Programación
Unidad 5: POO avanzado

Realizar los programas siguientes en lenguaje JAVA. A Moodle subirás un fichero zip con el código del fichero java correspondiente o el enlace GitHub público.

1. Realiza un programa que lee un fichero java indicado por teclado y que tiene comentarios indicados con //. Se quiere sobre escribir el mismo archivo de salida con el mismo código funcional de java pero sin los comentarios, para ello utilizarás un ArrayList<String> donde guardarás cada línea que vas leyendo y que luego pasarás para escribir. Sólo se borrarán los que tienen el comentario al principio de línea y hay que comprobar si el archivo que pasas existe o no. **(2 puntos)**
Ejemplo:

```
// Comentario de cabecera
public class Ejemplo {

   public static void main(String[] args) {
      System.out.println("Hola"); // este comentario NO se borra
   }

   // Otro comentario
}
```

Tras ejecutar vuestro programa quedará como este y este fichero java resultado funcionará igual:

```
public class Ejemplo {

   public static void main(String[] args) {
      System.out.println("Hola"); // este comentario NO se borra
   }

}
```

2. Procesa el siguiente JSON para que en un menú muestre: **(2,5 puntos)**

1- Número de empleados activos y cómo se llaman
2- Número de empleados que tienen rol administrador
3- Imprime los datos de la empresa (Nombre y dirección)
4- Pregunte por un empleado y diga sus datos o muestre que no es empleado

```
{
  "empresa": "TechCorp",
  "empleados": [
    {
      "id": 1,
      "nombre": "Laura",
      "roles": ["admin", "backend"],
      "activo": true
    },
    {
      "id": 2,
      "nombre": "Pedro",
      "roles": ["frontend"],
      "activo": false
```

```
    },
    {
      "id": 3,
      "nombre": "Juan",
      "roles": ["frontend","admin"],
      "activo": true
    }
  ],
  "direccion": {
    "ciudad": "Barcelona",
    "pais": "España"
  }
}
```

3. Se desea desarrollar un programa en Java que permita buscar sinónimos en español a partir de un diccionario español-inglés. Para ello, se utilizará una estructura de datos HashMap<String, String> donde: **(2 puntos)**

- La clave será una palabra en español.
- El valor será su traducción al inglés.

El programa irá pidiendo palabras en español y dará la palabra en inglés si está o no en otro caso o Salir en otro caso para finalizar la ejecución.
Contenido básico del diccionario:

Español	caliente	rojo	ardiente	verde	agujetas	abrasador	hierro	grande
Inglés	hot	red	hot	green	stiff	hot	iron	big

Ejemplo: Si por teclado le escribes “caliente”, te dará: “En inglés la palabra es hot”. Si le escribes “oro”, te dará: “Palabra no encontrada en el diccionario”

4. Se desea desarrollar una aplicación en Java para la gestión de un parking.
En el parking pueden estacionar distintos tipos de vehículos y todos ellos comparten una serie de comportamientos comunes. **(3 puntos)**

- Habrá una interfaz Parkeable, que tendrá el atributo de plazas=50 y las funciones entrar, salir y calcularPrecio(horas)
- Habrá una clase abstracta Vehiculo que implementa esta interfaz con el atributo matrícula y un atributo static de plazas ocupadas. En las funciones de la interfaz se mostrará “El vehículo con matrícula xxxxx ha entrado/salido del parking” e irá incrementando o decrementando las plazas ocupadas. El método de calcularPrecio(horas) quedará como abstracto y habrá un método que devuelva el número de plazas libres.
- Habrá una clase Coche y Moto que tendrán por atributo una constante precio_hora con 2€ para el coche y 1€ para la moto. Estas clases heredarán de Vehiculo y al implementar calcularPrecio dirán el número de horas y calcularán el precio como horas*precio_hora.
- Haz una clase que haga de main donde los coches y motos van entrando y saliendo del parking y vaya mostrando el estado del parking y lo que pagan al salir. Genera las horas que han estado de manera aleatoria.

5. ¿Cuándo puede interesarnos crear una clase abstracta? ¿Y una clase interfaz? **(0,5 puntos)**