# *Programming IV (420-B42-HR)*

# *Assignment 2 – Testing and Basic Functions*

Date assigned: February 8, 2017

Date due: March 6, 2017

**Learning Objectives**

Upon successful completion of this assignment, the student will be able to:

* Create some basic functions for retrieving and manipulating HVK information.
* Create the test classes for managing reservations.

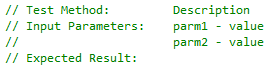
To do:

**Part A – Control Class Methods**

1. You are going to create a number of methods which will eventually be used to extract data from the HVK database. These are supporting functions which are useful and will be needed to populate the various forms you are creating in Web IV. Note: these will be part of control classes and NOT the entity classes you created in assignment 1. These should be in a new project which will reference the class library from assignment 1.
   1. listOwners – returns a list of all HVK owner names and numbers in last name order. The owner name should be in the form lastname, firstname.
   2. listPets – returns a list the pets for all pets owned by the owner specified in the input parameter. The output should be sorted by pet number.
   3. listReservations – returns a list of reservations for an owner. Each reservation should contain the reservation number, pet number, pet name and the start and end date for all reservations in the future for the owner. The owner number is the input parameter.
   4. listActiveReservations – returns a list of all currently active reservation including the reservation number, owner name, pet number, pet name run number and start and end date for all reservations currently active. If an owner number is specified, returns the active reservations only for that owner.
   5. listVaccinations – has a pet number as parameter and returns a list of the vaccinations for the pet including vaccination name, expiry date and validation flag.
   6. checkVaccinations – returns a list of vaccinations that need to be validated for a reservation. Parameters are reservation number and pet number. List is null if all vaccinations are up-to-date and validated.
   7. upcomingReservations – receives a date as parameter and returns a list of all reservations on or after that date. List should include reservation number, owner number, pet number, start and end date of reservation.
2. For now, create solution called B42A02\_HVKControl which contains a class library with the above methods. These methods will be in the control class(es) that will be accessed by the view (C40 assignments). They will, in turn, call classes in the model (next assignment) which will access the actual data.
3. Create a unit test project which will thoroughly test each of the above methods. Each method must be tested for all possible returns and combinations. The unit test project must be part of the same solution as the class library (two projects in the same solution, one a class library and the other a unit test project).
4. You will have to create stub methods to return data THAT MATCHES THE DATA IN THE HVK DATABASE. These stubs must be in a different project from both the class library and the unit tests. They can be in the same solution, however.

**Part B – Test cases for Make Reservation – in teams**

1. Using the user stories and business rules included and the use case narratives from last term, create a test library for testing the make reservation business flow.
2. Start by creating a new solution called B42A02\_ManageReservation. This solution contains a class library and a unit test library. Note, this solution MAY be combined with the above solution in the next assignment but for now MUST be a separate solution.
3. Start by creating empty control methods as listed below in the class library. These will be used so that the unit tests can reference something. Do NOT add any logic to these methods at this time. If you want to add some comments for things you will have to make sure you check (like cannot cancel a reservation that is already started) then you can do that.
4. For each of the methods listed below and any other ones you feel are required, generate unit test cases that will test the methods. These test cases MUST be based on data from the HVK database that was created. If there are scenarios which you feel should be tested where the database does not contain the correct or sufficient information, let me know and we will discuss.
5. Create a unit test library in the solution which will call the methods with the appropriate values. Each test method must be prefixed by comments explaining the test. The format is:



Where Test Method is a BRIEF description of what is being tested. Input Parameters are details of the parameters including what type of dog, etc and Expected Result is what will happen when the method is executed and HOW IT WILL BE VALIDATED.

1. Each test method must contain a set up section, an expected result section and an action section along with any asserts that can be done (if there are any).

Do:

1. Plan the project as a team. Identify the tasks required, prioritize them and estimate the time required for each task.
2. You will be given some class time for this (but not sufficient). At the end of each class you will be required to complete a self and team evaluation. These evaluations will form part of the mark for the class.
3. A student who misses a class/lab where we are working on this will receive zero for all the marks in that class.

**Make a Reservation User Stories**

Pet owners or kennel staff can issue a request to make a reservation.

Pets may only stay at the kennel if the pet’s vaccinations will all be up-to-date at the end of the stay.

Reservations are only created if there is suitable accommodation available for the pet for the entire duration of the pet’s stay.

**Business Rules:**

1. A reservation may be made for one or more dogs for the same owner.
2. A reservation may be made for any number of days in the future.
3. Dogs may only share when they have the same owner.
4. A maximum of two dogs can share a run.
5. A large dog may only be put in a large run.
6. A small or medium dog may be put in either a regular or a large run.
7. A reservation can only be completed if an appropriate sized run (or runs) is available for the duration of the stay.
8. The status for a successful reservation is confirmed.
9. An owner may make a reservation for a dog that has missing or expired vaccinations.
10. A dog must have all its vaccinations up to date for the duration of a stay at the kennel.
11. The vaccination certificates are checked by the staff when the dog arrives for the stay.
12. Dog runs are assigned manually by the kennel staff.

| **Function** | **Parameters** | **Description** |
| --- | --- | --- |
| **addReservation** | petNumber,  startDate,  endDate | Makes a reservation for a dog alone in a run |
| **addReservation** | petNumber1,  petNumber2,  startDate,  endDate | Makes a reservation for two dogs sharing a run |
| **addToReservation** | reservationNumber,  petNumber | Adds a new dog to an existing reservation |
| **addToReservation** | reservationNumber,  petNumber1,  petNumber2 | Adds two dogs sharing to an existing reservation. If one of the dogs is already included in the reservation, the other dog will share with it. |
| **cancelReservation** | reservationNumber | Cancels a reservation |
| **changeReservation** | reservationNumber, startDate,  endDate | Changes the start or end date of a reservation |
| **changeToSharing** | reservationNumber,  petNumber1,  petNumber2 | Changes two dogs in the same reservation who were not sharing to sharing |
| **changeToSolo** | reservationNumber,  petNumber1,  petNumber2 | Changes two dogs who were sharing to have separate runs |
| **deleteDogFromReservation** | reservationNumber,  petNumber | Deletes one dog and all the related services for that dog from a reservation. If there is only one dog in the reservation, the reservation is cancelled. |
| **checkVaccinations** | petNumber,  byDate | Ensures that the vaccinations for a dog will all be completed and up-to-date on a specified date |
| **checkRunAvailability** | startDate,  endDate,  runSize | Returns true if one or more runs of a given size are available during a given time period and false otherwise |

**To submit**

1. A ZIP format (*username*B42A02.zip) containing all submitted files on Moodle.