**Вопросы к лабораторной работе №15:**

1. Что такое процесс, домен, поток? Как они связаны между собой?

* **Процесс** - это экземпляр программы, который выполняется в операционной системе.
* **Домен** - это изолированная и управляемая среда выполнения внутри процесса
* **Поток (Thread)** - последовательность выполняемых команд процессора.

1. Как получить информацию о процессах?

В .NET процесс представлен классом Process из пространства имен System.Diagnostics.

1. Как создать и настроить домен?

Домен создаётся использованием статического метода **AppDomain.CreateDomain()**.

Загрузить сборку в домен можно с помощью метода **AppDomain.Load()**, с указанием имени сборки. Кроме того, существует метод **AppDomain.ExecuteAssembly()**, который позволяет загрузить сборку \*.ехе и выполнить метод Main().

С помощью метода **AppDomain.Unload()** можно производить избирательную выгрузку определенного домена приложения из обслуживающего процесса. В этом случае вместе с доменом приложения будут выгружаться и все содержащиеся в нем сборки.

1. Как создать и настроить поток?

использовать класс Thread из пространства имен System.Threading. Чтобы запустить поток, вызывается метод **Start**. Метод **Sleep** позволяет приостановить поток на промежуток времени, указанный как параметр данного метода. Поток можно сделать фоновым, изменив значение свойства **IsBackground**, изменить его приоритет **Priority**, изменить его имя **Name**.

1. В каких состояниях может быть поток?

* Выполнение;
* Ожидание - ожидания конца ввода-вывода;
* Готовность - готов работать, но компьютер занят другим.

1. Какие методы управления потоками вы знаете, для чего и как их использовать?

Thread.Sleep: Этот метод приостанавливает выполнение текущего потока на указанное количество миллисекунд. Он может использоваться, например, для создания задержки между выполнением определенных операций или для снижения нагрузки на процессор.

Thread.Join: Этот метод ожидает завершения выполнения указанного потока. Он может использоваться для синхронизации выполнения нескольких потоков, чтобы гарантировать, что определенные операции будут выполнены только после завершения других потоков.

Thread.Start: Этот метод запускает выполнение потока. Он вызывает метод, указанный при создании объекта Thread, и начинает выполнение кода в этом методе.

Thread.Abort: Этот метод прерывает выполнение потока. Он может использоваться для принудительного завершения потока, но рекомендуется использовать другие механизмы для контроля выполнения потока, так как Abort может привести к непредсказуемым состояниям и проблемам синхронизации.

Thread.IsAlive: Это свойство, которое указывает, выполняется ли поток в данный момент. Оно может быть использовано для проверки статуса выполнения потока и принятия соответствующих решений.

Thread.Priority: Это свойство, которое позволяет установить или получить приоритет выполнения потока. Приоритеты могут быть установлены на уровне ниже, нормального или выше, чтобы определить относительный приоритет выполнения потоков.

1. Какие приоритеты потока вы знаете?

Перечисление ThreadPriority:

* Lowest;
* BelowNormal;
* Normal (по умолч.);
* AboveNormal;
* Highest;
* Real time.

1. Что такое пул потоков и для чего он используется?

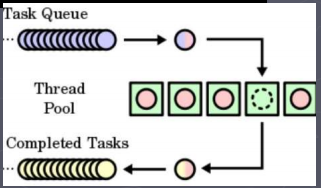
Вместо создания нового потока для каждой задачи, пул потоков предоставляет набор готовых к использованию потоков, которые могут выполнять задачи в фоновом режиме.

1. Что такое критическая секция? Поясните использование.

Критическая секция - это часть кода, которая обеспечивает многопоточную безопасность и предотвращает одновременный доступ нескольких потоков к общему ресурсу или участку кода.

10.Что такое мьютекс? Поясните использование

Mutex это средство синхронизации, используемое для контроля доступа к общим ресурсам в многопоточной среде.

**WaitOne()** – вход в критическую секцию

**ReleaseMutex()** – выход из нее (выход может быть произведен только в том же потоке выполнения, что и вход)

11.Что такое семафор? Поясните использование

Еще один инструмент, для управления синхронизацией, позволяющий войти в заданный участок кода не более чем N потокам (N – емкость семафора).

\* класс System.Threading.Semaphore – между процессами

\* класс SemaphoreSlim – в рамках одного процесса

\* Wait() – получ. блокировки

\* Release() – снятие блокировки.

12.Что такое неблокирующие средства синхронизации?

Неблокирующие средства синхронизации - это механизмы, используемые в многопоточных или распределенных системах для обеспечения согласованности и безопасности данных без блокировки или ожидания других потоков.

13.Для чего можно использовать класс Timer?

Данный класс позволяет запускать определенные действия по истечению некоторого периода времени.

Одна из перегрузок конструктора таймера принимает четыре параметра:

* объект делегата TimerCallback;
* объект, передаваемый в качестве параметра в метод делегата;
* количество миллисекунд, через которое таймер будет запускаться;
* интервал между вызовами метода делегата.

// устанавливаем метод обратного вызова

        TimerCallback tm = new TimerCallback(Count);

// создаем таймер

        Timer timer = new Timer(tm, num, 0, 2000);