//Ejercicio Integrador Swift

Proyecto Integrador Swift

Es momento de recopilar conocimientos y ponerlos en práctica en un corto proyecto; primero se explicarán los requerimientos del programa de forma general y luego se irá avanzando progresivamente sobre cada uno de ellos. Recibirás una parte de código inicial que deberás completar según los ejercicios lo indiquen para cumplir con los requerimientos.

i Durante los ejercicios verás unos mensajes de este tipo; son ayudas que te guiarán en las respuestas en caso que tengas dudas sobre cómo avanzar, de igual forma recuerda que puedes recurrir al mentor.

🅿️ 🚗 AlkeParking

AlkeParking es un estacionamiento que permite estacionar diferentes tipos de vehículos (auto, moto, minibus y bus) con un cupo máximo de 20 vehículos.

- Cuando se va a ingresar un vehículo se ingresa la patente y el tipo, y se valida que no haya ningún otro vehículo con esa misma patente en el estacionamiento.
- Cuando un vehículo va a ser retirado se cobra una tarifa determinada por las siguientes reglamentaciones
- Las primeras 2 horas de estacionamiento tendrán un costo fijo determinado por el tipo de vehículo (auto: $20, moto: $15, mini bus: $25, bus: $30).
- Luego de las 2 primeras horas se cobrarán $5 por cada 15 minutos o fracción independiente del tipo de vehículo. Por ejemplo para un auto se tendrían las siguientes tarifas:

Ingreso	Salida	Tarifa
18:00	18:46	$20
18:00	20:00	$20
18:00	21:13	$45
18:00	21:18	$50
18:00	21:30	$50

Los vehículos pueden tener (opcionalmente) una tarjeta de descuentos, la cual reduce la tarifa total de estacionamiento un 15%.

- Los valores de las tarifas no deben incluir centavos; en caso de tenerlos deben descartarse.

- Por solicitud de la administración de *AlkeParking*, debe tenerse un registro del total de vehículos que se retiran del estacionamiento, junto con el total de las ganancias recibidas.
- La administración podrá solicitar en cualquier momento la lista de las patentes de los vehículos que se encuentran en el estacionamiento.

Para comenzar, descarga el archivo `AlkeParking.playground`.

Punto de partida - Modelado

En primer lugar, se definió una estructura `Parking` que representa el estacionamiento del programa y contiene una propiedad `vehicles` en la cual se irán almacenando los vehículos que están estacionados.

💻 Ejercicio 1

¿ Por qué se define `vehicles` como un `Set`?

i Recuerda en qué se diferencian los `Set` de los `Array`. ¿Podrían existir dos vehículos iguales?

Para que se pueda estacionar un vehículo deben cumplirse ciertas características:

1) Tener una patente.
2) Debe ser de un tipo de vehículo permitido en el estacionamiento.
3) Se debe registrar la fecha de ingreso.
4) El vehículo podrá poseer una tarjeta de descuentos para el pago de la tarifa.
5) Calcular el tiempo total que permaneció en el estacionamiento.

Estas características definen el *acuerdo inicial* que deben cumplir todos los vehículos que ingresen al parqueadero; debido a este requerimiento se definió el protocolo `Parkable` que contiene la propiedad `plate` que corresponde a la patente del vehículo (requerimiento 1.); más adelante se irán incluyendo las propiedades las faltantes para cumplir con todos los requerimientos.

Se definió la estructura `Vehicle` que implementa dicho protocolo y, además, dado que un vehículo será único e identificado por la patente, `Vehicle` implementa también el protocolo `Hashable` que sirve para que los elementos puedan identificarse de forma única; se incluyen también las funciones

`hash(into:)` y `==(lhs:rhs:)`, necesarias para indicar cuáles propiedades son requeridas para identificar un vehículo como único.

Puedes ver más sobre el protocolo y estas funciones en la documentación de Apple.

💻 Ejercicio 2

1) El segundo requerimiento (2.) indica que el vehículo debe ser de uno de los tipos permitidos. Para esto, crea una enumeración llamada `VehicleType` que contenga los tipos de vehículo que se pueden estacionar (auto, moto, mini bus y bus).
2) Para satisfacer el requerimiento, incluye una propiedad `type` en el protocolo que sea del tipo de la enumeración.
   - ¿Puede cambiar el tipo de vehículo en el tiempo? ¿Debe definirse como variable o como constante en `Vehicle`?
3) Dado que la tarifa del parqueadero dependerá del tipo de vehículo es necesario agregar una propiedad en la enumeración; indique el valor correspondiente. Agrega una *computed property* de tipo `Int` que indique la tarifa para cada tipo:

Tipo	Tarifa
Auto	20
Moto	15
Mini bus	25
Bus	30

i ¿Qué elemento de control de flujos podría ser útil para determinar la tarifa de cada vehículo en la *computed property*: ciclo `for`, `if` o `switch`?

🚘 Requerimientos de vehículos

Todavía quedan 3 requerimientos del protocolo por satisfacer.

El tercer requerimiento (3.) pide que se almacene la fecha de ingreso que servirá para calcular más adelante el tiempo total de estadía en el estacionamiento. Para esto se puede utilizar la estructura `Date` de Swift, la cual se usa para representar *un punto específico en el tiempo*. Para tomar el tiempo actual se utiliza `Date()`

Para conocer más sobre la estructura `Date` visita el link de la documentación de Apple.

Por su parte, el cuarto requerimiento (4.) indica que cada vehículo puede contar (de forma opcional) con una tarjeta de descuentos. Esta tarjeta está determinada por un `String` y se validará más adelante en el cálculo de la tarifa final.

💻 Ejercicio 3

1) Agrega una propiedad `checkInTime` para satisfacer el requerimiento 3.
2) Agrega una propiedad `discountCard` para satisfacer el requerimiento 4.

i ¿Dónde deben agregarse las propiedades, en `Parkable`, `Vehicle` o en ambos?

i La tarjeta de descuentos es opcional, es decir que un vehículo puede no tener una tarjeta y su valor será `nil`. ¿Qué tipo de dato de Swift permite tener este comportamiento?

Solo falta el último requerimiento por cumplir: indicar el tiempo que ha transcurrido desde que el vehículo ingresó al estacionamiento. Para esto, se tendrá otra propiedad `parkedTime` que deberá indicar el total de minutos. Para calcular este valor, Swift posee una estructura Calendar que, especificando la fecha de inicio y de fin, indica los componentes de la fecha solicitados (hora, minutos, segundos, etc.)

```
let mins = Calendar.current.dateComponents([.minute], from: checkInTime, to: Date()).minute ?? 0
```

Para conocer más sobre la estructura Calendar visita el link de la documentación de Apple

---

💻 Ejercicio 4

1) Agrega una propiedad `parkedTime` para satisfacer el requerimiento 4, según las indicaciones anteriores.

i El tiempo de estacionamiento dependerá de `parkedTime` y deberá *computarse* cada vez que se consulta, teniendo como referencia la fecha actual. ¿Qué tipo de propiedad permite este comportamiento: *lazy properties*, *computed properties* o *static properties*?

En este punto el modelo `Vehicle` está completo y se han satisfecho todos los requerimientos para que puedan ser añadidos al estacionamiento. Para probarlo se crearán algunas instancias de vehículos diferentes y serán añadidas directamente al `Set` de `Parking`.

Al final del Playground agrega el siguiente código:

```
var alkeParking = Parking()


let car = Vehicle(plate: "AA111AA", type: VehicleType.car,
checkInTime: Date(), discountCard: "DISCOUNT_CARD_001")


let moto = Vehicle(plate: "B222BBB", type:
VehicleType.moto, checkInTime: Date(), discountCard: nil)


let miniBus = Vehicle(plate: "CC333CC", type:
VehicleType.miniBus, checkInTime: Date(), discountCard:
nil)


let bus = Vehicle(plate: "DD444DD", type: VehicleType.bus,
checkInTime: Date(), discountCard: "DISCOUNT_CARD_002")


alkeParking.vehicles.insert(car)


alkeParking.vehicles.insert(moto)


alkeParking.vehicles.insert(miniBus)


alkeParking.vehicles.insert(bus)
```

Se ha creado un vehículo de cada tipo y utilizando la función `insert(_:)` de los `Set` se ha agregado cada uno de ellos. Para validar que se hayan ingresado correctamente, verifica que en la zona de resultados (al lado derecho del Playground) en cada `insert` se tenga `inserted: true` o, si lo deseas, puedes incluir cada inserción dentro de un print y verificar que se muestren cuatro `true` en la consola:

```
print(alkeParking.vehicles.insert(car).inserted)  // true

print(alkeParking.vehicles.insert(moto).inserted)  // true

print(alkeParking.vehicles.insert(miniBus).inserted)// true

print(alkeParking.vehicles.insert(bus).inserted)  // true
```

Para validar que, en efecto, no se pueden ingresar dos vehículos con la misma patente prueba creando otra instancia de `Vehicle` con alguna de las patentes anteriores, y verifica que se muestre `false` en consola:

```
let car = Vehicle(plate: "AA111AA", type: VehicleType.car,
checkInTime: Date(), discountCard: "DISCOUNT_CARD_001")

let car2 = Vehicle(plate: "AA111AA", type: VehicleType.car,
checkInTime: Date(), discountCard: "DISCOUNT_CARD_003")

print(alkeParking.vehicles.insert(car).inserted)  // true

print(alkeParking.vehicles.insert(car2).inserted)  // false
```

Finalmente, para retirar un vehículo del estacionamiento, se utiliza la función `remove(_:)` de `Set`:

```
alkeParking.vehicles.remove(moto)
```

➕🚗 Ingresar vehículo

Actualmente se pueden ingresar tantos vehículos como se deseen accediendo directamente a la propiedad `vehicles` de `Parking`, pero en uno de los requerimientos iniciales se especifica que el estacionamiento tiene únicamente 20 cupos, por lo tanto debe restringirse el ingreso cuando se llega al tope. Para esto se agregará una función `checkInVehicle(_:onFinish:)` en `Parking`, que recibirá el vehículo a ingresar y una closure que tendrá como parámetro un `Bool` indicando si se pudo ingresar el vehículo correctamente o no:

```
mutating func checkInVehicle(_ vehicle: Vehicle, onFinish:
(Bool) -> Void) {


// Insert vehicle here


}
```

La función es definida como `mutating` pues al estar modificando una propiedad de una `struct` debe indicarse explícitamente que se desea que la instancia sea modificada. Este post explica un poco más el uso de `mutating`.

💻 Ejercicio 5

1) Agrega una propiedad que indique el cupo máximo de vehículos que se pueden estacionar.
2) Dentro de la función `checkInVehicle(_:onFinish:)`:
   - Agrega una validación utilizando `guard` que verifique si aún hay cupos disponibles. En caso de que no haya, ejecuta la closure `onFinish` mandando como parámetro `false`
   - Agrega una segunda validación que compruebe si la placa del vehículo que se desea ingresar ya se encuentra registrada. En caso

de que se sí ejecuta la closure `onFinish`, mandando como parámetro `false`.
- En caso de que el vehículo se haya podido ingresar correctamente ejecuta la closure `onFinish`, mandando como parámetro `true`.

i Revisa la documentación de Apple para identificar qué función de `Set` puede ser útil para obtener la cantidad de vehículos en el estacionamiento.

i La función `insert(_:)`de `Set` retorna una tupla y en uno de sus valores indica si el elemento fue insertado correctamente. Revisa la documentación de Apple para más información sobre esta función.

Para probar que esté funcionando correctamente se deben ingresar al menos 20 vehículos. Acá te dejamos algunos, completa los faltantes y llama la función de check-in para cada uno de ellos:

En caso que no se pueda ingresar el vehículo (es decir cuando se recibe `false`) debe mostrarse en consola el mensaje `"Sorry, the check-in failed"` y en caso que se realice correctamente deberá mostrarse `"Welcome to AlkeParking!"`.

```
var alkeParking = Parking()


let vehicle1 = Vehicle(plate: "AA111AA", type:
VehicleType.car, checkInTime: Date(), discountCard:
"DISCOUNT_CARD_001")


let vehicle2 = Vehicle(plate: "B222BBB", type:
VehicleType.moto, checkInTime: Date(), discountCard: nil)


let vehicle3 = Vehicle(plate: "CC333CC", type:
VehicleType.miniBus, checkInTime: Date(), discountCard:
nil)
```

```
let vehicle4 = Vehicle(plate: "DD444DD", type:
VehicleType.bus, checkInTime: Date(), discountCard:
"DISCOUNT_CARD_002")


let vehicle5 = Vehicle(plate: "AA111BB", type:
VehicleType.car, checkInTime: Date(), discountCard:
"DISCOUNT_CARD_003")


let vehicle6 = Vehicle(plate: "B222CCC", type:
VehicleType.moto, checkInTime: Date(), discountCard:
"DISCOUNT_CARD_004")


let vehicle7 = Vehicle(plate: "CC333DD", type:
VehicleType.miniBus, checkInTime: Date(), discountCard:
nil)


let vehicle8 = Vehicle(plate: "DD444EE", type:
VehicleType.bus, checkInTime: Date(), discountCard:
"DISCOUNT_CARD_005")


let vehicle9 = Vehicle(plate: "AA111CC", type:
VehicleType.car, checkInTime: Date(), discountCard: nil)


let vehicle10 = Vehicle(plate: "B222DDD", type:
VehicleType.moto, checkInTime: Date(), discountCard: nil)


let vehicle11 = Vehicle(plate: "CC333EE", type:
VehicleType.miniBus, checkInTime: Date(), discountCard:
nil)


let vehicle12 = Vehicle(plate: "DD444GG", type:
```

```
VehicleType.bus, checkInTime: Date(), discountCard:
"DISCOUNT_CARD_006")


let vehicle13 = Vehicle(plate: "AA111DD", type:
VehicleType.car, checkInTime: Date(), discountCard:
"DISCOUNT_CARD_007")


let vehicle14 = Vehicle(plate: "B222EEE", type:
VehicleType.moto, checkInTime: Date(), discountCard: nil)


let vehicle15 = Vehicle(plate: "CC333FF", type:
VehicleType.miniBus, checkInTime: Date(), discountCard:
nil)
```

Si ingresas 20 vehículos diferentes deberás ver en consola 20 veces `"Welcome to AlkeParking!"`; a partir del vehículo 21 deberás ver en consola `"Sorry, the check-in failed"`. Prueba ingresando un vehículo con patente repetida y deberás ver el mensaje `"Sorry, the check-in failed"`.

Una forma más sencilla para ingresar todos los vehículos, evitando tener el código repetido, es mediante un ciclo en el que se itere sobre cada vehículo y se ingrese al estacionamiento.

💻 Ejercicio 6

1) Elimina los llamados a la función de check-in anteriores (mantén los vehículos creados).
2) Crea un arreglo con los vehículos creados.
3) Agrega un ciclo que itere sobre el arreglo de vehículos, creado en el punto 2, y que vaya añadiendo cada uno. De esta forma tendrás un único lugar del código que llama la función de check-in y garantizarás que para todos se realice la misma acción al finalizar (`onFinished`).

Al finalizar el punto 3 deberás volver a ver en consola 20 mensajes `"Welcome to AlkeParking!"` y a partir del vehículo 21 deberás ver en consola `"Sorry, the check-in failed"`.

➖🚗 Retirar vehículo

Como se vio anteriormente, para retirar un vehículo se utiliza la función `remove(_:)`

```
alkeParking.vehicles.remove(moto)
```

Pero es necesario que al retirar un vehículo se realice el cobro correspondiente. Para esto, se creará primero la función que recibe los datos del vehículo y valida el check-out y posteriormente la función que calcula el cobro.

💻 Ejercicio 7

1) Agrega una función `checkOutVehicle` en `Parkable` que reciba:
   - La patente del vehículo `plate` como un `String`
   - Una closure `onSuccess` que se ejecutará cuando se complete el check-out y que reciba como parámetro un `Int` que corresponderá al monto que se debe pagar.
   - Una closure `onError` que se ejecutará en caso de un error en el check-out.
2) Valida que se encuentre un vehículo con esa patente en el estacionamiento. En caso de que no se encuentre haz un llamado a la closure correspondiente. En caso afirmativo mantén una referencia a ese vehículo, pues se requerirá para el cálculo de la tarifa.
3) Remueve el vehículo del estacionamiento y haz un llamado a la closure correspondiente, pasando como parámetro cualquier valor entero. (En el siguiente ejercicio se hará el cálculo de la tarifa y se reemplazará el valor en este parámetro).

i Revisa la documentación de Apple para identificar que función de `Set` puede ser útil para obtener un elemento dada cierta condición.

i Se está modificando una propiedad de un `struct` ¿Qué consideración debe tenerse en la definición de la función?

Ahora se agregará la función para calcular el total de la tarifa.

Volviendo al requerimiento inicial, para calcular el cobro debe tenerse en cuenta inicialmente las siguientes consideraciones:

1) Las primeras 2 horas de estacionamiento tendrán un costo fijo determinado por el tipo de vehículo (auto: $20, moto: $15, mini bus: $25, bus: $30).

2) Luego de las 2 primeras horas se cobrarán $5 por cada 15 minutos o fracción.

Por ejemplo para un auto:

Ingreso	Salida	Tarifa
18:00	20:00	$20
18:00	19:59	$20
18:00	21:13	$45
18:00	21:18	$50
18:00	21:30	$50

Por ejemplo, el auto ingresa a las 18:00 y se retira a las 21:18. Transcurrieron 3 horas y 18 minutos:

- Se inicia con los $20 de las 2 horas iniciales.
- Quitando las 2 horas iniciales queda 1 hora y 18 minutos = 78 minutos.
- Se dividen esos 78 minutos en grupos de 15 y da 5,2 bloques. Como la fracción también se incluye se redondea a 6 bloques.
- El total serían $20 más $5 por cada bloque = $50

💻 Ejercicio 8

1) Crea una función `calculateFee` que reciba un tipo de vehículo (`type`) y un `Int` (`parkedTime`) con la cantidad de minutos transcurridos. Además, la función debe retornar un `Int`que corresponderá al valor a pagar.
2) Calcula la tarifa final, según los requerimientos explicados anteriormente.

El último requerimiento para el cálculo de la tarifa especifica que los vehículos pueden tener (opcionalmente) una tarjeta de descuentos, la cual reduce la tarifa total de estacionamiento un 15%.

💻 Ejercicio 9

1) A la función anterior, agrega un parámetro booleano `hasDiscountCard` para indicar si debe aplicarse un descuento a la tarifa.
2) Luego de obtener el total de la tarifa, aplica el descuento si corresponde y retorna el valor actualizado (los centavos no deberán contarse en el cobro de la tarifa).

i Recuerda que las operaciones matemáticas solo pueden realizarse entre valores del mismo tipo.

Ahora que ya está completo el cálculo de la tarifa, solo falta actualizar este valor al momento de realizar el check-out. Para esto, en la función `checkOutVehicle(plate:onSuccess:onError:)`, antes de remover el vehículo se debe realizar el cálculo y finalmente enviar dicho valor en `onSuccess`

💻 Ejercicio 10

1) Completa la función de check-out para enviar la tarifa correspondiente en `onSuccess`

i ¿Qué validación debe hacerse para determinar si el vehículo tiene descuento? Recuerda que lo único que interesa es si tiene o no tarjeta de descuento, más no interesa el valor de la misma.

Verifica que el valor a cobrar corresponda con los ejemplos de la tabla anterior y realiza pruebas con los demás tipos de vehículo. En caso que se tenga un error durante el check-out debe mostrarse el mensaje `"Sorry, the check-out failed"` y cuando un vehículo se retira correctamente debe mostrarse el mensaje `"Your fee is $40. Come back soon"` donde 40 debe corresponder al valor que se cobra.

➖🚗 Requerimientos de administración

Solo restan los requerimientos de parte de la administración:
En el primer requerimiento se pide mantener un registro del total de vehículos que se retiran del estacionamiento, junto con el total de las ganancias recibidas. Para esto se agregará una tupla que contenga ambos valores.

💻 Ejercicio 11

1) Agrega una propiedad de tipo `(Int, Int)` en `Parking` para acumular tanto la cantidad de vehículos retirados como las ganancias recibidas y actualiza su valor cada que se retira un vehículo.
2) Agrega una función en `Parking` que muestre la información de las ganancias, por ejemplo `"15 vehicles have checked out and have earnings of $345"`

Finalmente, se pide que se listen las patentes de los vehículos que se encuentran en el estacionamiento.

💻 Ejercicio 12

1) Agrega una función `listVehicles` en `Parking` que liste las patentes de los vehículos.

i Debes interar sobre los vehículos y por cada uno mostrar un mensaje en consola.

Con esta última función se han completado los requerimientos de *AlkeParking*.

Puedes realizar las pruebas que consideres necesarias en el Playground y validar los resultados con el mentor.

alkemy