JDK	2
1.2.2	Переменная окружения PATH	3
1.2.3	CLASSPATH	3
1.3	□ Работа с IntelliJ IDEA	3
1.3.1	Почему IntelliJ IDEA?	3
1.4	☞ Горячие клавиши IntelliJ IDEA (must-have)	3
1.5	□ Полезные Live Templates (шаблоны кода)	4
1.6	□ Рефакторинги в IntelliJ IDEA	4
1.6.1	□ Extract Method (Вынести метод)	4
1.6.2	□ Inline Method (Встроить метод)	5
1.7	□ Основы синтаксиса Java	5
1.7.1	Структура программы	5
1.7.2	Переменные и типы	5
1.7.3	Управляющие конструкции	5
1.8	□ Советы для новичков	6
1.9	□ Полезные ссылки	6
1.10	□ Передача параметров по значению	8
1.10.1	Пример:	8
1.10.2	Но:	8
1.11	□ Пакеты Java	8
1.11.1	Основные пакеты:	8
1.11.2	Импорт:	9
1.12	□ Класс Object — корень иерархии	9
1.12.1	Основные методы:	9
1.13	1.1.1 equals	9
1.13.1	□ Поведение по умолчанию	9
1.13.2	□ Переопределение	10
1.13.3	□ Контракт equals()	10
1.14	1.1.2 hashCode	10
1.14.1	□ Контракт hashCode()	10
1.14.2	□ Реализация по умолчанию	11
1.14.3	□ Почему важно переопределять hashCode() вместе с equals()	11
1.15	□ Правильное переопределение	11
1.15.1	Способ 1: Вручную	11
1.15.2	Способ 2: Через Lombok (если используется)	12
1.15.3	Способ 3: В IntelliJ IDEA → Alt + Insert → equals() and hashCode()	12
1.16	□ Частые ошибки	12
1.17	□ Советы	12
1.18	□ Контракт equals() и hashCode()	13
1.18.1	Почему это важно?	13
1.18.2	Антипаттерн:	13

1.19	Частые ошибки	13
1.20	Переполнение (Overflow)	13
1.20.1	Пример с int:	13
1.20.2	Как избежать?	14
1.21	BigDecimal — для точных вычислений (деньги!)	14
1.21.1	Пример проблемы:	14
1.21.2	Решение — BigDecimal:	14
1.21.3	Операции:	14
1.22	StringBuilder — эффективная работа со строками	15
1.22.1	Проблема:	15
1.22.2	Решение — StringBuilder:	15
1.22.3	Основные методы:	15
1.23	Советы	15
1.24	Полезные ссылки	16

1 Лекция 1 — Основы синтаксиса Java + Настройка окружения и работа в IntelliJ IDEA

1.1 Основные темы

- Установка и настройка JDK
 - Компиляция и запуск через javac и java
 - Переменные среды: PATH, CLASSPATH
 - Выбор и настройка IDE (IntelliJ IDEA)
 - Горячие клавиши и рефакторинги в IntelliJ IDEA
 - Основы синтаксиса: классы, методы, переменные, управляющие конструкции
-

1.2 Настройка Java-окружения

1.2.1 Установка JDK

→ Скачать можно с: - [Oracle JDK](#) - [OpenJDK \(Adoptium / Temurin\)](#) ← **рекомендуется**
 Проверка установки:

```
\ExtensionTok{java} \AttributeTok{{-}version}
\ExtensionTok{javac} \AttributeTok{{-}version}
```

→ Должны вывести версию Java и компилятора.

1.2.2 Переменная окружения PATH

→ PATH — список директорий, где система ищет исполняемые файлы.

□ Добавить путь к bin JDK в PATH:

Linux/macOS (в ~/.bashrc или ~/.zshrc):

```
\BuiltInTok{export} \VariableTok{PATH}\OperatorTok{=}\StringTok{"/path/to/jdk/bin:}\VariableTok{$PATH}\StringTok{}
```

Windows: - Панель управления → Система → Дополнительные параметры → Переменные среды → PATH → Добавить путь, например:

C:\Program Files\Java\jdk-21\bin

1.2.3 CLASSPATH

→ Указывает JVM, где искать .class-файлы и библиотеки.

□ Обычно не нужно настраивать вручную при работе с IDE или Maven/Gradle.

→ Если компилируешь вручную:

```
\ExtensionTok{javac} \AttributeTok{-cp} \StringTok{".;lib/*"}\NormalTok{ MyClass.java}
\ExtensionTok{java} \AttributeTok{-cp} \StringTok{".;lib/*"}\NormalTok{ MyClass}
```

□ . — текущая директория.

□ ; — разделитель в Windows, : — в Linux/macOS.

1.3 □ Работа с IntelliJ IDEA

1.3.1 Почему IntelliJ IDEA?

- Самая популярная и мощная IDE для Java.
- Умное автодополнение, рефакторинг, отладка, интеграция с Maven/Gradle, Git.
- Community Edition — бесплатна и достаточно для обучения.

1.4 🖱️ Горячие клавиши IntelliJ IDEA (must-have)

Действие	Windows/Linux	macOS
Автодополнение	Ctrl + Space	Cmd + Space
Быстрое исправление / подсказки	Alt + Enter	Option + Enter
Запуск программы	Shift + F10	Ctrl + R
Отладка	Shift + F9	Ctrl + D
Поиск по проекту	Ctrl + Shift + F	Cmd + Shift + F
Поиск класса	Ctrl + N	Cmd + O
Поиск файла	Ctrl + Shift + N	Cmd + Shift + O

Действие	Windows/Linux	macOS
Переход к определению	Ctrl + B	Cmd + B
Рефакторинг: переименование	Shift + F6	Shift + F6
Закомментировать строку	Ctrl + /	Cmd + /
Форматирование кода	Ctrl + Alt + L	Cmd + Option + L
Открыть структуру класса	Ctrl + F12	Cmd + F12

1.5 ☐ Полезные Live Templates (шаблоны кода)

Шаблон	Результат	Описание
sout	System.out.println();	Быстрый вывод в консоль
iter	for (Type item : collection) { }	Цикл for-each
psvm	public static void main(String[] args) { }	Главный метод
itar	for (int i = 0; i < arr.length; i++) { }	Цикл по индексу
ifn	if (var == null) { }	Проверка на null
inn	if (var != null) { }	Проверка на не-null

→ Просто введи шаблон и нажми Tab.

1.6 ☐ Рефакторинги в IntelliJ IDEA

1.6.1 ☐ Extract Method (Вынести метод)

Выдели код → Ctrl + Alt + M → дай имя методу → готово!

Было:

```
\KeywordTok{public} \DataTypeTok{void} \FunctionTok{process}\OperatorTok{()} \OperatorTok{\}
  \DataTypeTok{int}\NormalTok{ a }\OperatorTok{=} \DecValTok{5}\OperatorTok{;}
  \DataTypeTok{int}\NormalTok{ b }\OperatorTok{=} \DecValTok{10}\OperatorTok{;}
  \DataTypeTok{int}\NormalTok{ sum }\OperatorTok{=} \NormalTok{ a }\OperatorTok{+}\NormalTok{ b }\OperatorTok{;}
  \BuiltInTok{System}\OperatorTok{.}\FunctionTok{out}\OperatorTok{.}\FunctionTok{println}\OperatorTok{()}\Stri
\OperatorTok{\}}
```

Стало:

```
\KeywordTok{public} \DataTypeTok{void} \FunctionTok{process}\OperatorTok{()} \OperatorTok{\}
  \FunctionTok{printSum}\OperatorTok{()}\DecValTok{5}\OperatorTok{,} \DecValTok{10}\OperatorTok{;}
\OperatorTok{\}

\KeywordTok{private} \DataTypeTok{void} \FunctionTok{printSum}\OperatorTok{()}\DataTypeTok{int}\NormalTok{ a }\Op
  \DataTypeTok{int}\NormalTok{ sum }\OperatorTok{=} \NormalTok{ a }\OperatorTok{+}\NormalTok{ b }\OperatorTok{;}
  \BuiltInTok{System}\OperatorTok{.}\FunctionTok{out}\OperatorTok{.}\FunctionTok{println}\OperatorTok{()}\Stri
\OperatorTok{\}}
```

→ Улучшает читаемость и переиспользование.

1.6.2 Inline Method (Встроить метод)

Если метод слишком простой — можно “встроить” его обратно:

Ctrl + Alt + N

→ Полезно при оптимизации или упрощении.

1.7 Основы синтаксиса Java

1.7.1 Структура программы

```
\KeywordTok{public} \KeywordTok{class}\NormalTok{ HelloWorld }\OperatorTok{\{}  
    \KeywordTok{public} \DataTypeTok{static} \DataTypeTok{void} \FunctionTok{main}\OperatorTok{\}\BuiltInTok{St  
        \BuiltInTok{System}\OperatorTok{.}\FunctionTok{out}\OperatorTok{.}\FunctionTok{println}\OperatorTok{\}  
    \OperatorTok{\}}  
\OperatorTok{\}}
```

→ Каждая программа начинается с main.

1.7.2 Переменные и типы

```
\DataTypeTok{int}\NormalTok{ age }\OperatorTok{=} \DecValTok{25}\OperatorTok{;}  
\DataTypeTok{double}\NormalTok{ price }\OperatorTok{=} \FloatTok{19.99}\OperatorTok{;}  
\DataTypeTok{boolean}\NormalTok{ isActive }\OperatorTok{=} \KeywordTok{true}\OperatorTok{;}  
\BuiltInTok{String}\NormalTok{ name }\OperatorTok{=} \StringTok{"Alice"}\OperatorTok{;}
```

1.7.3 Управляющие конструкции

```
\ControlFlowTok{if} \OperatorTok{\}\NormalTok{age }\OperatorTok{\textgreater{}} \DecValTok{18}\OperatorTok{\}  
    \BuiltInTok{System}\OperatorTok{.}\FunctionTok{out}\OperatorTok{.}\FunctionTok{println}\OperatorTok{\}\Stri  
\OperatorTok{\}} \ControlFlowTok{else} \OperatorTok{\}  
    \BuiltInTok{System}\OperatorTok{.}\FunctionTok{out}\OperatorTok{.}\FunctionTok{println}\OperatorTok{\}\Stri  
\OperatorTok{\}}  
  
\ControlFlowTok{for} \OperatorTok{\}\DataTypeTok{int}\NormalTok{ i }\OperatorTok{=} \DecValTok{0}\OperatorTok{;}  
    \BuiltInTok{System}\OperatorTok{.}\FunctionTok{out}\OperatorTok{.}\FunctionTok{println}\OperatorTok{\}\Norm  
\OperatorTok{\}}  
  
\ControlFlowTok{while} \OperatorTok{\}\NormalTok{condition}\OperatorTok{\}} \OperatorTok{\}  
    \CommentTok{// ...}  
\OperatorTok{\}}
```

1.8 ☐ Советы для новичков

- Всегда проверяй, что `java -version` работает в терминале.
 - Не бойся использовать `Alt + Enter` — IntelliJ IDEA часто знает, как исправить ошибку.
 - Учись пользоваться рефакторингами — они экономят кучу времени.
-

1.9 ☐ Полезные ссылки

- [Скачать IntelliJ IDEA Community](#)
- [OpenJDK \(Adoptium\)](#)
- [Горячие клавиши IntelliJ IDEA \(официальная шпаргалка\)](#)
- [Теория по Java](#)

```
# 2 - , - , Java, Object, equals/hashCode
```

```
---
```

```
##
```

```
- Java ( )
- - (Wrapper Classes)
-
-
- : `java.lang`, `java.util`, `java.io`
- `Object` -
- `equals()` `hashCode()`
-
- (overflow)
- `BigDecimal` - ( )
- `StringBuilder` -
-
```

```
---
```

```
##
```

```
Java 8 . ** **.
```

```
| | ( ) | ( ) | | |
|-----|-----|-----|-----|
| `byte` | 8 | 1 | -128 127 | `0` |
| `short` | 16 | 2 | -32768 32767 | `0` |
```

`int`	32	4	-2 ³¹ 2 ³¹ -1	`0`	
`long`	64	8	-2 ⁶³ 2 ⁶³ -1	`0L`	
`float`	32	4	±3.4e38 (7)	`0.0f`	
`double`	64	8	±1.7e308 (15)	`0.0d`	
`char`	16	2	`u0000` `uffff` (Unicode)	`u0000`	
`boolean`			`true` / `false`	`false`	

> `*` - `boolean` JVM - . `int` (32) .

- (Wrapper Classes)

:

		-	
-----	-----		
`int`		`Integer`	
`long`		`Long`	
`double`		`Double`	
`boolean`		`Boolean`	
`char`		`Character`	
...		...	

?

```
-      (`List<Integer>`,      `List<int>`).
```

```
-      `null`.
```

```
-      : `Integer.parseInt()`, `Character.isDigit()` ..
```

##

```
→ **      ** - .
```

```
→ **      ** - .
```

```
```java
Integer a = 10; // ← : int → Integer
int b = a; // ← : Integer → int
```

```
List<Integer> list = new ArrayList<>();
list.add(5); // ←
int first = list.get(0); // ←
```

☐ **Осторожно с null!**

```
\BuiltInTok{Integer}\NormalTok{ x }\OperatorTok{=} \KeywordTok{null}\OperatorTok{;}
\DataTypeTok{int}\NormalTok{ y }\OperatorTok{=} \NormalTok{ x }\OperatorTok{;} \CommentTok{// ← NullPointerException
```

→ Всегда проверяй на null перед распаковкой.

## 1.10 □ Передача параметров по значению

□ В Java всё передаётся по значению — даже объекты!

→ При передаче объекта — копируется **ссылка на объект**, а не сам объект.

### 1.10.1 Пример:

```
\KeywordTok{public} \DataTypeTok{static} \DataTypeTok{void} \FunctionTok{main}\OperatorTok{()}\BuiltInTok{String}
\NormalTok{ Person p }\OperatorTok{=} \KeywordTok{new} \FunctionTok{Person}\OperatorTok{()}\StringTok{"Alice"}
 \FunctionTok{changeName}\OperatorTok{()}\NormalTok{p}\OperatorTok{;}
 \BuiltInTok{System}\OperatorTok{.}\FunctionTok{out}\OperatorTok{.}\FunctionTok{println}\OperatorTok{()}\NormalTok{p}\OperatorTok{;}
\OperatorTok{\}}

\DataTypeTok{static} \DataTypeTok{void} \FunctionTok{changeName}\OperatorTok{()}\NormalTok{Person person}\OperatorTok{=}
\NormalTok{ person}\OperatorTok{.}\FunctionTok{name} \OperatorTok{=} \StringTok{"Bob"}\OperatorTok{;} \CommentTok{// ...}
\OperatorTok{\}}
```

→ Объект изменился, потому что мы **изменяли данные по скопированной ссылке**.

### 1.10.2 Но:

```
\DataTypeTok{static} \DataTypeTok{void} \FunctionTok{reassign}\OperatorTok{()}\NormalTok{Person person}\OperatorTok{=}
\NormalTok{ person }\OperatorTok{=} \KeywordTok{new} \FunctionTok{Person}\OperatorTok{()}\StringTok{"Charlie"}
\OperatorTok{\}}
```

→ После вызова `reassign(p)` — `p.name` всё ещё "Alice".

## 1.11 □ Пакеты Java

Пакеты — это пространства имён для классов.

### 1.11.1 Основные пакеты:

- `java.lang` — автоматически импортируется (`String`, `Object`, `System`, `Math`).
- `java.util` — коллекции, дата/время, `Scanner`, `Random`.
- `java.io` — ввод/вывод, файлы.
- `java.time` — современные даты (Java 8+).



### 1.11.2 Импорт:

```
\KeywordTok{import} \ImportTok{java}\OperatorTok{.}\ImportTok{util}\OperatorTok{.}\ImportTok{List}\OperatorTok{
\KeywordTok{import} \ImportTok{java}\OperatorTok{.}\ImportTok{util}\OperatorTok{.}\ImportTok{ArrayList}\OperatorTok{
\KeywordTok{import} \ImportTok{java}\OperatorTok{.}\ImportTok{time}\OperatorTok{.}\ImportTok{LocalDate}\OperatorTok{
\CommentTok{// :}
\KeywordTok{import} \ImportTok{java}\OperatorTok{.}\ImportTok{util}\OperatorTok{.*};
```

□ \* — не нагружает приложение, только упрощает написание кода.

### 1.12 □ Класс Object — корень иерархии

→ Корневой класс всей иерархии в Java.

→ Любой класс — наследник `Object` (явно или неявно).

### 1.12.1 Основные методы:

- `toString()` — строковое представление объекта.
- `equals(Object obj)` — сравнение объектов.
- `hashCode()` — хеш-код для использования в хеш-таблицах.
- `getClass()` — получить класс объекта.
- `clone()` — неглубокое клонирование (осторожно!).
- `finalize()` — освобождение ресурсов перед удалением, deprecated (Java 9+).

**1.13** 1.1.1 equals

```
\KeywordTok{public} \DataTypeTok{boolean} \FunctionTok{equals}\OperatorTok{()}\AttributeTok{@Nullable} \BuiltInTok{true}
```

□ **Назначение:** проверяет, равны ли два объекта *логически* (по содержимому), а не физически (по ссылке).

### 1.13.1 □ Поведение по умолчанию

→ В классе Object метод equals() сравнивает **ссылки**:

```
\BuiltInTok{Object}\NormalTok{ a }\OperatorTok{=} \KeywordTok{new} \BuiltInTok{Object}\OperatorTok{()};
\BuiltInTok{Object}\NormalTok{ b }\OperatorTok{=} \KeywordTok{new} \BuiltInTok{Object}\OperatorTok{()};
\BuiltInTok{System}\OperatorTok{.}\FunctionTok{out}\OperatorTok{.}\FunctionTok{println}\OperatorTok{()}\NormalTok{ }
```

→ Это эквивалентно  $a == b$ .



### 1.14.2 □ Реализация по умолчанию

→ В ранних версиях JVM hashCode() возвращал **адрес объекта в памяти**.

→ **Сейчас** — используется **псевдослучайное число**, которое: - Генерируется при первом вызове hashCode(). - Записывается в **заголовок объекта** (object header). - **Не меняется** в течение жизни объекта, даже если объект перемещается сборщиком мусора.

□ Почему изменили?

— При маленьком hear-e адреса были близки → хеши были не равномерны → **ухудшалась производительность хеш-таблиц.**

— Новая реализация даёт **лучшее распределение хешей** → меньше коллизий

→ быстрее работа HashMap.

### 1.14.3 □ Почему важно переопределять hashCode() вместе с equals()

→ Представь, что ты положил объект в `HashMap`, а потом изменил поле, участвующее в `equals()`, но не обновил `hashCode()`:

```
\NormalTok{Person p }\OperatorTok{=} \KeywordTok{new} \FunctionTok{Person}\OperatorTok{(}\StringTok{"Alice"}\OperatorTok{)}
\NormalTok{map}\OperatorTok{.}\FunctionTok{put}\OperatorTok{(}\NormalTok{p}\OperatorTok{,} \StringTok{"value"}\OperatorTok{)}

\NormalTok{p}\OperatorTok{.}\FunctionTok{setName}\OperatorTok{(}\StringTok{"Bob"}\OperatorTok{)}; \CommentTok{/}

\NormalTok{map}\OperatorTok{.}\FunctionTok{get}\OperatorTok{(}\NormalTok{p}\OperatorTok{)}; \CommentTok{// → null}
```

→ **Решение:** поля, используемые в `equals()` и `hashCode()`, должны быть **неизменяемыми (immutable)**.

### 1.15 □ Правильное переопределение

### 1.15.1 Способ 1: Вручную

```
\AttributeTok{@Override}
\KeywordTok{public} \DataTypeTok{boolean} \FunctionTok{equals}\OperatorTok{(}\BackslashBuiltInTok{Object}\NormalTok{ o}\Backslash
\ControlFlowTok{if} \OperatorTok{(}\BackslashKeywordTok{this} \OperatorTok{==}\NormalTok{ o}\OperatorTok{)} \Control
\ControlFlowTok{if} \OperatorTok{(}\BackslashNormalTok{o} \OperatorTok{==} \KeywordTok{null} \OperatorTok{||} \Func
\NormalTok{ Person person } \OperatorTok{=} \OperatorTok{(}\BackslashNormalTok{Person}\OperatorTok{)} \NormalTok{ o}\Op
\ControlFlowTok{return}\NormalTok{ Objects}\OperatorTok{.}\FunctionTok{equals}\OperatorTok{(}\BackslashNormalTok{name}
\NormalTok{ Objects}\OperatorTok{.}\FunctionTok{equals}\OperatorTok{(}\BackslashNormalTok{age}\OperatorTok{,}\Backslash
\OperatorTok{\}\}\}

\AttributeTok{@Override}
\KeywordTok{public} \DataTypeTok{int} \FunctionTok{hashCode}\OperatorTok{()} \OperatorTok{\}\}
\ControlFlowTok{return}\NormalTok{ Objects}\OperatorTok{.}\FunctionTok{hash}\OperatorTok{(}\BackslashNormalTok{name}
\OperatorTok{\}\}\}
```

### 1.15.2 Способ 2: Через Lombok (если используется)

```
\AttributeTok{@EqualsAndHashCode}
\KeywordTok{public} \KeywordTok{class}\NormalTok{ Person }\OperatorTok{\{}
 \KeywordTok{private} \BuiltInTok{String}\NormalTok{ name}\OperatorTok{;}
 \KeywordTok{private} \BuiltInTok{Integer}\NormalTok{ age}\OperatorTok{;}
\OperatorTok{\}}
```

### 1.15.3 Способ 3: В IntelliJ IDEA → Alt + Insert → equals() and hashCode()

→ IDE сгенерирует корректный код.

```
\AttributeTok{@Override}
\KeywordTok{public} \DataTypeTok{boolean} \FunctionTok{equals}\OperatorTok{(}\BackslashBuiltInTok{Object}\NormalTok{ o}\Backslash
 \ControlFlowTok{if} \OperatorTok{(}\BackslashKeywordTok{this} \OperatorTok{==}\NormalTok{ o}\OperatorTok{)} \Control
 \ControlFlowTok{if} \OperatorTok{(}\BackslashNormalTok{o} \OperatorTok{==} \KeywordTok{null} \OperatorTok{||} \Function
\NormalTok{ Person person }\OperatorTok{=} \OperatorTok{(}\BackslashNormalTok{Person}\OperatorTok{)}\NormalTok{ o}\OperatorTok{)} \Control
 \ControlFlowTok{return}\NormalTok{ Objects}\OperatorTok{.}\FunctionTok{equals}\OperatorTok{(}\BackslashNormalTok{name}
\NormalTok{ Objects}\OperatorTok{.}\FunctionTok{equals}\OperatorTok{(}\BackslashNormalTok{age}\OperatorTok{,}\Backslash
\OperatorTok{\}\}\}

\AttributeTok{@Override}
\KeywordTok{public} \DataTypeTok{int} \FunctionTok{hashCode}\OperatorTok{()}\OperatorTok{\}\}
 \ControlFlowTok{return}\NormalTok{ Objects}\OperatorTok{.}\FunctionTok{hash}\OperatorTok{(}\BackslashNormalTok{name}
\OperatorTok{\}\}\}
```

- `Objects.hash(...)` — безопасно обрабатывает `null` и генерирует хороший хеш на основе переданных полей.

## 1.16 □ Частые ошибки

- Сравнение через `==` для объектов → сравниваются ссылки, а не содержимое.
- Забыли `@Override` → можно случайно создать перегрузку, а не переопределение.
- Не переопределили `hashCode()` → проблемы с `HashMap`.
- Использовали изменяемые поля в `hashCode()` → объект “сломается” в `HashSet`, если поле изменится.
- Использовали `float/double` в `hashCode()` без округления → нестабильность из-за точности.

## 1.17 □ Советы

- Всегда переопределяй `equals()` и `hashCode()` **вместе**.
- Используй `java.util.Objects` — безопасно и читаемо.
- В IntelliJ IDEA: `Alt + Insert` → генерация методов — экономит время.
- Тестируй поведение в `HashMap` — это частый вопрос на собеседованиях.
- Поля в `hashCode()` и `equals()` — лучше делать `final`.

## 1.18 □ Контракт equals() и hashCode()

Если два объекта равны по equals() — их hashCode() **должен быть одинаковым**.

### 1.18.1 Почему это важно?

→ HashMap, HashSet, Hashtable используют hashCode() для определения “корзины”, а equals() — для точного сравнения.

### 1.18.2 Антипаттерн:

```
\AttributeTok{@Override}
\KeywordTok{public} \DataTypeTok{boolean} \FunctionTok{equals}\OperatorTok{()}\BuiltInTok{Object}\NormalTok{ o}\
\CommentTok{// ... }
\OperatorTok{\}}

\CommentTok{// hashCode()}
```

→ Объекты могут “потеряться” в HashMap.

---

## 1.19 □ Частые ошибки

- Сравнение через == для объектов → сравниваются ссылки, а не содержимое.
  - Забыли @Override → можно случайно создать перегрузку, а не переопределение.
  - Не переопределили hashCode() → проблемы с HashMap.
  - Использовали изменяемые поля в hashCode() → объект “сломается” в HashSet, если поле изменится.
- 

## 1.20 □ Переполнение (Overflow)

→ Происходит, когда результат операции **выходит за пределы диапазона типа**.

### 1.20.1 Пример с int:

```
\DataTypeTok{int}\NormalTok{ max }\OperatorTok{=}\BuiltInTok{Integer}\OperatorTok{.}\FunctionTok{MAX_VALUE}\C
\DataTypeTok{int}\NormalTok{ overflow }\OperatorTok{=}\NormalTok{ max }\OperatorTok{+}\DecValTok{1}\OperatorTok{
\BuiltInTok{System}\OperatorTok{.}\FunctionTok{out}\OperatorTok{.}\FunctionTok{println}\OperatorTok{()}\NormalTok{
```

→ Никакого исключения — просто “заворачивается” (как одометр в машине).

## 1.20.2 Как избежать?

- Используй long для больших чисел.
- Используй Math.addExact(), Math.multiplyExact() — бросают ArithmeticException при переполнении.

```
\ControlFlowTok{try} \OperatorTok{\{}
 \DataTypeTok{int}\NormalTok{ result }\OperatorTok{=} \BuiltInTok{Math}\OperatorTok{.}\FunctionTok{addExact}
\OperatorTok{\}} \ControlFlowTok{catch} \OperatorTok{\}\BuiltInTok{ArithmeticException}\NormalTok{ e}\OperatorTok{.}
 \BuiltInTok{System}\OperatorTok{.}\FunctionTok{out}\OperatorTok{.}\FunctionTok{println}\OperatorTok{\}\Stri
\OperatorTok{\}}
```

- Для критических вычислений — используй BigInteger.

---

## 1.21 BigDecimal — для точных вычислений (деньги!)


→ float и double **не подходят** для финансовых расчётов — из-за ошибок округления.

### 1.21.1 Пример проблемы:

```
\DataTypeTok{double}\NormalTok{ a }\OperatorTok{=} \FloatTok{0.1}\OperatorTok{;}
\DataTypeTok{double}\NormalTok{ b }\OperatorTok{=} \FloatTok{0.2}\OperatorTok{;}
\BuiltInTok{System}\OperatorTok{.}\FunctionTok{out}\OperatorTok{.}\FunctionTok{println}\OperatorTok{\}\NormalTok{ a + b }
```

### 1.21.2 Решение — BigDecimal:

```
\BuiltInTok{BigDecimal}\NormalTok{ a }\OperatorTok{=} \KeywordTok{new} \BuiltInTok{BigDecimal}\OperatorTok{\}\S
\BuiltInTok{BigDecimal}\NormalTok{ b }\OperatorTok{=} \KeywordTok{new} \BuiltInTok{BigDecimal}\OperatorTok{\}\S
\BuiltInTok{BigDecimal}\NormalTok{ sum }\OperatorTok{=} \NormalTok{ a }\OperatorTok{.}\FunctionTok{add}\OperatorTok{T
\BuiltInTok{System}\OperatorTok{.}\FunctionTok{out}\OperatorTok{.}\FunctionTok{println}\OperatorTok{\}\NormalTok{ a + b }
```

 **Всегда создавай BigDecimal из String, а не из double!**

```
\KeywordTok{new} \BuiltInTok{BigDecimal}\OperatorTok{\}\FloatTok{0.1}\OperatorTok{\}} \CommentTok{// ←
\KeywordTok{new} \BuiltInTok{BigDecimal}\OperatorTok{\}\StringTok{"0.1"}\OperatorTok{\}} \CommentTok{// ← }
```

### 1.21.3 Операции:

```
\NormalTok{a}\OperatorTok{.}\FunctionTok{add}\OperatorTok{\}\NormalTok{b}\OperatorTok{\}}
\NormalTok{a}\OperatorTok{.}\FunctionTok{subtract}\OperatorTok{\}\NormalTok{b}\OperatorTok{\}}
\NormalTok{a}\OperatorTok{.}\FunctionTok{multiply}\OperatorTok{\}\NormalTok{b}\OperatorTok{\}}
\NormalTok{a}\OperatorTok{.}\FunctionTok{divide}\OperatorTok{\}\NormalTok{b}\OperatorTok{\}, \DecValTok{2}\Opera
```

→ Используй BigDecimal для: - Денег - Процентов - Точных научных расчётов

## 1.22 StringBuilder — эффективная работа со строками

→ String в Java **неизменяем (immutable)** → каждая операция "a" + "b" создаёт новый объект.

### 1.22.1 Проблема:

```
\BuiltInTok{String}\NormalTok{ result }\OperatorTok{=}\StringTok{" "}\OperatorTok{;}
\ControlFlowTok{for}\OperatorTok{({}\DataTypeTok{int}\NormalTok{ i }\OperatorTok{=}\DecValTok{0}\OperatorTok{;}
\NormalTok{ result }\OperatorTok{+=}\StringTok{"a"}\OperatorTok{;} \CommentTok{// + 1000
\OperatorTok{\}}
```

→ Медленно и расходует память.


### 1.22.2 Решение — StringBuilder:

```
\BuiltInTok{StringBuilder}\NormalTok{ sb }\OperatorTok{=}\KeywordTok{new}\BuiltInTok{StringBuilder}\OperatorTok{
\ControlFlowTok{for}\OperatorTok{({}\DataTypeTok{int}\NormalTok{ i }\OperatorTok{=}\DecValTok{0}\OperatorTok{;}
\NormalTok{ sb }\OperatorTok{.}\FunctionTok{append}\OperatorTok{({}\StringTok{"a"}\OperatorTok{)}\; \CommentTok{
\OperatorTok{\}}
\BuiltInTok{String}\NormalTok{ result }\OperatorTok{=}\NormalTok{ sb }\OperatorTok{.}\FunctionTok{toString}\OperatorTok{}
```

→ В **100+ раз быстрее** для больших объёмов.

### 1.22.3 Основные методы:

```
\NormalTok{sb}\OperatorTok{.}\FunctionTok{append}\OperatorTok{({}\StringTok{"text"}\OperatorTok{)}
\NormalTok{sb}\OperatorTok{.}\FunctionTok{insert}\OperatorTok{({}\DecValTok{0}\OperatorTok{,}\StringTok{"prefix"}
\NormalTok{sb}\OperatorTok{.}\FunctionTok{delete}\OperatorTok{({}\DecValTok{0}\OperatorTok{,}\DecValTok{5}\OperatorTok{)}
\NormalTok{sb}\OperatorTok{.}\FunctionTok{reverse}\OperatorTok{()}
\NormalTok{sb}\OperatorTok{.}\FunctionTok{toString}\OperatorTok{() }
```

 Если нужна потокобезопасность — используй StringBuffer (но он медленнее из-за синхронизации).

---

## 1.23 Советы

- Всегда переопределяй equals() и hashCode() **вместе**.
  - Используй java.util.Objects — безопасно и читаемо.
  - В IntelliJ IDEA: Alt + Insert → генерация методов — экономит время.
  - Для денег — только BigDecimal.
  - Для конкатенации строк в цикле — только StringBuilder.
  - Проверяй переполнение в критических местах — используй Math.\*Exact().
-

## 1.24 □ Полезные ссылки

- [Oracle: Primitive Data Types](#)
- [Oracle: BigDecimal](#)
- [Oracle: StringBuilder](#)
- [Хабр: Контракт equals/hashCode](#)
- [Baeldung: Guide to hashCode\(\)](#)
- [Baeldung: BigDecimal](#)

““