План урока

1. Концепция асинхронного программирования
2. Разница между асинхронным, последовательным (синхронным) и многопоточным программированием
3. Реализация асинхронного программирования на Java. Интерфейс Callable, Future
4. Пример создания асинхронной задачи
5. Пример приложения с использованием асинхронности

Реализация асинхронного программирования в Java. Интерфейс Callable, Future

```
@FunctionalInterface
public interface Callable<V> {

    V call() throws Exception;
}
```

Асинхронное программирование в Java

Реализация асинхронного программирования в Java. Интерфейс Callable, Future

```
public interface Future<V> {

    boolean cancel(boolean mayInterruptIfRunning);

    boolean isCancelled();

    boolean isDone();

    V get() throws InterruptedException, ExecutionException;

    V get(long timeout, TimeUnit unit)
        throws InterruptedException, ExecutionException, TimeoutException;
}
```

Асинхронное программирование в Java

Пример создания асинхронной задачи

```
System.out.println("Main начал работу");
ExecutorService executorService = Executors.newFixedThreadPool(3);

Callable<String> callable = () -> {
    System.out.printf("%s начал работу\n", Thread.currentThread().getName());
    Thread.sleep(1000);
    return Thread.currentThread().getName();
};

List<Future<String>> futures = new ArrayList<>();

for (int i = 0; i < 3; i++) {
    futures.add(executorService.submit(callable));
}

System.out.println("Main продолжает работу");

for (Future<String> future : futures) {
    System.err.printf("Результат из потока %s\n", future.get());
}

executorService.shutdown();
```

Future .isDone() – возвращает true если задача уже выполнилась.

Future .cancel() – завершить выполнение в Future, отменить выполнение.