1. **(25P).** Implementar una Clase Automóvil, dentro de una jerarquía de clases. Considere que, además de ser un Vehículo, un automóvil es también una comodidad, un modo de transporte. Además, declare e implemente una interfaz con el comportamiento común de los Automóviles. En la jerarquía tenga en cuenta al menos 5 clases, al final, desarrolle una clase AppAuto que implemente el método main que permita probar la clase Automóvil.

2. **(25P).** Descargue del classroom el archivo de nombre lab2_herencia.zip, dentro del tema ejemplos, luego implemente una clase del tipo aplicación PruebaEmpleadoPorComision y otra PruebaEmpleado-BaseMasComision para asegurar que las clases EmpleadoPorComision y EmpleadoBaseMasComision (contenidas en dicho archivo de formato .zip) funcionen efectivamente. Usted puede pasar parámetros "duros", de su elección, al constructor para realizar las pruebas. Debe utilizar todos los métodos implementados en las clases a fin de que pueda obtener un resultado similar al siguiente;

```
Información del Empleado:
=========================
Nombre: Juan
Apellido: Perez
N° de cédula: 1234567
Las ventas brutas son: 5000000.00
La comisión es: 0.05
El salario base es: 3000000.00
Su ingreso total:  3250000.00
```

3. **(25P).** En este ejercicio, creará una superclase Empleado más general que extraiga los atributos y comportamientos de la clase EmpleadoPorComision que son comunes para todos los objetos Empleado. Los atributos y comportamientos comunes de todos los objetos Empleado son:

- primerNombre,
- apellidoPaterno,
- numeroCedulal,
- obtenerPrimerNombre,
- obtenerApellidoPaterno,
- obtenerNumeroCedula y una parte del método toString.

Cree una nueva superclase Empleado que contenga estas variables y métodos de instancia, además de un constructor. A continuación, vuelva a escribir la clase EmpleadoPorComision (ejemplo entregado por el profesor) como una subclase de Empleado. La clase EmpleadoPorComision debe contener sólo las variables y métodos de instancia que no se declaren en la superclase Empleado. El constructor de la clase EmpleadoPorComision debe invocar al constructor de la clase Empleado y el método toString de EmpleadoPorComision debe invocar al método toString de Empleado.

Una vez que complete estas modificaciones, vuelva a verificar y probar las aplicaciones PruebaEmpleadoPorComision y otra PruebaEmpleado-BaseMasComision para asegurar que las mismas sigan funcionando para tales objetos.

4. **(25P).** Otros tipos de objetos Empleado podrían incluir objetos EmpleadoAsalariado que reciban un salario semanal fijo, TrabajadoresPiezas a quienes se les pague por el número de piezas que produzcan, o EmpleadosPorHoras que reciban un sueldo por horas con tiempo y medio (1.5 veces el sueldo por horas) por las horas trabajadas que sobrepasen las 40 horas.

Cree la clase EmpleadoPorHoras que herede de la clase Empleado y tenga la variable de instancia horas(de tipo double) que represente las horas trabajadas, la variable de instancia sueldo(de tipo double) que represente los sueldos por hora, un constructor que reciba como argumentos el primer nombre, el apellido paterno, el número de seguro social, el sueldo por horas y el número de horas trabajadas, métodos establecer y obtener para manipular las variables hora y sueldo, un método ingresos para calcular los ingresos de un EmpleadoPorHoras con base en las horas trabajadas, y un método toString que devuelva la representación String del EmpleadoPorHoras. El método establecerSueldo debe asegurarse de que sueldo sea mayor o igual a 0, y establecerHoras debe asegurar que el valor de horas esté entre 0 y 168 (el número total de horas en una semana). Use la clase EmpleadoPorHoras en un programa de prueba que sea similar al ejercicio anterior.