Cenecmálaga

Examen práctico 1º trimestre

Módulo: Programación multimedia y dispositivos móviles

kotlin

* *Leer todo el contenido de este documento, antes de empezar.*
* *Será obligatorio realizar todos los ejercicios, dejar alguno en blanco, supondrá un punto menos por cada pregunta en blanco.*
* *Será obligatorio comentar el código, los comentarios se realizarán en aquellas partes que se considere necesarios, solo por bloques, por ejemplo, bloque de definición de variables, etc. NO REALIZARLO LÍNEA POR LÍNEA, solo las partes más importantes.*
* *Los nombres de las variables, métodos y clases que aparecen en las instrucciones son meramente de ayuda, se podrán dar otros nombres.*
* *Los ejercicios deberán ejecutarse sin ningún error, si el código da error, se restará un punto por cada ejercicio que de error.*
* *Se valorará cualquier mejora o ampliación que quiera realizar el alumno basado en el ejercicio propuesto.*
* *Cada ejercicio se realizará en una clase diferente, se denominará: por ejemplo Ejercicio1Array ,Ejercicio2funciones, Ejercicio3herencia, Ejercicio4interfaces, Ejercicio5colecciones.*
* *Se podrá enviar el paquete que contenga los ejercicios, o bien comprimidos los ejercicios y el paquete en un fichero que se denomine ExamenPractico+ el nombre y apellidos del alumno por ejemplo:*

*ExamenPracticoAlbertoGonzalez.zip*

* *No se permite el uso de internet ni la ayuda de código de los ejercicios de clase.*

**EJERCICIO 1: ARRAYS (1,5 PUNTO)**

Se desea realizar un programa en el lenguaje de programación Kotlin que gestione información sobre empleados de una empresa. El programa almacenará datos como nombres y salarios base en una matriz bidimensional.

**Instrucciones:**

**Mostrar Nombres:**

* Extrae la primera fila de la matriz, que contiene los nombres de los empleados, y muestra los nombres separados por comas. Utiliza la función joinToString para formatear la salida.

**Calcular Nota Media:**

* Extrae la segunda fila de la matriz, que contiene los salarios base de los empleados, y muestra los salarios base separados por comas. Utiliza la función joinToString para formatear la salida.
* Calcula y muestra la nota media de los salarios base de los empleados.
* Calcula y muestra la suma total de los salarios base.

**Resultado**

**Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente**

**EJERCICIO 2: FUNCIONES (2,5 PUNTOS)**

Deberás solicitar al usuario el número de partidas a jugar, permitirle elegir entre Piedra, Papel o Tijeras en cada partida, y determinar el ganador al final de todas las partidas.

**Instrucciones:**

* Solicita al usuario ingresar el número de partidas que desea jugar.

Inicializa las variables para contar el número de partidas ganadas por el usuario y la máquina.

Jugadas del Usuario y de la Máquina:

* En cada partida, permite al usuario elegir entre "piedra", "papel" o "tijeras".

Genera de manera aleatoria la elección de la máquina (la máquina también puede elegir "piedra", "papel" o "tijeras").

Determinar el Ganador:

* Compara las elecciones del usuario y de la máquina para determinar el ganador de cada partida.

Muestra por consola el resultado de cada partida (quién ganó o si fue un empate).

Actualiza las variables de partidas ganadas por el usuario y la máquina.

Resultado Final:

* Al finalizar todas las partidas, muestra por consola el resultado final con el número total de partidas ganadas por el usuario y la máquina.

**Resultado**

Texto

Descripción generada automáticamente

**Nota**

* Puedes cambiar el menú por números en vez de escribir piedra, papel o tijeras.
* También se valorará si quieres añadir un menú de inicio, 1 jugar 2 salir

**EJERCICIO 3: HERENCIA (2,5 PUNTOS)**

Diseñar un sistema de información deportiva utilizando herencia en Kotlin. Se trabajará con una clase base Deporte y cuatro clases hijas: Futbol, Baloncesto, Balonmano y Voleibol. Cada clase hija representará un deporte específico y heredará propiedades de la clase base.

**Instrucciones:**

**1º Clase Padre - Deporte:**

Crea una clase base llamada Deporte con propiedades nombre y tipoTerreno. Implementa un método llamado mostrarDetalles() que imprima información básica sobre el deporte.

**2º Clases Hijas - Deportes Específicos:**

**Crea cuatro clases hijas: Futbol, Baloncesto, Balonmano y Voleibol.**

Cada clase hija debe tener propiedades específicas que la distingan del resto. De esta forma:

Clase Futbol:

* numeroJugadores: int
* esCampoGrande Boolean

clase Baloncesto:

* alturaCanasta: Double
* esDeporteEquipo: Boolean

clase Balonmano:

* esDeporteOlimpico: Boolean
* esContacto: Boolean

clase Voleibol:

* esDeportePlaya: Boolean
* numeroJugadoresEquipo: Int

**Método mostrarDetalles():**

En cada clase hija, sobrescribe el método mostrarDetalles() de la clase padre para mostrar información específica del deporte.

**3º Creación de Instancias y Mostrar Detalles:**

En la función main(), crea instancias de cada deporte (Fútbol, Baloncesto, Balonmano y Voleibol).

Llama al método mostrarDetalles() de cada instancia para mostrar la información detallada de cada deporte.

**Resultado**

Texto

Descripción generada automáticamente

**Nota**

El tipo Boolean es igual que en Java

**EJERCICIO 4: INTERFACES (2,5 PUNTOS)**

**Desarrollo de Interfaces para Videojuegos**

Se utilizarán tres interfaces: Jugable, SonidoAmbiente y GuardadoPartida. Deberás crear clases concretas para representar diferentes tipos de videojuegos que implementen estas interfaces.

**Instrucciones:**

**Interfaces:**

* Crea una interfaz llamada Jugable que contenga un método jugar(nombreUsuario: String) para representar la acción de jugar un videojuego. El método debe aceptar un argumento adicional que represente el nombre del usuario que está jugando.
* Crea una interfaz llamada SonidoAmbiente que contenga un método reproducirSonido() para simular el sonido ambiente del videojuego.
* Crea una interfaz llamada GuardadoPartida que contenga un método guardarPartida() para modelar la capacidad de guardar el progreso del juego.

**Clases Concretas:**

* Implementa tres clases concretas que representen diferentes tipos de videojuegos. Ejemplos de clases pueden ser Aventura, Deportes y Estrategia.
* Cada clase debe heredar de las interfaces Jugable, SonidoAmbiente y GuardadoPartida, implementando los métodos correspondientes.

**Demostración en la Función main():**

* En la función main(), crea instancias de cada tipo de videojuego y demuestra el uso de los métodos definidos en las interfaces.
* Llama al método jugar(nombreUsuario: String) para iniciar el juego proporcionando un nombre de usuario.
* Llama al método reproducirSonido() para simular el sonido ambiente del videojuego.
* Llama al método guardarPartida() para guardar el progreso del juego.

**Solución**

Texto

Descripción generada automáticamente

**EJERCICIO 5: COLECCIONES (1 PUNTO)**

Crear un programa en Kotlin que gestione información sobre asignaturas, nombres y notas de estudiantes utilizando una lista mutable llamada centro. El programa debe realizar las siguientes operaciones:

**Mostrar Lista Original:**

Declarar una función llamada mostrar que acepte una lista de elementos de tipo Any como parámetro e imprima cada elemento en la consola.

**Modificar Asignaturas:**

* En el programa principal (main), crear una lista mutable llamada centro que inicialmente contenga información sobre asignaturas, nombres y notas de estudiantes. La lista debe ser de tipo MutableList<Any>.
* Imprimir en la consola la lista original utilizando la función mostrar.
* Modificar dos asignaturas en la lista ("MATEMATICAS" por "HISTORIA" y "LENGUA" por "FÍSICA") utilizando el índice correspondiente.
* Imprimir en la consola la lista después de la modificación utilizando la función mostrar.

**Añadir Nuevos Elementos:**

* Añadir tres nuevos elementos a la lista centro que representen una nueva asignatura "QUÍMICA", un nuevo estudiante "Cristina", y una nueva nota 7.
* Imprimir en la consola la lista después de añadir los nuevos elementos utilizando la función mostrar.

**Instrucciones Adicionales:**

* Utiliza la función mostrar para imprimir el contenido de la lista en diferentes puntos del programa.
* Asegúrate de gestionar correctamente las operaciones de modificación y adición en la lista centro.
* Puedes ajustar el código según tus necesidades y preferencias.

**Resultado**

Texto

Descripción generada automáticamente