Практические задания по модуль 4. Синхронизация в Мире Фэнтези

Цель модуля: Научиться безопасно управлять общими ресурсами в многопоточных условиях, предотвращая состояние гонки, взаимные блокировки и порчу данных.

Тема 1: Race Condition (Состояние гонки) и lock

Объяснение: Несколько героев пытаются одновременно забрать предмет из сундука. Без синхронизации они могут "увидеть" предмет одновременно, и он будет забран несколько раз. lock создает "магический барьер", через который в один момент времени может пройти только один поток.

Задания на выбор:

1. **Простое: "Сундук с золотом"**. Создайте класс Chest с целочисленным полем Gold (например, 100). Запустите 3 потока ("Героя"), которые будут 10 раз забирать из сундука по 10 золотых. Без использования lock вы увидите, что итоговое количество золота будет отрицательным или некорректным. Добавьте lock для исправления ситуации.
2. **Среднее: "Зелье лечения"**. Создайте класс HealingPotion с свойством Charges (заряды, например, 5). Несколько потоков-Player пытаются использовать зелье (Use()), уменьшая количество зарядов. Реализуйте механизм, при котором заряд тратится только если он есть, и два игрока не могут использовать последний заряд одновременно.
3. **Сложное: "Магический артефакт"**. Создайте класс Artifact с методом Activate(string playerName). Метод должен выводить в консоль [playerName] активировал артефакт! и увеличивать счетчик активаций. Запустите 5 потоков-Wizard, которые в цикле вызывают Activate. Без lock сообщения будут накладываться друг на друга, а счетчик будет неверным. Исправьте это.

Тема 2: Deadlock (Взаимная блокировка) и его предотвращение

Объяснение: Два мага одновременно пытаются обменяться магическими компонентами: первый держит "Корень мандрагоры" и хочет получить "Перо феникса", а второй наоборот. Каждый ждет, когда другой освободит нужный ресурс, и оба замирают навечно.

Задания на выбор:

1. **Простое: "Ссора двух гномов"**. Создайте два объекта-lock (anvil и forge). Создайте два потока. Первый поток (DwarfBlacksmith) захватывает anvil, а потом пытается захватить forge. Второй поток (DwarfEngineer) захватывает forge, а потом пытается захватить anvil. Продемонстрируйте взаимную блокировку. Для решения измените порядок захвата в одном из потоков, чтобы он был одинаковым.
2. **Среднее: "Торговля в таверне"**. Создайте два класса: Trader (торговец) с методом Trade(Trader other, int amount) и Tavern (таверна) с методом MakeDeal(Trader a, Trader b, int amount). Внутри MakeDeal необходимо заблокировать обоих торговцев. Реализуйте обмен золотом между двумя торговцами, избежав дедлока с помощью Monitor.TryEnter с таймаутом.
3. **Сложное: "Зачарование двух свитков"**. У вас есть два магических свитка (ScrollA и ScrollB). Два потока-Enchanter должны наложить заклинание, требующее одновременного доступа к ОБОИМ свиткам, но каждый начинает с разного свитка. Смоделируйте дедлок. Исправьте его, используя стратегию упорядочивания ресурсов (всегда сначала блокировать ScrollA, потом ScrollB).

Тема 3: Data Corruption (Порча данных) и ReaderWriterLockSlim

Объяснение: Карта сокровищ (shared resource) может читаться множеством искателей приключений (readers), но обновляться (write) только одним картографом за раз. ReaderWriterLockSlim позволяет множеству читателей работать одновременно, но дает эксклюзивный доступ писателю.

Задания на выбор:

1. **Простое: "Библиотека заклинаний"**. Создайте класс SpellBook (список строк с названиями заклинаний). Несколько потоков-Apprentice (учеников) постоянно читают и выводят случайное заклинание из книги. Один поток-Archmage (архимаг) периодически добавляет в книгу новое заклинание. Реализуйте это с помощью ReaderWriterLockSlim, чтобы чтение не блокировало других читателей.

2. **Среднее: "Доска заданий в гильдии"**. Создайте класс QuestBoard с списком доступных заданий. Многие потоки-Adventurer могут одновременно просматривать доску (ReadQuests). Один поток-GuildMaster может периодически добавлять новое задание или снимать старое (AddQuest/RemoveQuest). Используйте ReaderWriterLockSlim для обеспечения потокобезопасности.
3. **Сложное: "Журнал монстров"**. Создайте класс Bestiary с словарем Dictionary<string, int>, где ключ - название монстра, значение - количество убитых. Множество потоков-Hunter увеличивают счетчик для определенного монстра (это операция **чтения + изменения + записи**, поэтому требует эксклюзивной блокировки!). Один поток-Scholar раз в несколько секунд выводит всю статистику (чтение). Реализуйте с ReaderWriterLockSlim, не забывая, что запись должна быть эксклюзивной.

Тема 4: SemaphoreSlim / ManualResetEventSlim

Объяснение: SemaphoreSlim - это "магический портал", который может пропустить только N существ одновременно. ManualResetEventSlim - это "сигнальная ракета", которая сообщает всем ожидающим потокам, что событие произошло.

Задания на выбор:

1. **Простое: "Ограниченный мост"**. Создайте SemaphoreSlim(3), имитирующий мост, который выдерживает только 3 героев одновременно. Запустите 10 потоков-Hero, которые пытаются перейти мост (WaitAsync) на случайное время, а затем покидают его (Release). Следите, чтобы на мосту одновременно было не более 3 героев.
2. **Среднее: "Собрание у короля"**. Создайте ManualResetEventSlim. Один поток-Herald (герольд) ждет 5 секунд (подготовка указа), а затем вызывает .Set(). 10 потоков-Knight (рыцари) ждут этого события (.Wait()). Когда герольд подает сигнал, все рыцари одновременно "получают указ" и начинают действовать.
3. **Сложное: "Ритуал призыва"**. Создайте CountdownEvent(5). 5 потоков-Acolyte (послушников) выполняют свою часть ритуала (занимают 1-2 секунды) и сигнализируют о готовности (Signal()). Главный поток-Summoner (заклинатель) ждет, когда все 5

послушников доложат о готовности (Wait()), и только тогда выводит в консоль "Демон призван!".

Важные напутствия для студентов:

- **Критическая секция** — это участок кода, где вы работаете с общим ресурсом. Делайте его как можно короче!
- lock — ваш главный и самый простой инструмент.
- ReaderWriterLockSlim — используйте, когда чтение происходит часто, а запись редко.
- SemaphoreSlim — используйте для ограничения одновременного доступа к пулу ресурсов.
- Manual/AutoResetEvent — используйте для оповещения между потоками.
- **Никогда не пишите lock в async методе** (кроме await внутри заблокированной секции), это верный путь к дедлокам. Для асинхронных сценариев используйте SemaphoreSlim.WaitAsync().

Удачи в освоении магии многопоточности!