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

**Процесс** – это экземпляр программы, выполняющийся на компьютере. У каждого процесса есть своё виртуальное адресное пространство, ресурсы и минимум один поток.

**Домен приложения (AppDomain)** – это изолированная область внутри процесса, в которой выполняются приложения .NET. Домен предоставляет изоляцию, безопасность и управление памятью.

**Поток (Thread)** – используемый внутри процесса путь выполнения.

**Связь:** Один процесс может содержать несколько доменов, а каждый домен может иметь несколько потоков.

**2. Как получить информацию о процессах?**

Для получения информации о процессах используется класс System.Diagnostics.Process.

**Пример:**

using System.Diagnostics;

foreach (var process in Process.GetProcesses())

{

Console.WriteLine($"ID: {process.Id}, Name: {process.ProcessName}");

}

**3. Как создать и настроить домен?**

AppDomain newDomain = AppDomain.CreateDomain("NewDomain");

Console.WriteLine(newDomain.FriendlyName);

AppDomain.Unload(newDomain);

**4. Как создать и настроить поток?**

using System.Threading;

Thread thread = new Thread(() =>

{

Console.WriteLine("Thread running");

});

thread.Start();

**5. В каких состояниях может быть поток?**

* **Unstarted** – поток создан, но не запущен.
* **Running** – поток выполняется.
* **WaitSleepJoin** – поток находится в состоянии ожидания.
* **Suspended** – поток приостановлен.
* **Stopped** – поток завершён.
* **Aborted** – поток прерван.

**6. Методы управления потоками**

* Start() – запуск потока.
* Join() – ожидание завершения потока.
* Abort() – завершение потока (устарел).
* Sleep() – приостановка на заданное время.
* Suspend() и Resume() – приостановка и возобновление потока (устарели).

**7. Приоритеты потока**

* Highest
* AboveNormal
* Normal (по умолчанию)
* BelowNormal
* Lowest

thread.Priority = ThreadPriority.AboveNormal;

**8. Что такое пул потоков и для чего он используется?**

**Пул потоков** – это группа заранее созданных потоков, которые используются для выполнения задач без необходимости создавать новый поток каждый раз.

**Пример:**

ThreadPool.QueueUserWorkItem(\_ =>

{

Console.WriteLine("Task in thread pool");

});

**9. Что такое критическая секция?**

**Критическая секция** – это механизм синхронизации, который позволяет только одному потоку выполнять код внутри секции.

**Пример:**

object lockObject = new object();

lock (lockObject)

{

// Критическая секция

Console.WriteLine("In critical section");

}

**10. Что такое мьютекс?**

**Мьютекс** – это примитив синхронизации, который может использоваться между несколькими процессами.

**Пример:**

using (Mutex mutex = new Mutex(false, "GlobalMutex"))

{

if (mutex.WaitOne(TimeSpan.FromSeconds(5), false))

{

Console.WriteLine("Mutex acquired");

mutex.ReleaseMutex();

}

else

{

Console.WriteLine("Failed to acquire mutex");

}

}

**11. Что такое семафор?**

**Семафор** – это примитив синхронизации, который ограничивает количество потоков, одновременно выполняющих определённый блок кода.

**Пример:**

Semaphore semaphore = new Semaphore(2, 2);

semaphore.WaitOne();

Console.WriteLine("Semaphore acquired");

semaphore.Release();

**12. Неблокирующие средства синхронизации**

**Неблокирующие средства** позволяют избежать блокировок при доступе к общим данным.

* SpinLock
* ConcurrentDictionary
* ReaderWriterLockSlim

**13. Класс Timer**

**Timer** используется для выполнения кода через заданные интервалы времени.

**Пример:**

using System.Threading;

Timer timer = new Timer(\_ =>

{

Console.WriteLine("Timer executed");

}, null, 0, 1000);

Thread.Sleep(5000);