Задача № 1 по лабе № 6

1. Что такое поток (Thread) и чем отличается от процесса (Process)?

Процесс — программа, выполняющаяся в текущий момент времени, и все её элементы: адресное пространство, глобальные переменные, регистры, потоки.

Поток — это единица реализации программного кода. Последовательность данных, которая могут работать параллельно с другими своими «аналогами».

Поток отвечает за выполнение инструкций запущенного процесса, к которому он относится. Все это происходит параллельно с иными потоками этого же process.

1. Что такое упреждающее планирование (preemptive scheduling), разделение времени (time slicing), Совместное планирование (cooperative scheduling)?

Упреждающее планирование — планирование, когда процесс переходит из состояния выполнения в состояние готовности или из состояния ожидания в состояние готовности.

Разделение времени — способ планирования, где происходит деление всего времени работы процессора по частям (time slices), в каждую из которых указывается конкретная задача.

Совместное планирование — алгоритм, в котором поток освобождает ЦПУ только на добровольной основе (блокируясь, останавливаясь или уступая). Даже если поток с более высоким приоритетом становится готовым к выполнению, ОС никогда не будет вытеснять текущий поток, и она будет перепланировать его снова в случае внешнего прерывания. Эта форма планирования возлагает всю ответственность на программиста, который должен тщательно проектировать потоки.

1. Приведите примеры, где многопоточность используется или может быть использована?

Многопоточность в Java — это выполнение двух или более потоков одновременно для максимального использования центрального процесса.

Реальные примеры: разработка интерфейсов (как пример, интерфейс загрузки и скачивания файла), разработка игр (потоки анимации, работы с сетью и расчёта физики в основном), социальных сетей.

1. Каковы различные этапы жизненного цикла потока в Java? Какие методы можно использовать для изменения состояния потока?

Стадии жизненного цикла потока:

* [NEW](https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/lang/Thread.State.html#NEW)  
  В этом состоянии находится еще не запущенный поток.
* [RUNNABLE](https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/lang/Thread.State.html#RUNNABLE)  
  В этом состоянии находится поток, выполняющийся на виртуальной машине Java.
* [BLOCKED](https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/lang/Thread.State.html#BLOCKED)  
  Поток, заблокированный в ожидании блокировки монитора, находится в этом состоянии.
* [WAITING](https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/lang/Thread.State.html#WAITING)  
  В этом состоянии находится поток, бесконечно ожидающий, пока другой поток выполнит определенное действие.
* [TIMED\_WAITING](https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/lang/Thread.State.html#TIMED_WAITING)  
  Поток, ожидающий выполнения действия другим потоком до указанного времени ожидания, находится в этом состоянии.
* [TERMINATED](https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/lang/Thread.State.html#TERMINATED)  
  Вышедший поток находится в этом состоянии.

start() (запуск потока)

sleep() (смена в спящее состояние)

wait() (усыпляет поток до вызова метода notify)

notify() (пробуждает поток)

notifyall() (пробуждает все потоки)

yield() (приостановка текущего потока и запуск других).

1. Что такое демон-поток в Java?

Демон-потоки — это фоновые потоки (служебные), выполняющие какую-то работу в фоне.

Как сказано в документации, JVM продолжает выполнение программы (процесса), до тех пор, пока:

* Не вызван метод [Runtime.exit](https://docs.oracle.com/javase/8/docs/api/java/lang/Runtime.html" \l "exit-int-" \t "_blank)
* Все НЕ демон-потоки завершили свою работу (как без ошибок, так и с выбрасыванием исключений)

Отсюда и важная деталь: демон-потоки могут быть завершены на любой выполняемой команде. Поэтому целостность данных в них не гарантируется. Поэтому, демон потоки подходят для каких-то служебных задач. Например, в Java есть поток, который отвечает за обработку методов finalize или потоки, относящиеся к сборщику мусора (Garbage Collector, GC).

1. Что такое пул потоков (Thread Pool) в Java?

Пул потоков повторно использует ранее созданные потоки для выполнения текущих задач и предлагает решение проблемы накладных расходов цикла потока и перегрузки ресурсов. Поскольку на момент поступления запроса поток уже существует, задержка, связанная с созданием потока, устраняется, что делает приложение более отзывчивым.

Java предоставляет структуру Executor, которая сосредоточена вокруг интерфейса Executor, его подинтерфейса –ExecutorService и класса-ThreadPoolExecutor, который реализует оба этих интерфейса. При использовании исполнителя нужно только реализовать объекты Runnable и отправить их исполнителю для выполнения.

Чтобы использовать пулы потоков, мы сначала создаем объект ExecutorService и передаем ему набор задач. Класс ThreadPoolExecutor позволяет установить размер ядра и максимальный размер пула. Runnables, которые запускаются конкретным потоком, выполняются последовательно.

1. Что такое группа потоков (Thread Group)?

ThreadGroup создает группу потоков. Он предлагает удобный способ управления группами потоков как единым целым. Это особенно ценно в ситуации, когда вы хотите приостановить и возобновить несколько связанных потоков.

Группа потоков образует дерево, в котором каждая группа потоков, кроме исходной группы потоков, имеет родителя.

Потоку разрешен доступ к информации о своей собственной группе потоков, но не доступ к информации о родительской группе потоков его группы потоков или любой другой группе потоков.

1. Что такое реентерабельный метод (reentrant method)?

Это метод который может быть совместно использована несколькими пользователями или процессами. Для обеспечения реентерабельности необходимо выполнение нескольких условий:

• никакая часть вызываемого кода не должна модифицироваться;

• вызываемая процедура не должна сохранять информацию между вызовами;

• если процедура изменяет какие-либо данные, то они должны быть уникальными для каждого пользователя;

• процедура не должна возвращать указатели на объекты, общие для разных пользователей.

1. Для чего используется ключевое слово synchronized?

Им помечается определенный кусок нашего кода. Если блок кода помечен ключевым словом synchronized, это значит, что блок может выполняться только одним потоком одновременно.

Смысл прост. Если один поток зашел внутрь блока кода, который помечен словом synchronized, он моментально захватывает мьютекс объекта (механизм предоставления доступа к объекту в определенное время только одному потоку), и все другие потоки, которые попытаются зайти в этот же блок или метод вынуждены ждать, пока предыдущий поток не завершит свою работу и не освободит монитор.

1. Для чего используются методы wait(), notify() и notifyAll() в классе Object?

Wait():

Методы ожидания объекта имеют три варианта, один из которых бесконечно ожидает, пока любой другой поток не вызовет метод notify или notifyAll для объекта, чтобы разбудить текущий поток. Другие две вариации помещают текущий поток в ожидание на определенное время, прежде чем он проснется.

Notify():

метод notify пробуждает только один поток, ожидающий объекта, и этот поток начинает выполнение. Поэтому, если есть несколько потоков, ожидающих объекта, этот метод разбудит только один из них. Выбор потока для пробуждения зависит от реализации управления потоками в ОС.

NotifyAll():

Метод notifyAll пробуждает все потоки, ожидающие объекта, хотя какой из них будет обрабатываться первым, зависит от реализации ОС. Эти методы можно использовать для реализации проблемы производителя-потребителя, когда потоки-потребители ожидают объекты в очереди, а потоки-производители помещают объект в очередь и уведомляют ожидающие потоки.­­