1. **Control de Excepciones**

Crea una clase *Utilidad*, donde crees un método estático llamado *maximo* que pida 6 números por teclado y nos devuelva cuál es el máximo. Si el usuario introduce un dato erróneo (algo que no sea un número entero) el programa debe indicarlo mostrando un mensaje por consola y debe pedir de nuevo el número.

Prueba la funcionalidad anteriormente implementada desde un programa Principal.

1. **Libro**

Crear una clase Libro que contenga los siguientes atributos:

* ISBN
* Titulo
* Autor
* Número de páginas
* Editorial

Crear sus respectivos métodos get/set correspondientes para cada atributo. Crear el método toString() para mostrar la información relativa al libro con el siguiente formato: «El libro con ISBN creado por el autor XXXX tiene XXXX páginas»

Crear una clase *Principal*, donde en su método *main*, crear 2 objetos Libro (los valores que se quieran) y mostrarlos por pantalla.

Indicar cuál de los 2 tiene más páginas, para lo cual, crea un método en la clase *Libro* que te permita compararlo con otro y muestre por consola un mensaje indicando cuál de los dos tiene más páginas siguiendo el siguiente formato “El libro con ISBN xxxx tiene más páginas que el libro con ISBN xxxx”.

1. **Videoclub**

Desarrolla un programa que permita gestionar un videoclub.

Al iniciar la aplicación se nos pedirán los siguientes datos para crear un Cliente:

* Nombre
* Apellidos
* Dni
* Teléfono
* Sexo (hombre, mujer)
* Edad
* TienePeliculaAlquilada (si o no)

Mediante un menú se podrán realizar determinadas operaciones:

1. Alquilar película. Para alquilar una película se deberá pedir la siguiente información sobre la Película a alquilar (título, director, año de publicación, género (terror, romántica, ficción), precio de alquiler). El cliente podrá alquilar una película si tiene saldo suficiente. Al alquilar la película, se disminuye el saldo correspondiente al precio de alquiler de dicha película. Pone el atributo peliculaAlquilada a true.
2. Devolver. Pone el atributo peliculaAlquilada a false
3. Aumentar saldo del cliente
4. Cambiar teléfono del cliente
5. esMayorDeEdad: Devuelve un valor lógico indicando si es mayor de edad.
6. Ver datos del cliente (nombre, apellidos, dni, teléfono y saldo)
7. Salir de la aplicación

Además del programa *Principal* de la aplicación (clase con un método *main*), habrá que escribir las clases necesarias que proporcionen las herramientas necesarias para trabajar con dicho programa (Pelicula y Videoclub).

NOTA: Controla las posibles excepciones que se produzcan.

1. **Cafetera**

Desarrolla una clase *Cafetera* con atributos capacidadMaxima (la cantidad máxima de café que puede contener la cafetera) y cantidadActual (la cantidad actual de café que hay en la cafetera). Implementa, al menos, los siguientes métodos:

* Constructor vacio: establece la capacidad máxima en 1000 (c.c.) y la actual en cero (cafetera vacía).
* Constructor con la capacidad máxima de la cafetera; inicializa la cantidad actual de café igual a la capacidad máxima.
* Constructor con la capacidad máxima y la cantidad actual. Si la cantidad actual es mayor que la capacidad máxima de la cafetera, la ajustará al máximo.
* llenarCafetera(): hace que la cantidad actual sea igual a la capacidad máxima
* servirTaza(int): simula la acción de servir una taza con la capacidad indicada. Si la cantidad actual de café “no alcanza” para llenar la taza, se sirve lo que quede.
* vaciarCafetera(): pone la cantidad de café actual en cero.
* agregarCafe(int): añade a la cafetera la cantidad de café indicada.
* volcarCafe(Cafetera) : vuelca el café de la cafetera pasada como parámetro, a la cafetera actual. Si la cantidad de la cafetera es mayor a la cantidad que se puede añadir, solo se llena lo que coja. Se actualizará la cantidad de café de la cafetera pasada por parámetro, restando de su capacidad actual la cantidad que se ha volcado.

1. **Automoción**

Crea las siguientes clases (cada una en su archivo) con los métodos que te parezcan adecuados:

* Motor: con métodos para arrancar el motor y apagarlo.
* Rueda: con métodos para inflar la rueda y desinflarla.
* Ventana: con métodos para abrirla y cerrarla.
* Puerta: con una Ventana y métodos para abrir la puerta y cerrar la puerta
* Coche: con un Motor, cuatro Ruedas y dos Puertas y métodos que arranque el coche, cambie la rueda delantera izquierda, abra las ventanillas etc…

Crea una clase Principal, que cree un Coche y luego por ejemplo arranque el coche etc..

1. **Empleado**

Crear una clase *Empleado* que modele la información que una empresa mantiene sobre cada empleado: DNI, sueldo base, pago por hora extra, horas extra realizadas en el mes, tipo de IRPF, casado o no y número de hijos.

Al crear los empleados se podrá proporcionar, si se quiere, el número de DNI. Todos los empleados tienen el mismo sueldo base 1200€. Además, crea los siguientes métodos:

* calcularHorasExtra: el cual, deberá calcular el complemento correspondiente a las horas extras realizadas y devolver dicho importe como dato devuelto del método, teniendo en cuenta que cada hora extra se paga a 16€.
* calularSueldoBruto: el cual deberá calcular y devolver el sueldo bruto del empleado (sueldo base + pago horas extra).
* calcularRetencionIRPF: el cual, parte del tipo de IRPF normal que es un 19%. Este método deberá calcular el importe de IRPF, teniendo en cuenta que el porcentaje de retención que hay que aplicar es el tipo, menos 2 puntos si el empleado está casado y menos 1 punto por cada hijo que tenga; el porcentaje se aplica sobre todo el sueldo bruto.
* printInfoBaseEmpleado: visualización de la información básica del empleado.
* printAllEmpleado: visualización de toda la información del empleado. La básica más el sueldo base, el complemento por horas extra, el sueldo bruto, la retención de IRPF y el sueldo neto.
* copia: Crea un empleado nuevo con la misma información que el pasado por parámetro.
* ganaMasQue: El cual recibe otro empleado y compara su sueldo bruto con el empleado actual. Devuelve verdadero o falso, en función de si el empleado actual gana mas o menos que el empleado que recibe por parámetro.
* métodos get/set necesarios.

1. **Cancion**

Desarrolla una clase *Cancion* con los siguientes atributos:

* titulo: una variable String que guarda el título de la canción.
* autor: una variable String que guarda el autor de la canción.
* duracion: una variable int que guarda la duración en segundos de la canción.

y los siguientes métodos:

* Cancion(String, String): constructor que recibe como parámetros el título y el autor de la canción (por este orden).
* Cancion(): constructor predeterminado que inicializa el título y el autor a cadenas vacías.
* dameTitulo(): devuelve el título de la canción
* dameAutor(): devuelve el autor de la canción.
* ponTitulo(String): establece el título de la canción.
* ponAutor(String): establece el autor de la canción.
* ponDuracion(int): estable la duración de la canción
* acortar(int): acorta la duración de la canción en los segundos indicados, siempre que sea posible. Si la canción es mas corta, no se hará nada. Mostrar mensaje por consola indicando que se hace.
* copia(Cancion): Devuelve una nueva canción, con el mismo título, autor y duración que la pasada por parámetro
* cancionMasLarga(Cancion): Devuelve la canción cuya duración es mayor.
* cancionMasCorta(Cancion): Devuelve la canción cuya duración es menor.

1. **Electrodoméstico**

Crear una clase llamada *Electrodomestico* con las siguientes características:

* Sus atributos son precio base, color, consumo energético (letras entre A y F) y peso.
* Por defecto, el color será blanco, el consumo energético será F, el precioBase es de 100 € y el peso de 5 kg. Usa constantes para ello.
* Los colores disponibles son: blanco, negro, rojo, azul y gris.
* Los constructores que se implementarán serán
  + Un constructor por defecto.
  + Un constructor con el precio y peso. El resto por defecto.
  + Un constructor con todos los atributos.
* Los métodos que implementara serán:
  + Métodos get/set (solo los necesarios)
  + comprobarConsumoEnergetico(char letra): comprueba que la letra es una de las letras posibles correspondientes a los distintos tipos de consumo, sino es correcta usara la letra por defecto. Se invocará en el constructor que permite introducir un consumo y no será visible.
  + comprobarColor(String color): comprueba que el color es uno de los posibles colores, sino lo es, usa el color por defecto. Se invocará en el constructor que permite introducir un color y no será visible.
  + precioFinal(): según el consumo energético, aumentara su precio, y según su tamaño, también. Utiliza las siguientes tablas:

| Letra | Precio |
| --- | --- |
| A | 100€ |
| B | 80€ |
| C | 60€ |
| D | 50€ |
| E | 30€ |
| F | 10€ |
| Tamaño | Precio |
| Entre 0 y 19 kg | 10€ |
| Entre 20 y 49 kg | 50€ |
| Entre 50 y 79 kg | 80€ |
| Mayor que 80 kg | 100€ |

1. **Paquetes**

Crear una clase llamada *Galleta* dentro del paquete *comida.postre* . Esta clase solo tendrá el *constructor* por defecto que muestra por consola el mensaje “Galleta creada”, y un método protected llamado *morder* que muestra por consola el mensaje “Galleta mordida”.

Crea una clase llamada *Cena* dentro del paquete *comida.cena*. Esta será una clase con un método *main* donde quieren comer galletas, pero no pueden. ¿Por qué no pueden comer galletas? ¿Qué hay que hacer para permitirlo?

1. **Botella y Sacacorchos**

a) Escribir una clase *Corcho*, que contenga un atributo bodega (cadena de caracteres con el nombre de la bodega).

b) Escribir una clase *Botella* que contenga un atributo corcho con una referencia al corcho que la tapa, o null si está destapada. Cuando se crea una botella, tiene el corcho puesto por defecto, por lo que es necesario indicar la bodega en el constructor de la botella.

c) Escribir una clase *Sacacorchos* que tenga un método *destapar* que le reciba una botella, cuyo comportamiento es sacar el corcho de la botella (solo en el caso que la botella no esté ya destapada).

Crear una clase *Principal*, con un método *main*, desde el cual crear una botella. Después usa la clase *Sacacorchos* para destapar la botella creada.

Muestra mensajes por consola que vayan indicando las acciones que se van produciendo.

1. **Mate**

Crea la clase *Mate* que describa el funcionamiento de la conocida bebida. La clase debe contener un atributo para la cantidad de cebadas que hay en el recipiente (representada por un número), y otro atributo para el estado (lleno o vacío). Crear los siguientes métodos:

* El *constructor* por defecto. Cuando se crea, su estado será vacío.
* Métodos get/set necesarios.
* Un método *cebar*, que llena el mate con la cantidad de cebadas que se introduzca por parámetro. Al cebar, el estado del mate pasará a lleno. Si se intenta cebar con el mate lleno, se debe mostrar el mensaje: “Çuidado! Te quemaste!".
* Un método *beber*, que resta una cebada disponible. Si se intenta beber un mate vacío, se debe imprimir el mensaje: “El mate está vacío!". El mate pasará a vacío, si la cantidad de cebadas es 0.
* Un método vaciar, que lave el recipiente y por tanto ponga el número de cebadas a 0 y su estado a vacío. Además, se debe mostrar el mensaje “El mate está lavado”.
* Un método migrar(Mate), que pase el contenido de un mate a otro siempre que no esté vacio.

Crear una clase *Principal* donde en su método *main*, creará al menos un par de mates, prueba todas las posibles opciones entre ellos.