به نام خدا

گزارش کار و پاسخ سوالات پروژه ی اول تست نرم افزار

سارا اسعدی-810197448

مهرناز شمس‌آبادی-810196493

OwnerTest:

ابتدا در تابع setup مقادیر نمونه از دیتای مورد نظر خود را می سازیم برای وابسته ساختن دیتای تمامی تست ها از قبل این متود را قبل از تمامی تست ها صدا میکنیم.

getPets:

این تابع تابعی است که باید تمامی حیوانات متعلق به owner را به صورت مرتب شده بر حسب الفبا برگرداند.

1) `Sorted_pets_are_returned_correctly`
ابتدا لیست owner pets را با مقادیر مورد نظر پر میکنیم.
توقع می‌رود خروجی تست لیست مرتب شده ای از لیست داده شده و ذکر شده در بالا باشد.

استفاده از لیست های دیگر تنها تابع mutable را تست خواهد کرد که در دستور کار این پروژه نیست

2) `Owner_with_no_pet_returns_empty_list`
در این حالت چک می شود که در صورت خالی بودن لیست pet های یک owner، مقدار بازگشتی به چه صورت است تا احیانا برابر با null و یا دیتای نا مطلوب دیگری نباشد.

addPet:

1) `New_pet_is_added_to_owner_correctly`
در اینجا چک می‌کنیم که آیا یک pet که id نداشته باشد (new باشد) به درستی در لیست اضافه می شود یا خیر.
همچنین چک می‌کنیم تا pet owner به درستی مقدار دهی شده باشد

2) `Pet_with_id_is_not_added_to_owner_correctly`
در این حالت ابتدا به یکی از حیوان های تعریف شده id اختصاص می‌دهیم تا بتوانیم سناریو ی مربوطه را چک کنیم.
و سپس چک می‌کنیم که با وجود دستور addPet حیوان به لیست owner pets اضافه نشده باشد.
همچنین چک می‌کنیم تا owner حیوان به مقدار جدید تغییر کرده باشد.

* *getPet (این بخش اضافه بر بخش theorized نوشته شده است):*

1) `New_pet_not_found_if_we_ignore_new_pets`

در این حالت ابتدا حیوانی بدون id را به لیست حیوانات اضافه می‌کنیم تا چک کنیم در صورت صدا زدن getPet با نام این حیوان و مقدار ignoreNew صحیح تابع مقدار null را برمی‌گرداند

2) `Pet_with_id_found_if_we_ignore_new_pets`

در این حالت پس از set کردن مقدار id برای یک pet و اضافه کردن آن به لیست، تابع getPet با مقادیر نام pet و true باید حیوان صحیح را برگرداند.

تا اینجا پیدا کردن حیوان ها با id و بدون id با مقدار ignoreNew صحیح چک شده است

3) `New_pet_found_generally`

این سناریو چک می‌کند که پس از اضافه شدن pet بدون id به لیست و صدا زدن تابع getPet تابع، pet مورد نظر را برگرداند.

از تست 1و 3 به صحت شرط مرتبط با ignoreNew پی می‌بریم بنابراین ترکیبی مشابه با تست 3 برای تست 2 بی‌معنی خواهد بود. علی الخصوص که تغییری در مقدار خروجی ایجاد نخواهد کرد

4) `New_pet_not_found_generally`

در این حالت بررسی می‌شود که در صورتی که حیوانی خارج از لیست صدا زده شود مقدار برگشتی برابر با null باشد. به دلیل بالا new بودن یا نبودن و مقدار ignoreNew در تست تغییری ایجاد نخواهد کرد.

Owner_getPetTest:

در این قسمت ابتدا در کانستراکتور یکسری Pet تعریف کرده و به یکی از آن ها id اختصاص می‌دهیم.

برای datapoint اول مقدار اسم حیوانی را که می‌خواهیم به دنبال آن بگردیم را ست می‌کنیم

سه مقدار برای سه سناریو با id، بدون id و خارج از لیست.

در datapoint بعدی مقدار لیست pets را مشخص می‌کنیم شامل دو سناریو کامل و خالی

1) `Owner_with_no_pets_returns_null`

در این حالت تست می‌کنیم که در صورتی که لیست خالی باشد جدای از نام حیوان به ما مقدار null را برگرداند.
برای چک کردن خالی بودن لیست از assume برابر با لیست خالی استفاده می‌کنیم.

2) `Pet_not_found_generally`

در این حالت تست میکنیم که در صورتی که حیوان اصلا در لیست نباشد مقدار null را برگرداند
به همین خاطر با استفاده از assume از برابر بودن مقدار pet با نام حیوان خارج از لیست اطمینان حاصل میکنیم

3) `Pet_found_generally`

چک می‌کنیم که تابع getPet جدای از new بودن یا نبودن pet و در صورت خالی نبودن لیست حیوان درست را برگرداند
برای تست کردن صحت آن چک می‌کنیم که نام حیوان برگشتی از تابع با نام اولیه برابر باشد.

4) `New_pet_not_found_if_we_ignore_new_one`

در این حالت چک می‌کنیم که Pet بدون id با شرط true بودن مقدار ignoreNew مقدار null را برمی‌گرداند

به دلایل ذکر شده در بالا چک کردن سناریو های بیشتر صرفا چک کردن شرط if است و duplicate خواهند بود

Pet_getVisitsTest:

در این حالت ما چندین تاریخ را به عنوان datapoint تعریف می‌کنیم تا در نهایت بتوانیم جایگشتی از این تاریخ ها را داشته باشیم

1) `Sorted_visits_are_return_correctly`

برای چک کردن در نظر می‌گیریم که تاریخ ها با هم برابر نباشند تا مقدار expected را به توان به درستی مقدار دهی کردن

در نهایت چک می‌کنیم که تاریخ نتایج خروجی تابع getVisit به ترتیب مناسب باشند و در صورت برابری تک تک اعضای لیست مقدار صحیح بوده است

در نظر گرفتن مقادیر دیگری برای این تابع تنها تابع sort را تست خواهد کرد

PetService_findPetTest:

Constructor را به گونه ای تعریف می‌کنیم که بتواند متناسب با مقادیر پارامتر attribute های خود را ست کند

در قسمت setup قبل از هرچیز برای pet های از پیش تعریف شده ی خود id مناسب تعریف می‌کنیم

پارامتر ها را به گونه ای تعریف می‌کنیم که در صورتی که مقدار ورودی عدد ذکر شده باشد خروجی pet تعریف شده ی پس از آن باشد

1) `Pet_found_correctly`

برای استفاده از کلاس petService نیازمند به داشتن اطلاعات دیگری هستیم که تمامی آن ها را به صورت mock شده در اختیار constructor قرار میدهیم چرا که همچون fixture هایی هستند که ارتباطی به بخش تحت تست ندارند

سپس تست خواهیم کرد که آیا pet id بازگشتی از تابع petId آیا برابر با id ابتدایی هست یا خیر.

یکی از ساده‌ترین راه‌های تشخیص بین چارچوب و کتابخانه این است که توجه کنیم این کد خارجی مورد نظر، توسط کد ما فراخوانی و کنترل شده و مورد استفاده قرار می‌گیرد یا این که کد خارجی، کدی که ما توسعه داده‌ایم را فراخوانی و کنترل می‌کند. در حالت اول با یک کتابخانه طرف هستیم و در حالت دوم با یک چارچوب.

در مورد JUnit با توجه به این که ابزاری برای تست نرم‌افزار است و کد ما را فراخوانی و تست می‌کند، از لفظ چارچوب برای آن استفاده می‌کنیم.

تست اول صحیح است.

تست دوم در این تست سه مقدار assert وجود دارد ولی مقدار expected تست تنها یک exception را می پذیرد پس تنها عدم برقراری یکی از این سه شرط برای درستی تست کافی است و نمیتوان متوجه شد که ایراد از کجاست و دقیقا متعلق به کدام شرط می باشد.

تست سوم لازم است که مقدار new date را ابتدا در متغییری ذخیره بکنیم چرا که در این حالت هر بار صدا کردن new date دیتای جدیدی را به ما بر می‌گرداند.

تست 4 این تست وابسته به تست قبلی است و به طور مستقل صحیح نیست. تست ها باید از یکدیگر مستقل باشند و بر روی دیتای یکدیگر تاثیر نگذارند.