1. Crear e implementar clases `Rueda`, `Motor`, `Puerta`, `Ventanilla`, `Auto` y `Vehiculo` con métodos y atributos apropiados. Discutir posibilidades de implementación vía composición y/o herencia.
2. Crear una jerarquía de 4 clases, 3 derivadas y una embebida. Dotarlas de destructores y constructores que se anuncien en `cout`. Implementar un `main` utilizando objetos de estas clases y verificar orden de llamado de constructores y destructores. orden
3. Crear dos clases llamadas `Traveler` y `Pager` con constructores que reciben como argumento un `string` que se copia a un atributo privado. Generar constructores y operadores de copia y move para cada una. Crear una clase `BusinessTraveler` que herede de `Traveler` y contenga un objeto `Pager` como atributo. Generarle un constructor default, un constructor que reciba 2 `string`s, constructores y operadores de copy y move. Hacer un programa para probar lo implementado. move =?
4. Crear una clase `SpaceShip` con un método `fly()`. Crear una clase `Shuttle` derivada de `SpaceShip` con un método `land()`. Crear un objeto Shuttle, hacer un upcast a puntero o referencia a `SpaceShip` y llamar al método `land()` sobre el upcast. Explicar.
5. Crear una clase `Cosa` con constructor default, constructores y operadores de copy y move, y destructor cuyas implementaciones se anuncien en `cout`. Hacer un programa donde haya un `vector<Cosa>`, agregar varias `Cosa`s y explicar el output. Repetir con un `vector<Cosa *>`.

1. Class Auto: public Vehículo {
public:
Motor motor ;
Rueda ruedas[4];
Puerta puertas[4];
Ventanilla venta[4];
}

4. No funciona ya que al hacer el upcast a SpaceShip pierdo las funciones nuevas de Shuttle (land())