EJERCICIO GUIADO 2

César Manuel Blázquez Martín 100495797

Hugo Becerra Fernández 100495717

Iñaki Feijoó Basagoiti 100495947

ÍNDICE

Contenido

Función 1

	CRITERIA	VALID CLASSES	INVALID CLASSSES
CREDIT CARD	cifra de 16 digitos con algoritmo de Luhm valido	CEV1 VALIDA CREDIT CARD (5105105105105100)	CENV1 , son 16 digitos pero uno no es valido
CREDIT CARD	tipo de dato	CEV2 5105105105105100	CENV2 "no es un numero"
CREDIT CARD	longitud	CEV3 5105105105105100	CENV3 mas de 16 digitos (17 numeros)
CREDIT CARD		VLV1 5105105105105100	CENV4 menos de 16 digitos (15 numeros)
CREDIT CARD			VLNV1 17 digitos
CREDIT CARD			VLNV2 15 digitos
ID_CARD	cifra de 8 dígitos y una letra al final	CEV4 VALIDA ID_CARD(12345678Z)	CENV5 no hay exactamente una letra al final
ID_CARD	tipo de dato	CEV5 12345678Z	CENV6 "no es un número"
ID_CARD	longitud	CEV6 12345678Z	CENV7 menos de 8 dígitos
ID_CARD		VLV2 12345678Z	CENV8 mas de 8 digitos
ID_CARD			VLNV3 9 digitos
ID_CARD			VLNV4 7 digitos
ID_CARD			VLNV5 0 letras
ID_CARD			VLNV6 2 letras
NAME_SURNAME	al menos 2 cadenas de caracteres	CEV7 VALIDA NAME_SURNAME(JOSE LOPEZ)	CENV9 no hay al menos 2 cadenas de caracteres
NAME_SURNAME	tipo de dato	CEV8 JOSE LOPEZ	CENV10 "no son caracteres"
NAME_SURNAME	debe ser un valor entre 10 y 50	CEV9 JOSE LOPEZ	CENV11 "está fuera del rango"
NAME_SURNAME		VLV3 JOSE LOPEZ	VLNV7 9 caracteres
NAME_SURNAME			VLNV8 51 caracteres
PHONE_NUMBER	conjunto de 9 dígitos	CEV10 VALIDA PHONE_NUMBER(911234567)	CENV12 "no todos son números"
PHONE_NUMBER			CENV13 no son exactamente 9 dígitos
PHONE_NUMBER			VLNV9 10 digitos
PHONE_NUMBER			VLNV10 8 digitos
ROOM_TYPE	Solo puede ser "single","double" o "suite"	CEV11 VALIDA ROOM_TYPE(SINGLE)	CENV14 "El tipo de habitación no coincide con los posibles valores"
ARRIVAL	El formato solo puede ser "DD/MM/YYYY"	CEV12 VALIDA ARRIVAL(14/06/2024)	CENV15 "Formato de fecha inválido"
ARRIVAL	La fecha no puede ser anterior al día actual	VLV4 14/06/2024	VLNV11, la fecha es anterior al día actual
ARRIVAL	El día no puede ser mayor a 31		CENV16 "Error de día"
ARRIVAL	El día no puede ser menor a 1		CENV16 "Error de día"
ARRIVAL	El día no puede ser mayor a 30 en aquellos meses que tengan 30 días		CENV16 "Error de día"
ARRIVAL	El día no puede ser mayor a 28 en febrero		CENV16 "Error de día"
ARRIVAL	El día no puede ser mayor a 29 en febrero bisiesto		CENV16 "Error de día"
ARRIVAL	El mes debe estar entre 1 y 12		CENV17 "Error de mes"
NUM_DAYS	Debe ser un valor entre 1 y 10	CEV13 VALIDA NUM_DAYS(2)	CENV18 "Está fuera del rango"
NUM_DAYS		VLV5 2	VLNV12 0 días
NUM_DAYS			VLNV13 11 días

Clases de equivalencia y valores límite en los distintos parámetros de la función roomReservation:

CREDIT_CARD:

Como clases de equivalencia válidas, tenemos 3. La primera se cumple cuando el algoritmo de Luhn es válido. La segunda es cuando el tipo de dato es válido, es decir, son números. Por último, la longitud del número de la tarjeta. En cuanto a las clases de equivalencia no válidas, tenemos 4. Uno de los 16 dígitos no es válido, la segunda es cuando no es un número. La tercera describe el caso de que sean más de 16 dígitos y por último que sean menos de 16. En los valores límite tenemos como válido el caso de que tenga 16 dígitos y como no válidos 15 y 17.

ID_CARD:

Las clases de equivalencia válidas de Id_card, son las mismas son las mismas que las de credit_card cambiando la primera, ya que el D.N.I tiene 8 dígitos numéricos y una letra final. Las clases de equivalencia no válidas son aquellas que contemplan los casos de que no haya una letra final, no sea un número o tenga mas de 8 dígitos o menos. En cuanto a los valores límite, válido es el caso de 8 números y 1 letra final, y los no válidos, aquellos que tengan 7 o 9 dígitos numéricos y 0 o 2 letras.

NAME_SURNAME:

En este caso las clases de equivalencia válidas contemplan al menos 2 cadenas de caracteres, que el tipo de dato sea un string y que tenga entre 10 y 50 caracteres. Las no válidas recogen los casos de que no sean caracteres, este fuera de rango o no se proporcionen dos cadenas. Los valores límite son el nombre mas pequeño, 10 caracteres, y los no válidos son 9 y 51 caracteres.

PHONE_NUMBER:

La única clase de equivalencia válida del teléfono es que tenga 9 dígitos. En el caso de las no válidas es que no sean números o no sean exactamente 9 dígitos. Los valores límite no válidos contemplan que sean 8 o 10 dígitos.

ROOM_TYPE:

La clase de equivalencia válida es que sea una palabra dentro de estas tres posibles; single, double o suite. La clase de equivalencia no válida es aquella en la que el tipo de habitación no es alguno de esos tres.

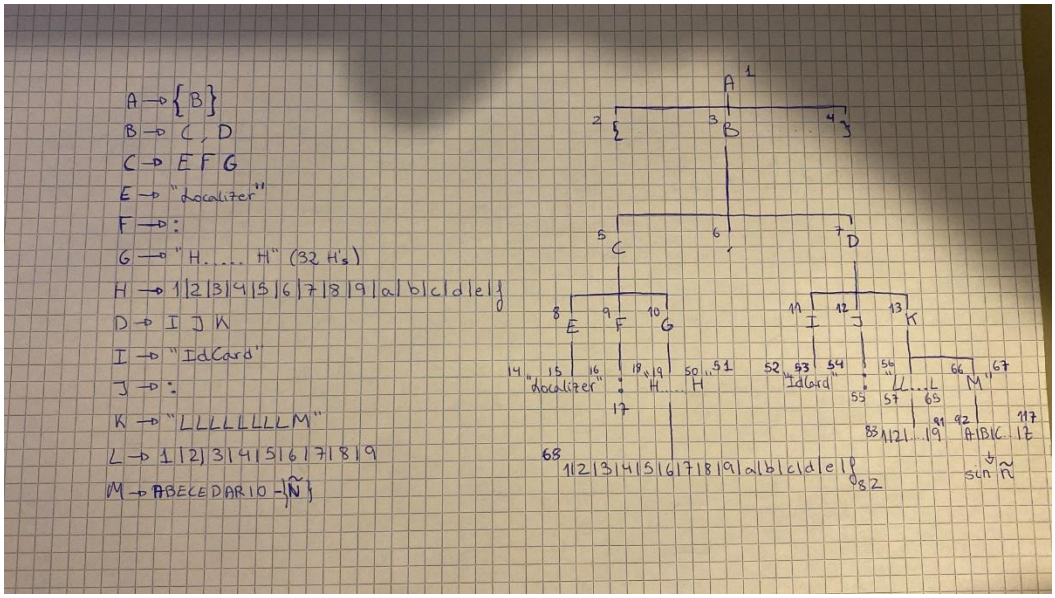
ARRIVAL:

Las clases de equivalencia de la fecha de llegada de un cliente manejan que el formato de la fecha sea el correcto; DD/MM/YYYY. En los valores límite se multiplica el número de casos, ya que contemplamos mucha variedad de situaciones que se pueden dar, que el día sea entre 1 y 31, o 1 y 30 los meses que no tengan 31 días, o 28 en el caso de febrero. Así también como que los meses no pueden ser de 13 hacia arriba o el año de llegada sea anterior al actual.

NUM_DAYS:

El número de días que un cliente puede quedarse en el hotel es entre 1 y 10, ahí está su clase de equivalencia válida, y por consiguiente la no válida, que es que se salga de ese rango. Un valor límite válido es un número de días entre 1 y 10, y los dos tipos de valores límite no válidos es que tengamos 0 días u 11.

Función 2



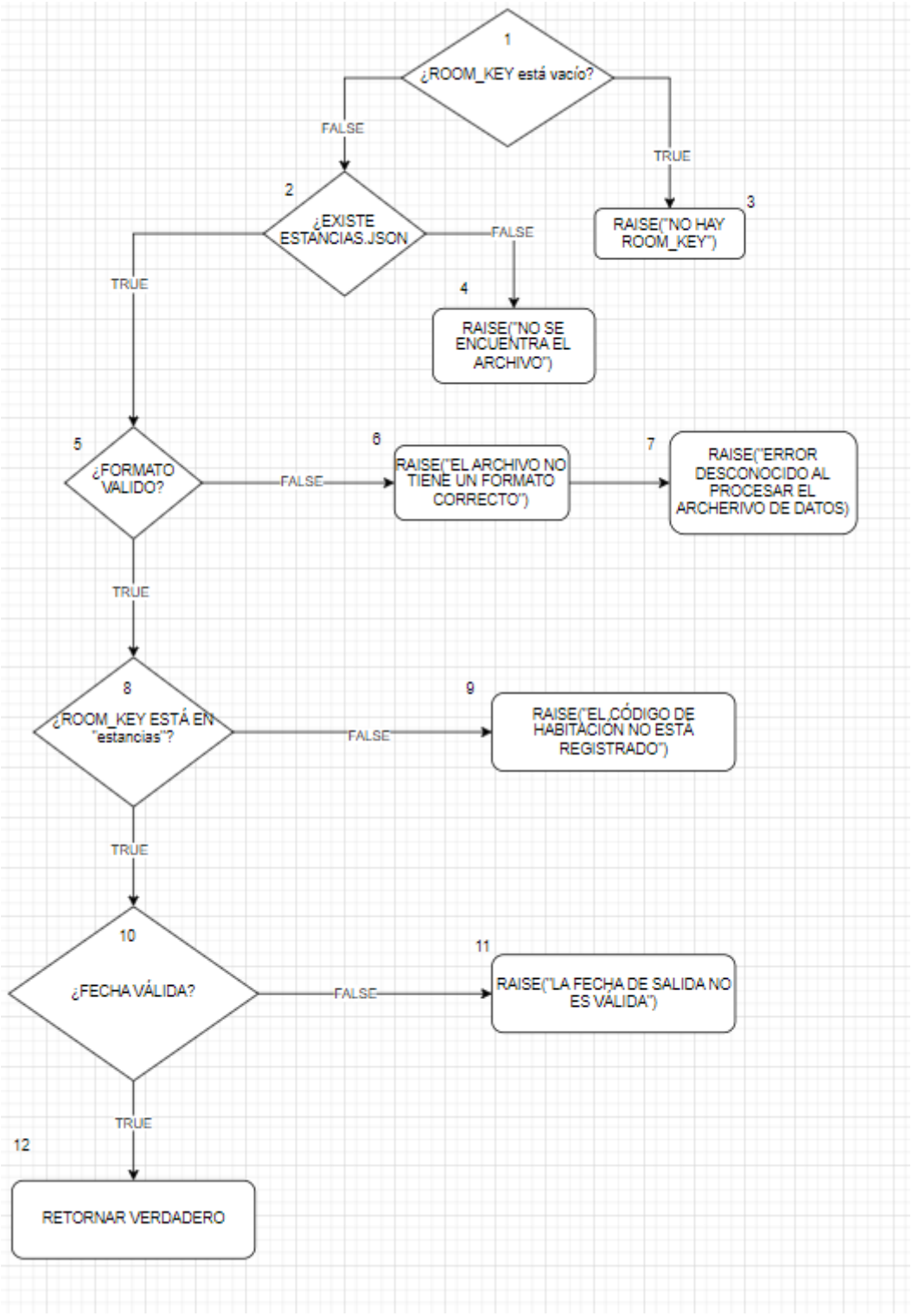
En base a la gramática definida anteriormente, se han definido una serie de casos de prueba, concretamente 21, en los cuales se comprueba la integridad del árbol. El primero es un caso de prueba que cubre todos los nodos, y por tanto es válido (se especifica en el Excel). A partir del segundo se comienzan a contemplar errores en distintos módulos o distintos nodos, los cuales se especifican también en el Excel.

Centrándonos en el código, se han seguido dos estructuras:

- Para el caso válido, se hace un assertEqual básico donde compara el localizador que se ha creado en el test con el almacenado en el .json

- Para el resto de casos, al definir los tipos de errores, se comprueba con un assertEqual la excepción que genera el test con el esperado.

Función 3


1
¿ROOM_KEY está vacío?
FALSE
TRUE
2
¿EXISTE ESTANCIAS.JSON
FALSE
TRUE
3
RAISE("NO HAY ROOM_KEY")
4
RAISE("NO SE ENCUENTRA EL ARCHIVO")
5
¿FORMATO VALIDO?
FALSE
6
RAISE("EL ARCHIVO NO TIENE UN FORMATO CORRECTO")
7
RAISE("ERROR DESCONOCIDO AL PROCESAR EL ARCHERIVO DE DATOS)
TRUE
8
¿ROOM_KEY ESTÁ EN "estancias"?
FALSE
9
RAISE("EL CÓDIGO DE HABITACION NO ESTÁ REGISTRADO")
TRUE
10
¿FECHA VÁLIDA?
FALSE
11
RAISE("LA FECHA DE SALIDA NO ES VÁLIDA")
TRUE
12
RETORNAR VERDADERO

En cuanto al diagrama de flujo hemos creado 12 nodos. El primero sirve para comprobar la existencia de room_key, en caso negativo lanzamos una excepción y en caso positivo pasamos a comprobar la existencia del archivo json. Si no existe este archivo, se lanza su correspondiente excepción. Si existe, comprobamos la validez de su formato y en caso de que no sea el correcto lanzamos dos excepciones, una que avisa de que el formato no es el correcto y otra de error de procesamiento. Si el archivo está en el formato correcto habrá que ver si la room_key proporcionada se encuentra en dicho archivo, si no se encuentra se lanza una excepción y si se encuentra pues ya pasaríamos a comprobar la validez de la fecha y en caso afirmativo devolvemos True. En caso negativo salta una excepción.

Para los casos de prueba del código solo pudimos implementar 3, uno válido y dos inválidos sobre si no se introducía room_key o si la room_key no estaba registrada dado a que a esta función solo le entra como argumento room_key.