1.

Employee объявлен с модификатором public, а класс Person, от которого он наследуется internal.

2.

Конструктор Employee принимает параметры: строку name и строку company. Конструктор Person: один принимает строку name, а другой — строку name и цело число age.

Sealed

3.

Класс Auto содержит Seats (количество сидений в автомобиле)

Класс Truck является наследником класса Auto и наследует все его свойства и методы. В классе Truck новое свойство Capacity (грузоподъемность грузовика).

В методе Main создается объект класса Truck, в конструктор передаются значения для количества сидений и грузоподъемности.

Программа выводит на консоль значение свойства Capacity созданного экземпляра Truck выведено: Грузовик с грузоподъемностью 1.1 тонн.

4.

При создании объекта Truck сначала вызывается конструктор без параметров класса Auto, тк как Truck наследуется от Auto, и выводится сообщение "Auto has been created". Затем выполняется конструктор класса Truck, который устанавливает количество сидений равным 2 и грузоподъемность равной переданному значению после выводится сообщение "Truck has been created. В методе Main отображается информация о грузоподъемности грузовика с использованием свойства Capacity выведено:Auto has been created Truck has been created Truck with capacity 1.1

5.

В классе Person есть свойство Name, изначально равно "Ben". В конструкторе класса Person это свойство устанавливается в "Tim". В конструкторе класса Employee вызывается конструктор класса Person с аргументом "Bob", и свойство Name устанавливается в "Bob". При создании объекта emp свойство Name устанавливается в "Sam".

Создается объект Person с именем "Tom", и происходит попытка приведения этого объекта к типу Employee с помощью оператора as. Поскольку tom является экземпляром Person, а не Employee, переменная empl будет равна null. После этого имя tom изменчется на "Bob". Когда программа пытается вывести empl.Name, возникает ошибка NullReferenceException, так как empl равно null и у него нет свойства Name.

6.

В результате выполнения будет выведено null.

Создается объект Employee с именем "Tom", который является экземпляром Employee, но ссылается на переменную типа Person. Переменная bob ссылается на тот же объект, что и tom. Происходит приведение bob к типу Employee, и переменная empl ссылается на тот же объект, что и tom. Имя tom изменяется на "Bob". Когда программа выводит empl.Name, она обращается к свойству Name объекта Employee, которое имеет значение "Bob". В результате выполнения следующей программы будет выведено "Bob"

Оператор as безопасен, так как он возвращает null, если приведение невозможно

Программа выведет null, поскольку tom является объектом класса Person, а не Employee. При попытке выполнить приведение типа as к Employee, переменная empl будет равна null

Программа выведет "Bob". В этом случае tom является объектом класса Employee, и при присваивании bob = tom обе переменные ссылаются на один и тот же объект. Когда мы изменяем tom.Name на "Bob", это изменение также отразится в empl, так как они ссылаются на один и тот же объект

Рекомендуется использовать оператор as, так как этот способ безопасен. Если tom не является объектом типа Employee, empl1 будет равен null, и программа не вызовет исключение

Программа выведет "Track is moving", потому что метод Move переопределен в классе Track. При вызове auto.Move() будет вызван метод Move из класса Track, так как auto ссылается на объект типа Track

Модификатор sealed предотвращает дальнейшее наследование от класса

Программа выведет "Auto". Метод Display в классе Auto помечен как sealed, что означает, что он не может быть переопределен в производных классах. В методе DisplayInfo мы вызываем base.Display(), что вызывает метод Display из класса Auto и выводит соответствующее сообщение