**Леонид Бадеев**

Администратор

Thread class и Runnable interface: Основные инструменты для создания и управления потоками в Java. Вы можете создать новый поток, реализовав интерфейс Runnable или расширив класс Thread. Synchronization: Синхронизация необходима для предотвращения одновременного доступа к общим данным из разных потоков, что может привести к некорректным результатам или неожиданному поведению. В Java для синхронизации можно использовать ключевое слово synchronized или объекты блокировки из пакета java.util.concurrent.locks. Thread Pool: Thread Pool - это набор предварительно созданных потоков, которые могут быть использованы для выполнения задач. Использование пула потоков может существенно улучшить производительность по сравнению с созданием нового потока для каждой задачи. В Java есть несколько реализаций пулов потоков, таких как ThreadPoolExecutor и ForkJoinPool. Futures and Callable: Future представляет результат вычислений, которые могут быть выполнены в параллельном потоке. Callable - это задача, которую можно выполнить в другом потоке и которая возвращает результат. Atomic Variables: Классы в пакете java.util.concurrent.atomic, такие как AtomicInteger, обеспечивают операции безопасные для потоков на одной переменной без использования блокировок. Concurrent Collections: Java предоставляет ряд потокобезопасных коллекций в пакете java.util.concurrent, таких как ConcurrentHashMap и CopyOnWriteArrayList, которые были специально разработаны для использования в многопоточных программах.

22:02

ЕБ

**Егор Бойко**

супер, спасибо!

22:04

КД

**Королев Дмитрий**

Спасибо!

22:04

public class Calculator { public double divide(double a, double b) { if (b == 0) throw new ArithmeticException("Division by zero"); return a / b; } public double multiply(double a, double b) { if(a > Double.MAX\_VALUE / b) throw new ArithmeticException("Double type overflow"); return a \* b; } public double subtract(double a, double b) { if (a < b) throw new ArithmeticException("Первое число меньше второго!"); return a - b; } public static void main(String[] args) { Calculator calculator = new Calculator(); try { System.out.println(calculator.divide(10, 0)); } catch (Exception e) { System.out.println("Вышла ошибка: " + e.getMessage()); } try { System.out.println(calculator.multiply(10, 0)); } catch (Exception e) { System.out.println("Вышла ошибка: " + e.getMessage()); } try { System.out.println(calculator.subtract(0, 10)); } catch (Exception e) { System.out.println("Вышла ошибка: " + e.getMessage()); } } }

**Илья Плутов**

public class Calculator { public double divide(double a, double b) { if (b == 0) throw new ArithmeticException("Division by zero"); return a / b; } public double multiply(double a, double b) { if(a > Double.MAX\_VALUE / b) throw new ArithmeticException("Double type overflow"); return a \* b; } public double subtract(double a, double b) { if (a < b) throw new ArithmeticException("Первое число меньше второго!"); return a - b; } public static void main(String[] args) { Calculator calculator = new Calculator(); try { System.out.println(calculator.divide(10, 0)); } catch (Exception e) { System.out.println("Вышла ошибка: " + e.getMessage()); } try { System.out.println(calculator.multiply(10, 0)); } catch (Exception e) { System.out.println("Вышла ошибка: " + e.getMessage()); } try { System.out.println(calculator.subtract(0, 10)); } catch (Exception e) { System.out.println("Вышла ошибка: " + e.getMessage()); } } }

21:53

ЛБ

**Леонид Бадеев**

Администра

Добрый вечер всем!

20:59

ЛБ

**Леонид Бадеев**

Администратор

+

21:00

КД

**Королев Дмитрий**

+

21:00

СС

**Сергей Старостин**

+

21:00

АВ

**Александр Восьмерик**

+

21:00

ИП

**Илья Плутов**

+

21:00

НБ

**Никита Банников**

+

21:00

ИС