



# Java EE levelup

# Немного о себе

- Студент 2 курса  
ПОИТ
- Java Backend  
Developer
- BrakhMen



# Телеграм

- @java\_leveup

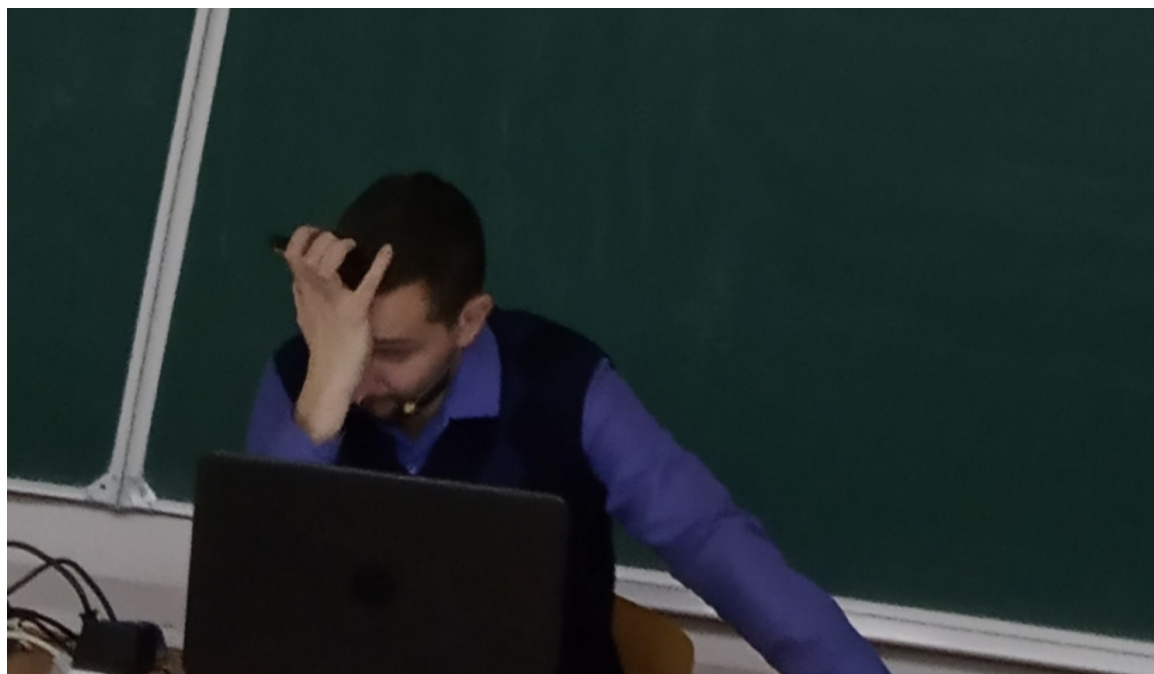


# ПРОГРАММА КУРСА

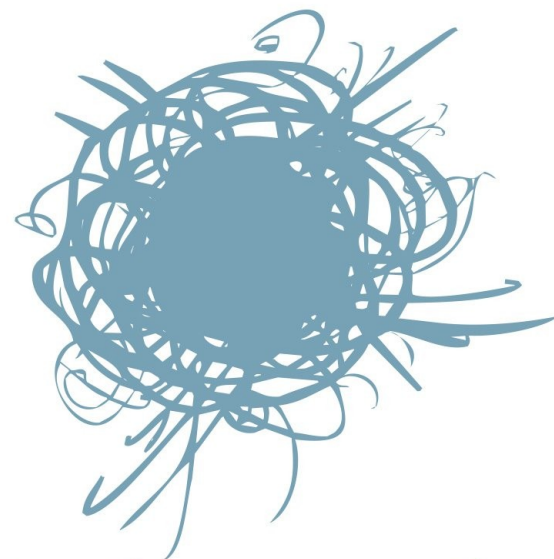
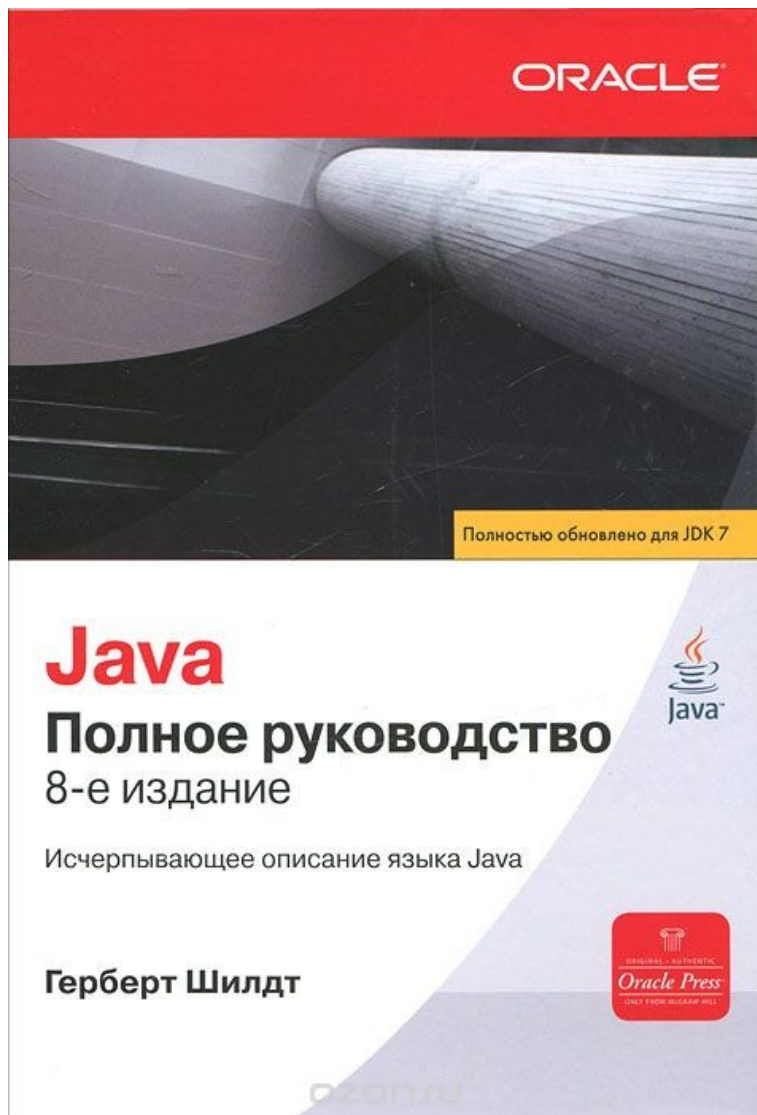


# МЫ НЕ:

- Будем изучать, как складывать числа на Java
- Будем писать Hello World
- Рассматривать синтаксис Java
- Тупо слушать теорию
- Изучать Delphi




# ЧТО ПОЧИТАТЬ



Хабрахабр





Google



# ООП

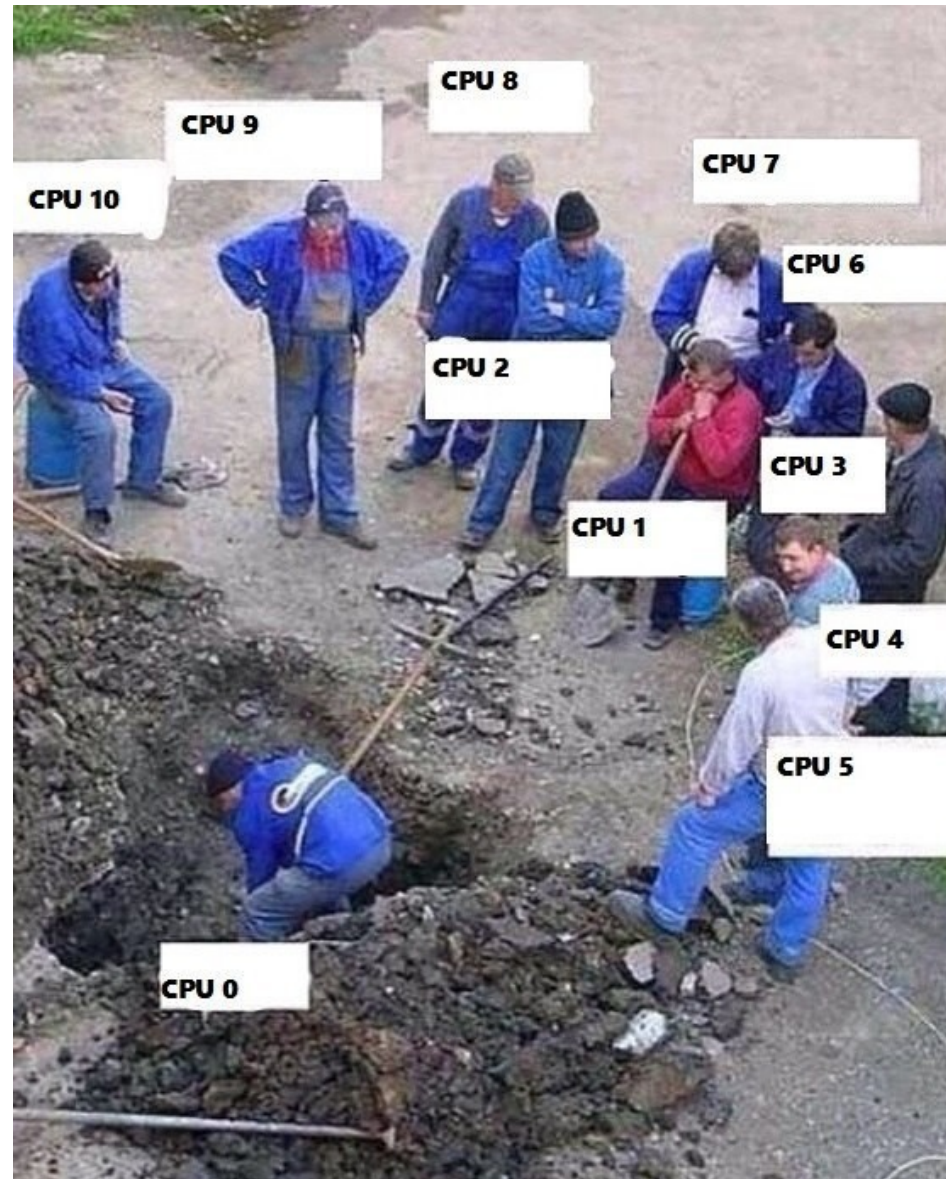




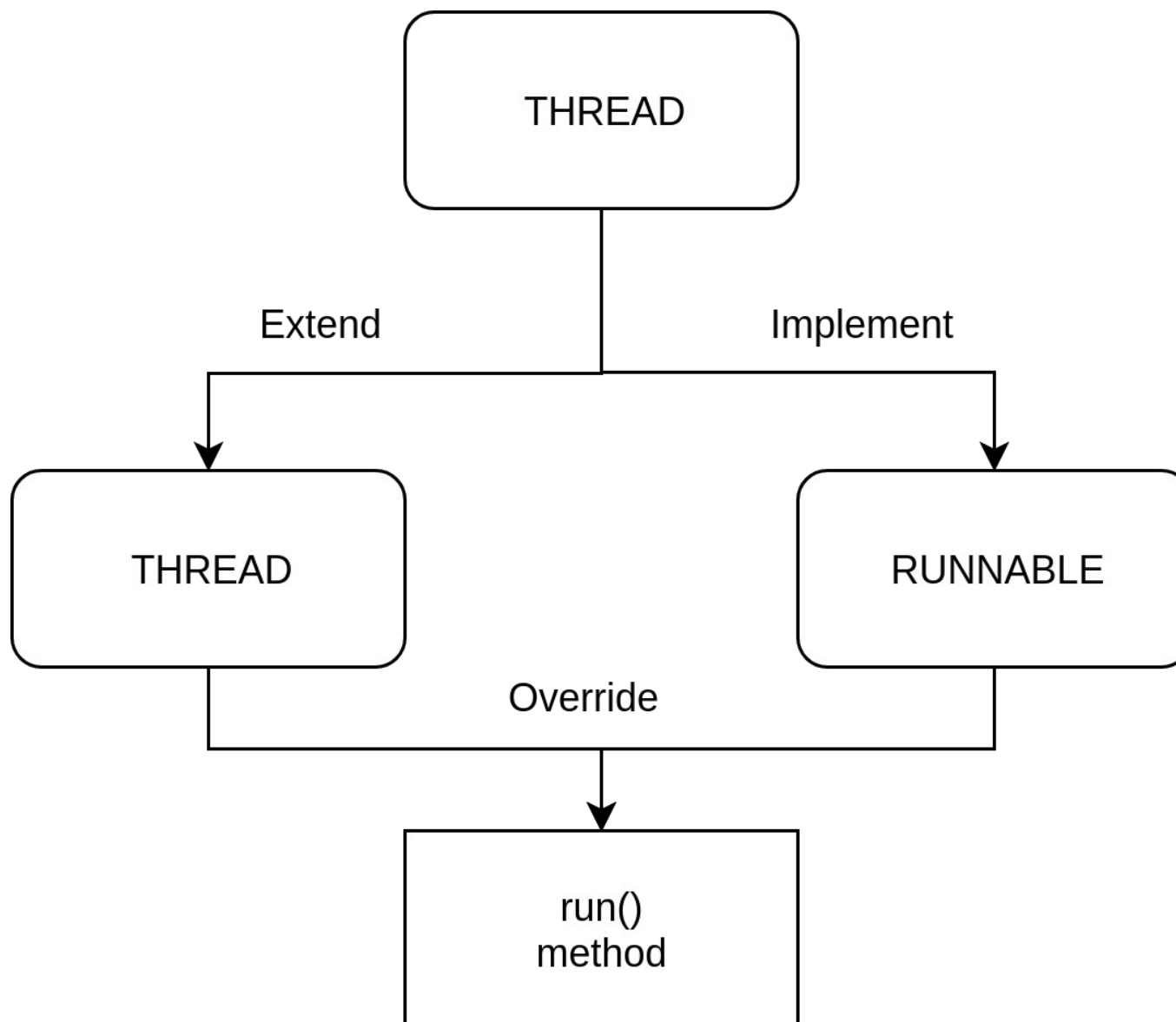
# Exceptions



# Многопоточность



# Создание потока





extends  
Thread



implements  
Runnable

```
public class PerfectThread implements Runnable {
    @Override
    public void run() {
        for(int i = 0; i < 10; i++) {
            System.out.println(i);
            try {
                Thread.sleep(100);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}

public class Main {

    public static void main(String[] args) throws InterruptedException {

        for(int i = 0; i < 10; i++) {
            Thread perfectThread = new Thread(new PerfectThread());
            perfectThread.start();
        }

        Thread.sleep(1000);
    }
}
```

```
public class PerfectThread extends Thread {
    @Override
    public void run() {
        for(int i = 0; i < 10; i++) {
            System.out.println(i);
            try {
                Thread.sleep(100);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}

public class Main {

    public static void main(String[] args) throws InterruptedException {

        for(int i = 0; i < 10; i++) {
            Thread perfectThread = new PerfectThread();
            perfectThread.start();
        }

        Thread.sleep(1000);
    }
}
```

```
0
0
0
0
0
0
0
0
0
0
0
1
1
1
1
```



# Race Condition



**MULTITHREADING**

THREADS ARE NOT GOING TO SYNCHRONIZE THEMSELVES

# Race Condition

```
volatile int x;
```

```
// Поток 1:  
while (!stop) {  
    x++;  
    ...  
}
```

```
// Поток 2:  
while (!stop) {  
    if (x%2 == 0)  
        System.out.println("x=" + x);  
    ...  
}
```

HOTFIX

```
// Поток 2:  
while (!stop)  
{  
    int cached_x = x;  
    if (cached_x%2 == 0)  
        System.out.println("x=" + cached_x);  
    ...  
}
```



# Синхронизация

```
public class Kek {  
    synchronized void doSomething() {  
        ...  
    }  
}
```

```
public class Kek {  
    void doSomething() {  
        synchronized(this) {  
            ...  
        }  
    }  
}
```

```
public class Main() {  
    public static void main(String[] args) {  
        Kek kek = new Kek();  
        synchronized(kek) {  
            kek.doIt()  
        }  
    }  
}
```

```
// Поток 1:  
while (!stop)  
{  
    synchronized(someObject)  
    {  
        x++;  
    }  
    ...  
}
```

```
// Поток 2:  
while (!stop)  
{  
    synchronized(someObject)  
    {  
        if (x%2 == 0)  
            System.out.println("x=" + x);  
    }  
    ...  
}
```



# Как тестировать Race Condition?

- Sleep
- Breakpoints

# Досинхронизировались? или Dead Lock



## DEADLOCK

Game over, man, game over.

```

public class Deadlock {
    static class Friend {
        private final String name;
        public Friend(String name) {
            this.name = name;
        }
        public String getName() {
            return this.name;
        }
        public synchronized void bow(Friend bower) {
            System.out.format("%s: %s" + " has bowed to me!\n", this.name, bower.getName());
            bower.bowBack(this);
        }
        public synchronized void bowBack(Friend bower) {
            System.out.format("%s: %s"
                + " has bowed back to me!\n",
                this.name, bower.getName());
        }
    }

    public static void main(String[] args) {
        final Friend alphonse = new Friend("Alphonse");
        final Friend gaston = new Friend("Gaston");
        new Thread(new Runnable() {
            @Override
            public void run() {
                System.out.println("Thread 1");
                alphonse.bow(gaston);
                System.out.println("Th: gaston bowed to alphonse");
            }
        }).start();

        new Thread(new Runnable() {
            @Override
            public void run() {
                System.out.println("Thread 2");
                gaston.bow(alphonse);
                System.out.println("2.gaston waiting alph bowed");
            }
        }).start();
    }
}

```

A: alphonse.bow(gaston) —  
 получает лок alphonse  
 G: gaston.bow(alphonse) —  
 получает лок gaston  
 G: пытается вызвать  
 alphonse.bowBack(gaston), но  
 блокируется, ожидая лока  
 alphonse  
 A: пытается вызвать  
 gaston.bowBack(alphonse), но  
 блокируется, ожидая лока  
 gaston



# Потоки (Streams)







# Читаем файл

```
public class StreamsTest {
    public static void main(String[] args) throws IOException {
        if(args.length < 2) {
            return;
        }
        String filename = args[1];

        // Способ 1
        try (BufferedReader reader = new BufferedReader(
            new InputStreamReader(
                new FileInputStream(filename), StandardCharsets.UTF_8))) {
            String line;
            while ((line = reader.readLine()) != null) {
                System.out.println(line);
            }
        }

        // Способ 2
        try (FileReader reader = new FileReader(filename)) {
            int c;
            while ((c = reader.read()) != -1) { // читаем посимвольно
                System.out.print((char) c);
            }
        }

        // Способ 3
        List<String> lines = Files.readAllLines(Paths.get(filename), StandardCharsets.UTF_8);
        for (String line: lines) {
            System.out.println(line);
        }

        // Способ 4 (Java 8+)
        Files.lines(Paths.get(filename), StandardCharsets.UTF_8).forEach(System.out::println);
    }
}
```

# Домашнее задание

- Условие:
  - Написать программу, в которой будет два потока: первый читает вводимую в консоль информацию и обрабатывает команды, второй поток каждую секунду читает файл и проверяет, есть ли для пользователя новые сообщения. Если есть — выводит их в консоль. В сообщении должно содержаться имя отправителя.

Команды:

- `/reg USERNAME` — регистрация (без этой команды остальной функционал недоступен)
- `/exit` — выход из приложения
- Просто текст — обычное сообщение
- `/pause` и `/resume` [ДОПОЛНИТЕЛЬНО] — приостановить и возобновить чтение файла

# Домашнее задание

- Пример работы:

» /reg Alex

Привет, Alex!

» Джава — круто

[Alex]: Джава — круто

[Bob]: Я пишу из файла!

» /exit

Пока, Alex. Приходи еще

*``Process finished with exit code 0``*

# Домашнее задание

- Требования:
  - git
  - maven/gradle
  - Читабельный код