## Multithreading

Demonstrate the work of the each case with console utility

1. Write a program, which creates an array of 100 Tasks, runs them and wait all of them are not finished. Each Task should iterate from 1 to 1000 and print into the console the following string: “Task #0 – {iteration number}”.
2. Write a program, which creates a chain of four Tasks. First Task – creates an array of 10 random integer. Second Task – multiplies this array with another random integer. Third Task – sorts this array by ascending. Fourth Task – calculates the average value. All this tasks should print the values to console
3. Write a program, which multiplies two matrices and uses class Parallel.
4. Write a program which recursively creates 10 threads. Each thread should be with the same body and receive a state with integer number, decrement it, print and pass as a state into the newly created thread. Use Thread class for this task and Join for waiting threads.
5. Write a program which recursively creates 10 threads. Each thread should be with the same body and receive a state with integer number, decrement it, print and pass as a state into the newly created thread. Use ThreadPool class for this task and Semaphore for waiting threads.
6. Write a program which creates two threads and a shared collection: the first one should add 10 elements into the collection and the second should print all elements in the collection after each adding. Use Thread, ThreadPool or Task classes for thread creation and any kind of synchronization constructions.
7. Create a Task and attach continuations to it according to the following criteria:
   1. Continuation task should be executed regardless of the result of the parent task.
   2. Continuation task should be executed when the parent task finished without success.
   3. Continuation task should be executed when the parent task would be finished with fail and parent task thread should be reused for continuation
   4. Continuation task should be executed outside of the thread pool when the parent task would be cancelled

## Async/await

1. Напишите консольное приложение для асинхронного расчета суммы целых чисел от 0 до N. N задается пользователем из консоли. Пользователь вправе внести новую границу в процессе вычислений, что должно привести к перезапуску расчета. Это не должно привести к «падению» приложения.
2. Напишите простейший менеджер закачек. Пользователь задает адрес страницы, которую необходимо загрузить. В процессе загрузки пользователь может ее отменить. Пользователь может задавать несколько источников для закачки. Скачивание страниц не должно блокировать интерфейс приложения.
3. Напишите простейший магазин по заказу еды. Пользователь может выбрать товар, и он добавляется в корзину. При изменении товаров происходит автоматический пересчет стоимости. Любые действия пользователя с меню или корзиной не должны влиять на производительность UI (замораживать).
4. У вас есть Entity, которое описывает класс пользователя, хранящийся в БД. Пользователь хранит информацию об Имени, Фамилии, Возрасте. Напишите пример асинхронных CRUD операций для этого класса.