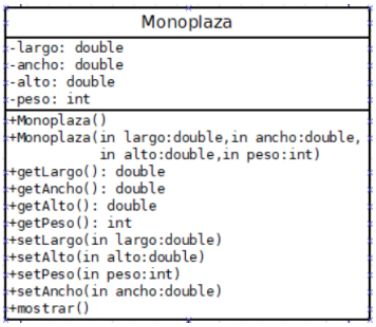
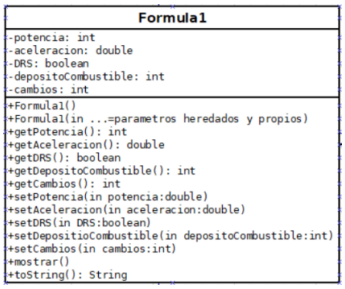
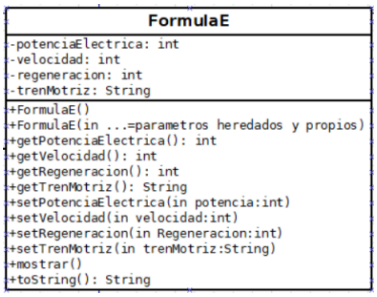
Ejercicio 9 - Jerarquía de clases, herencia y polimorfismo

Tenemos las siguientes clases y su relación de herencia. Se representan las clases y la relación de herencia mediante diagramas UML:







Hay que realizar las siguientes tareas:

- Implementar la jerarquía de clases.

  ¿Hay alguna clase abstracta? Argumenta la respuesta de tu implementación.

  Sí, la clase `Monoplaza` con un método abstracto `mostrar`.

- Constructores:
  - Vació.
  - Parametrizado. Los constructores parametrizados de las clases derivadas reciben todos los parámetros.
- Getters/setters: los indicados en el diagrama UML.
- Método mostrar: muestra el contenido de la instancia del objeto. Se indica el nombre del atributo y el valor. Hay que imprimir en pantalla: una cabecera (Formula 1 o Formula E) y un cuerpo de datos (un atributo por línea).
- toString: redefinición del método heredado de la clase Object, para devolver un string con el contenido del objeto concreto. Este método está incluido en ambas clases derivadas y debe ser implementado en cada clase. Una cabecera (Formula 1 o Formula E) y un cuerpo de datos (un atributo por línea).
- Clase `Flow` que contenga el método `main`. Se realiza lo siguiente:

- Declarar un objeto de la clase Monoplaza. Crear un objeto de la clase Formula1 y otro objeto de la clase Formula E. Los datos a contener en cada objeto se muestran en las imágenes que aparecen más abajo.
- Mostrar el contenido de los objetos derivados.
- Asignar el objeto creado de Formula1 al objeto declarado de la clase base y llamar al método mostrar

  ¿Qué ocurre?

  Asigna la referencia del objeto de la clase `Formula1` a la variable de la clase `Monoplaza`.

  ¿Que código se ejecuta?

  Se ejecuta el método `mostrar()` de la clase `Formula1`.

  ¿Que explicación encuentras?

  Es posible gracias al polimorfismo, que permite que una variable de tipo base se comporte como un objeto de una clase derivada.

- Hacer lo mismo con el otro objeto.

  ¿Qué sucede?

  El valor del objeto de la clase `Formula1` se asigna a la variable de la clase `Monoplaza`.

  ¿Que explicación tiene?

  Lo mismo que pasaba anteriormente, la variable de la clase `Monoplaza` obtiene el valor del objeto de la clase `FormulaE` gracias al polimorfismo.

Datos de las instancias creadas:

Características	Fórmula 1
Dimensiones: largo	5,6-5,7 metros
Dimensiones: ancho	+/- 2 metros
Altura	0,95 metros
Peso	746 kg con piloto
Motor	V6 turbo 1,6 litros – 880-1000 CV a 15.000 rpm + ERS (163 CV durante 33,3 segundos por vuelta)
Aceleración (0 a 100 km/h)	2,4 segundos
DRS	Sí
Depósito de combustible	110 litros
Caja de cambios	Secuencial de 8 velocidades
Ruedas	13 pulgadas
Tiempo de pole en Montmeló en 2020	1:15.584

Longitud	5,320 metros
Altura	1,050mm
Anchura	1,780mm
Distancia entre ejes	3,100 metros
Peso (incluído el piloto)	920kg (batería 450kg)
Potencia máxima	200kW
Regeneración máxima	100kW
Velocidad máxima	140mph
Tren motriz	Trasero
Neumáticos	Michelin

```
class Flow {

    public static void main(String[] args) {

        Monoplaza monoplaza;

        Formula1 williams = new Formula1(5.65, 2, 0.95, 746,
        FormulaE jaguar = new FormulaE(5.32, 1.78, 1.05, 920,

        williams.mostrar();
        jaguar.mostrar();

        monoplaza = williams;
        monoplaza.mostrar();

        monoplaza = jaguar;
        monoplaza.mostrar();

    }

}

abstract class Monoplaza {
```

```
    private double largo, ancho, alto;
    private int peso;

    Monoplaza() {
        this.largo = 0;
        this.ancho = 0;
        this.alto = 0;
        this.peso = 0;
    }

    Monoplaza(double largo, double ancho, double alto, int pe
        this.largo = largo;
        this.ancho = ancho;
        this.alto = alto;
        this.peso = peso;
    }

    public double getLargo() {
        return largo;
    }

    public double getAncho() {
        return ancho;
    }

    public double getAlto() {
        return alto;
    }

    public int getPeso() {
        return peso;
    }

    public void setLargo(double largo) {
        this.largo = largo;
    }

    public void setAncho(double ancho) {
```

```
        this.ancho = ancho;
    }

    public void setPeso(int peso) {
        this.peso = peso;
    }

    public void setAlto(double alto) {
        this.alto = alto;
    }

    public abstract void mostrar();

}

class Formula1 extends Monoplaza {

    private int potencia, depositoCombustible, cambios;
    private double aceleracion;
    private boolean drs;

    public Formula1() {
        super();
        this.potencia = 0;
        this.aceleracion = 0;
        this.drs = false;
        this.depositoCombustible = 0;
        this.cambios = 0;
    }

    public Formula1(double largo, double ancho, double alto,
            int depositoCombustible, int cambios) {
        super(largo, ancho, alto, peso);
        this.potencia = potencia;
        this.aceleracion = aceleracion;
        this.drs = drs;
        this.depositoCombustible = depositoCombustible;
        this.cambios = cambios;
```

```
    }

    public int getPotencia() {
        return potencia;
    }

    public double getAceleracion() {
        return aceleracion;
    }

    public boolean getDrs() {
        return drs;
    }

    public int getDepositoCombustible() {
        return depositoCombustible;
    }

    public int getCambios() {
        return cambios;
    }

    public void setPotencia(int potencia) {
        this.potencia = potencia;
    }

    public void setAceleracion(double aceleracion) {
        this.aceleracion = aceleracion;
    }

    public void setDrs(boolean drs) {
        this.drs = drs;
    }

    public void setDepositoCombustible(int depositoCombustibl
        this.depositoCombustible = depositoCombustible;
    }
```

```
    public void setCambios(int cambios) {
        this.cambios = cambios;
    }

    public void mostrar() {
        System.out.println(this.toString());
    }

    public String toString() {
        return "\n\n--------Formula 1--------\n\n\tLargo: " +
                + " metros\n\tAlto: " + super.getAlto() + " m
                + this.getPotencia() + " rpm\n\tAceleración (
                + " segundos\n\tDRS: " + (this.getDrs() ? "Sí
                + this.getDepositoCombustible() + " litros\n\
    }

}

class FormulaE extends Monoplaza {

    private int potenciaElectrica, velocidad, regeneracion;
    private String trenMotriz;

    public FormulaE() {
        super();
        this.potenciaElectrica = 0;
        this.velocidad = 0;
        this.regeneracion = 0;
        this.trenMotriz = "";
    }

    public FormulaE(double largo, double ancho, double alto,
            int regeneracion, String trenMotriz) {
        super(largo, ancho, alto, peso);
        this.potenciaElectrica = potenciaElectrica;
        this.velocidad = velocidad;
        this.regeneracion = regeneracion;
        this.trenMotriz = trenMotriz;
```

```
    }

    public int getPotenciaElectrica() {
        return potenciaElectrica;
    }

    public int getVelocidad() {
        return velocidad;
    }

    public int getRegeneracion() {
        return regeneracion;
    }

    public String getTrenMotriz() {
        return trenMotriz;
    }

    public void setPotenciaElectrica(int potenciaElectrica) {
        this.potenciaElectrica = potenciaElectrica;
    }

    public void setVelocidad(int velocidad) {
        this.velocidad = velocidad;
    }

    public void setRegeneracion(int regeneracion) {
        this.regeneracion = regeneracion;
    }

    public void setTrenMotriz(String trenMotriz) {
        this.trenMotriz = trenMotriz;
    }

    public void mostrar() {
        System.out.println(this.toString());
    }
```

```
    public String toString() {
        return "\n\n--------Formula E--------\n\n\tLargo: " +
                + " metros\n\tAlto: " + super.getAlto() + " m
                + " Kg\n\tPotencia máxima: " + this.getPotenc
                + this.getVelocidad() + " mph\n\tRegeneración
                + " kW\n\tTren motriz: " + this.getTrenMotriz
    }

}
```

¿Qué dificultades aparecen al implementar las clases? y ¿La jerarquía de clases?

Que la clase base tiene que ser `abstract` y las clases derivadas se tiene que extender de la clase base usando `extends`.

¿Cuantas consultas has realizado por tu cuenta para comprender esas dificultades?

No he realizado consultas.

¿Cuál es tu solución?

Asignar a la clase `Monoplaza` como abstracta y extender la clase `Formula1` y `FormulaE` con `extends`.