Ejercicios

Ejercicio 1

Crear la clase Avion
En src\main\scala del proyecto plantilla
Usar como nombre de archivo Avion.scala
Crear una instancia en el REPL (sbt console desde el directorio del proyecto)

Ejercicio2

Añadir un parámetro a Avion llamado numero de tipo Int
Salir del REPL y volver a entrar
Crear una instancia de Avion

Ejercicio3

Intentar acceder al número del avión que hemos creado
Salir del REPL
Promover el parámetro de Avion usando un val
Añadir a Avion un primer parámetro llamado tipo, una String y promoverlo
Volver a entrar en el REPL y repetir el intento

Ejercicio4

Crear una clase llamada Duracion
Dos parámetros “promovidos”: horas y minutos de tipo Int
Crear una instancia en el REPL y acceder a los valores

Ejercicio5

Añadir un atributo inmutable llamado enMinutos a la clase Duración
Inicializarlo a las horas convertidas a minutos + los minutos
Crear una instancia y comprobar su valor

Ejercicio6

Añadir un método llamado menos que retorne Int y con un parámetro llamado otro de tipo Duracion
El método debe obtener la diferencia de las dos duraciones en minutos
Comprobar en el REPL

Ejercicio7

Añadir un método llamado “-” a Duracion que delegue en el método anterior
Comprobar en el REPL con la notación normal y la infija

Ejercicio8

Modificar los dos parámetros de Duracion para que tengan 0 como valor por defecto
Comprobar en el REPL

Ejercicio9

Mover las dos clases al paquete com.curso.scala, descomentar en build.sbt y comprobar en el REPL

Ejercicio10

Crear un companion para Duracion con un método llamado desdeMinutos, con un parámetro de tipo Int que representa minutos y devuelve una Duracion.
Comprobar en el REPL

Ejercicio11

Usar require para asegurarse de que las horas y minutos de Duracion están en rango: entre 0 y 23 y entre 0 y 59
Comprobar en el REPL

Ejercicio12

Convertir ambas clases en case clases. Comprobar en el REPL

Ejercicio13

Crear la clase Aeropuerto con un parámetro llamado nombre de tipo String
Añadir un parámetro de clase a Avion, llamado itinerario de tipo secuencia de Aeropuerto
Verificar que itinerario tiene al menos dos elementos. Comprobar en el REPL

Ejercicio14

Cambiar el tipo de itinerario a Seq[(Duracion,Aeropuerto)]
Añadir un atributo aeropuertos a Avion, de tipo Seq[Aeropuerto]. Inicializarlo a partir de itinerario

Ejercicio15

Crear la clase PlanDeVuelo con un parámetro de clase aviones de tipo Set[Avion]
Añadir un atributo a la clase llamado aeropuertos e inicializarlo con todos los aeropuertos de todos los aviones, sin aviones duplicados

Ejercicio16

Añadir a PlanDeVuelo un método llamado avionesEnAeropuerto que recibe como parámetro un aeropuerto y devuelve el conjunto de aviones que tengan ese aeropuerto en su itinerario, sin duplicados

Ejercicio17

Añadir a PlanDeVuelo un método llamado aterrizaEn con un parámetro de tipo Aeropuerto y que devuelve un Set[(Duracion, Avion)] compuesto por todos los aviones que tienen en su itinerario el aeropuerto dado y la duración que pasan en dicho aeropuerto

Ejercicio18

Redefinir (usando lazy val) toString en Duracion para que se muestre con el formato horas:minutos, por ejemplo 10:09. Emplear como patrón de formato %02d (dos dígitos, rellenar por la izquierda)