Unit Testing with C# – Practical exam - 09 August 2016

# IntergalacticTravel

---

# Description

You are given an already built software modules. Figure out how the different modules are working. Check out the test requirements, to know exactly what you must test. For a warm-up - start with the tests that look trivial to you (those that you know you can implement in seconds). Think of the different testing strategies you learned during the course - \_Extending classes, Mocking dependencies, Testing private members through the class public API, etc\_. Follow the \_best practices\_ for unit testing that we discussed in our \_lectures\_ and \_workshops\_.

# Requirements

## Test cases for \_IntergalacticTravel.UnitsFactory\_ (5 tests)

1. GetUnit should return new Procyon unit, when a valid corresponding command is passed (i.e. "create unit Procyon Gosho 1");

- GetUnit should return new Luyten unit, when a valid corresponding command is passed (i.e. "create unit Luyten Pesho 2");

- GetUnit should return new Lacaille unit, when a valid corresponding command is passed (i.e. "create unit Lacaille Tosho 3");

- GetUnit should throw InvalidUnitCreationCommandException, when the command passed is not in the valid format described above. (Check for at least 2 different cases)

## Test cases for \_IntergalacticTravel.ResourcesFactory\_ (10 tests)

1. GetResources should return a newly created Resources object with correctly set up properties(Gold, Bronze and Silver coins), no matter what the order of the parameters is, when the input string is in the correct format. (Check with all possible cases):

\_Example:\_ The following lines should all create a new Resources object with 40 Bronze Coins, 30 Silver Coins and 20 Gold Coins.

create resources gold(20) silver(30) bronze(40)

create resources gold(20) bronze(40) silver(30)

create resources silver(30) bronze(40) gold(20)

create resources silver(30) gold(20) bronze(40)

create resources bronze(40) gold(20) silver(30)

create resources bronze(40) silver(30) gold(20)

- GetResources should throw InvalidOperationException, which contains the string "command", when the input string represents an invalid command. (Check with at least 2 different cases).

Invalid commands are any commands that does not follow the pattern described above.

\_Example:\_

create resources x y z

tansta resources a b

absolutelyRandomStringThatMustNotBeAValidCommand

- GetResources should throw OverflowException, when the input string command is in the correct format, but any of the values that represent the resource amount is larger than uint.MaxValue. (Check with at least 2 different cases)

\_Example:\_

create resources silver(10) gold(97853252356623523532) bronze(20)

create resources silver(555555555555555555555555555555555) gold(97853252356623523532999999999) bronze(20)

create resources silver(10) gold(20) bronze(4444444444444444444444444444444444444)

## Test cases for \_IntergalacticTravel.TeleportStation\_ (15 tests)

1. Constructor should set up all of the provided fields (owner, galacticMap & location), when a new TeleportStation is created with valid parameters passed to the constructor.

- TeleportUnit should throw ArgumentNullException, with a message that contains the string "unitToTeleport", when IUnit unitToTeleport is null.

- TeleportUnit should throw ArgumentNullException, with a message that contains the string "destination", when ILocation destination is null.

- TeleportUnit should throw TeleportOutOfRangeException, with a message that contains the string "unitToTeleport.CurrentLocation", when а unit is trying to use the TeleportStation from a distant location (another planet for example).

- TeleportUnit should throw InvalidTeleportationLocationException, with a message that contains the string "units will overlap" when trying to teleport a unit to a location which another unit has already taken.

- TeleportUnit should throw LocationNotFoundException, with a message that contains the string "Galaxy", when trying to teleport a unit to a Galaxy, which is not found in the locations list of the teleport station.

- TeleportUnit should throw LocationNotFoundException, with a message that contains the string "Planet", when trying to teleport a unit to a Planet, which is not found in the locations list of the teleport station.

- TeleportUnit should throw InsufficientResourcesException, with a message that contains the string "FREE LUNCH", when trying to teleport a unit to a given Location, but the service costs more than the unit's current available resources.

- TeleportUnit should require a payment from the unitToTeleport for the provided services, when all of the validations pass successfully and the unit is ready for teleportation.

- TeleportUnit should obtain a payment from the unitToTeleport for the provided services, when all of the validations pass successfully and the unit is ready for teleportation, and as a result - the amount of Resources of the TeleportStation must be increased by the amount of the payment.

- TeleportUnit should Set the unitToTeleport's previous location to unitToTeleport's current location, when all of the validations pass successfully and the unit is being teleported.

- TeleportUnit should Set the unitToTeleport's current location to targetLocation, when all of the validations pass successfully and the unit is being teleported.

- TeleportUnit should Add the unitToTeleport to the list of Units of the targetLocation (Planet.Units), when all of the validations pass successfully and the unit is on its way to being teleported.

- TeleportUnit should Remove the unitToTeleport from the list of Units of the unit's current location (Planet.Units), when all of the validations pass successfully and the unit is on its way to being teleported.

- PayProfits should return the total amount of profits(Resources) generated using the TeleportUnit service, when the argument passed represents the actual owner of the TeleportStation.

## Test cases for \_IntergalacticTravel.BusinessOwner\_ (1 test)

1. CollectProfits should increase the owner Resources by the total amount of Resources generated from the Teleport Stations that are in his possession.

## Test cases for \_IntergalacticTravel.Unit\_ (3 tests)

1. Pay should throw NullReferenceException if the object passed is null.

2. Pay should decrease the owner's amount of Resources by the amount of the cost.

3. Pay should return Resource object with the amount of resources in the cost object.