Файл Runner.cs => стал PlaneRegistry.cs

1. Переносим метод main в отдельный модуль Program.cs из модуля Runner.cs
2. Переименовываем класс Runner в PlaneRegistry и делаем его статическим
3. Статическое свойство planes переименовываем в Planes
4. Изменим Console.WriteLine() – внутри цепочку методов расположим в одну строку

Файл Program.cs

1. Обращение к planes в main производим через поле уже статического класса PlaneRegistry.planes
2. Заменяем название обращения к статическому свойству класса PlaneRegistry с planes на Planes

Файл PassengerPlane.cs

1. Переменная \_passengersCapacity объявляется private
2. Метод PassengersCapacityIs() переименовываем в GetPassengersCapacity()
3. Переопределенный для класса метод ToString() становится однострочным – убираем лишние переводы строк
4. В переопределенном для класса методе Equals() возвращаемое значение, представленное несколькими логическими условиями, становится однострочным – убираем лишние переводы строк

Файл MilitaryPlane.cs

1. Переменная \_type объявляется private
2. Переопределенный для класса метод ToString() становится однострочным – убираем лишние переводы строк
3. В переопределенном для класса методе Equals() возвращаемое значение, представленное несколькими логическими условиями, становится однострочным – убираем лишние переводы строк
4. Метод PlaneTypeIs() переименовываем в GetPlaneType()

Файл Plane.cs

1. Переменные объявляются private

\_model

\_maxSpeed

\_maxFlightDistance

\_maxLoadCapacity

1. Метод GetMS() переименовывается в GetMaxSpeed()
2. Метод MAXFlightDistance() переименовывается в GetMaxFlightDistance()
3. Метод MAXLoadCapacity() переименовывается в GetMaxLoadCapacity()
4. Переопределенный для класса метод ToString() становится однострочным – убираем лишние переводы строк
5. В переопределенном для класса методе Equals() возвращаемое значение, представленное несколькими логическими условиями, становится однострочным – убираем лишние переводы строк

Файл Airport.cs

1. Переопределенный для класса метод ToString() становится однострочным – убираем лишние переводы строк
2. В методе GetTransportMilitaryPlanes() подняли return на одну строку выше
3. Метод SortByMaxDistance() переименовывается в SortPlanesByMaxFlightDistance()
4. Метод SortByMaxSpeed() переименовывается в SortPlanesByMaxSpeed()
5. Метод SortByMaxLoadCapacity() переименовывается в SortPlanesByMaxLoadCapacity()

Файл MilitaryType.cs

1. Переименовываем элементы перечислений FIGHTER, BOMBER, TRANSPORT в Fighter, Bomber, Transport

Исключаем из проекта файлы с перечислениями, которые нигде не используются – ClassificationLevel.cs и ExperimentalTypes.cs. Удаляем их

Файл AirportTest.cs

1. Для свойства planes в инициализации расположим скобку под скобкой
2. Список planes заполняется из статического класса PlaneRegistry, теперь он не будет повторять фрагмент кода инициализации самолетов
3. Переменные объявляются private и переименовываются

\_planes и \_planeWithMaxPassengerCapacity

1. Переименуем тестовый метод MyTest1 в TransportMilitaryPlanesCanBeDetected()
2. Суть проверки в обнаружении в аэропорту явно присутствующих военных транспортных самолетов

Исправляем код тестового метода:

Airport airport = new Airport(\_planes);

List<MilitaryPlane> transportMilitaryPlanes = airport.GetTransportMilitaryPlanes().ToList();

Assert.IsTrue(transportMilitaryPlanes.Count!=0);

1. Переименуем тестовый метод MyTest2 в GetPassengerPlaneWithMaxPassengersCapacityCanBeDetected ()
2. Исправляем код тестового метода:

Airport airport = new Airport(\_planes);

PassengerPlane expectedPlaneWithMaxPassengersCapacity = airport.GetPassengerPlaneWithMaxPassengersCapacity();

Assert.IsTrue(expectedPlaneWithMaxPassengersCapacity.Equals(\_planeWithMaxPassengerCapacity));

1. В тестовом методе MyTest3 убираем строку

airport = airport.SortPlanesByMaxLoadCapacity();

Ее значение сразу вставляется в эту строку:

List<Plane> planesSortedByMaxLoadCapacity = airport.SortPlanesByMaxLoadCapacity().GetPlanes().ToList();

1. Создаем поле класса – отсортированный по возрастанию максимальной загрузки список самолетов

private List<Plane> \_planesOrderedByMaxLoadCapacity = PlaneRegistry.Planes.OrderBy(x => x.GetMaxLoadCapacity()).ToList();

1. Исправляем код тестового метода MyTest3:

Airport airport = new Airport(\_planes);

List<Plane> expectedPlanesSortedByMaxLoadCapacity = airport.SortPlanesByMaxLoadCapacity().GetPlanes().ToList();

Assert.IsTrue(expectedPlanesSortedByMaxLoadCapacity.SequenceEqual(\_planesOrderedByMaxLoadCapacity));

1. Переименуем тестовый метод MyTest3 в PlanesAreSortedByMaxLoadCapacity()
2. Заменяем консольный вывод на запись в файл (по сути – собственный лог) – убираем выводы в консоль, ее редко читают или не читают вообще