Introducción a la Programación Orientada a Objetos
DCIC - UNS

TERCER PARCIAL – 10 de junio de 2023

APELLIDO Y NOMBRE: ..... LU: .....

CANTIDAD DE HOJAS ENTREGADAS (SIN ENUNCIADO): 3

Importante: La interpretación del enunciado es parte del examen.

LA EVALUACIÓN SE REALIZARÁ CONSIDERANDO LA CORRECTITUD, LEGIBILIDAD Y EFICIENCIA DE LAS SOLUCIONES PROPUESTAS.

NO RESUELVA LOS EJERCICIOS EN EL ENUNCIADO. REALICE LOS PROBLEMAS EN HOJAS SEPARADAS.

Ejercicio 1

```
abstract class Alfa{
  protected int n;
  public Alfa(){
    n = 1;
  }
  public String a(){
    return "a en Alfa" + "n:" + n;
  }
  public String b(int x){
    return "b en Alfa" + "x:" + x + "n:"+n;
  }
  public String b(){
    return a()+ "b en Alfa" + "n:" + n;
  }
}
```

```
class Beta extends Alfa{
  private String t;
  public Beta(){
    t = "IPOO";
  }
  public String a(int x){
    return "a en Beta" + "x:" + x + "n:" + n + a();
  }
  public String b(String x){
    return "b en Beta" + "x:" + x + "t:" +t+"n:"+n;
  }
  public String c(){
    return a() + "c en Beta";
  }
}
```

```
class Gamma extends Alfa{
  private int t;
  public Gamma(){
    t = 10;
  }
  public String a(){
    return "a en Gamma" + "t:" + t + "n:" + n;
  }
  public String b(){
    return "b en Gamma" + "t:" + t + "n:" + n;
  }
  public String f(){
    String s = super.b();
    return s + "f() en Gamma";
  }
}
```

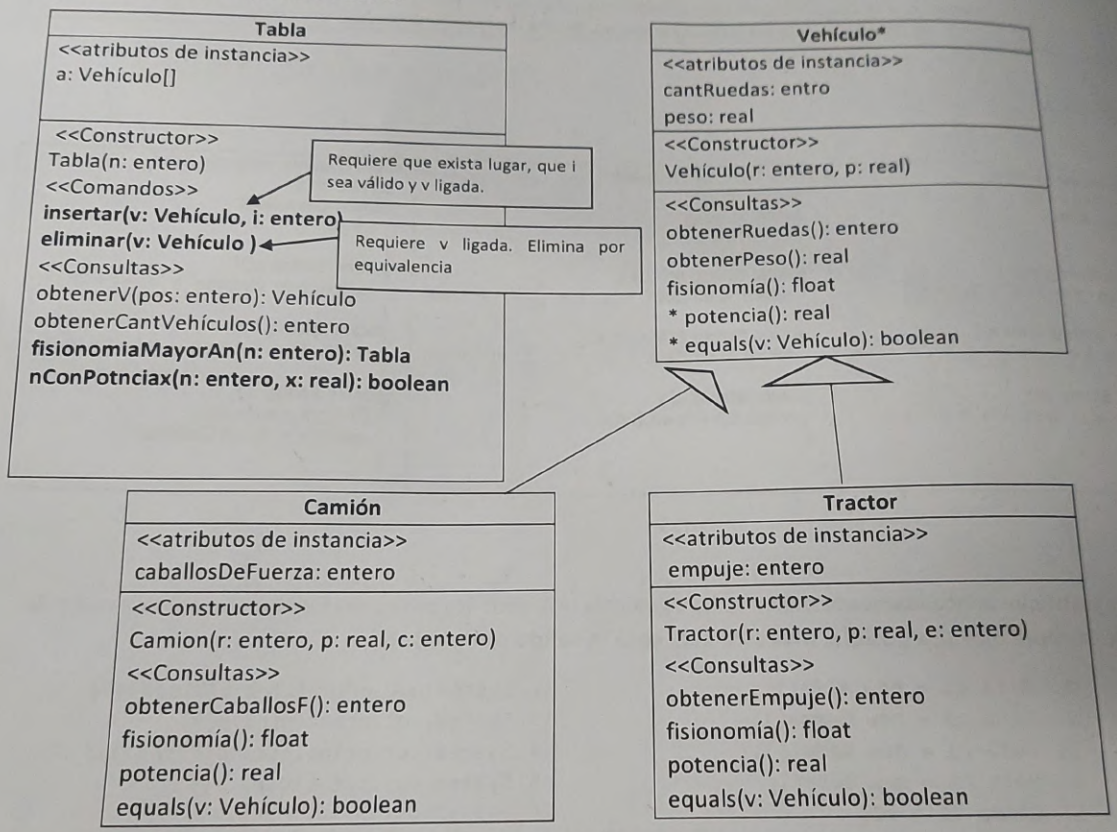
Indique y justifique adecuadamente cuáles de las siguientes instrucciones provocarán errores de compilación y de ejecución. Siempre que sea posible, **muestre cuál será la salida por consola**.

```
1.  Alfa c1 = new Alfa();
2.  Alfa c2 = new Gamma();
3.  Beta r1 = new Gamma();
4.  Beta r2 = new Beta();
5.  Gamma v1 = r2;
6.  Gamma v2 = new Gamma();
7.  Alfa c3 = new Beta();
8.  Beta r3 = null;
9.  System.out.println(v2.f());
10. System.out.println(v2.b("dos"));
11. System.out.println(c2.b(3));
12. System.out.println(r2.b("tres"));
13. System.out.println(r2.b());
14. System.out.println(c3.b("tres"));
15. System.out.println(r2.a(3));
16. System.out.println(r3.a(3));
17. System.out.println(c2.a());
```

ASUMIMOS QUE NADA VALE CERO.

Ejercicio 2:

Dado el siguiente diagrama de clases:

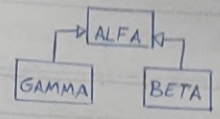


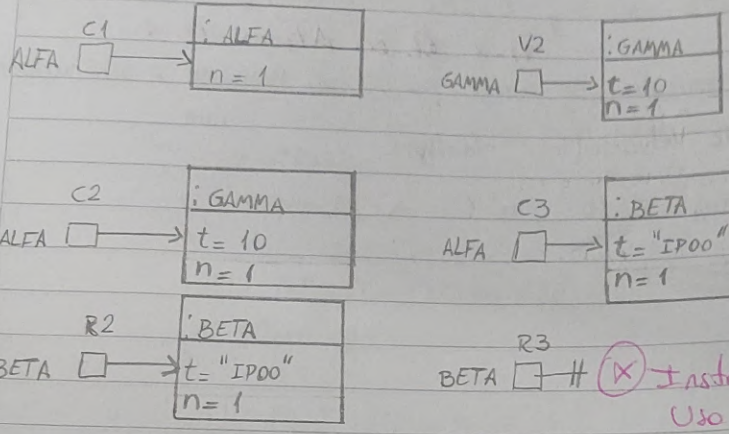
a) Implemente las clases Vehículo, Camión y Tractor completas. Tenga en cuenta que:
   - La fisionomía de cualquier vehículo se calcula como el peso dividido la cantidad de ruedas. En el caso de un camión la fisionomía se calcula como la de cualquier vehículo multiplicado los caballos de fuerza. Para los tractores el cálculo es completamente diferente: peso/empuje.
   - La potencia de un camión es su fisionomía por los caballos de fuerza. En el caso de un tractor su potencia es el empuje por la cantidad de ruedas.
   - Dos vehículos son equivalentes si son del mismo tipo y tienen los mismos valores en sus atributos de instancia.

b) Para la clase Tabla, implemente los siguientes métodos:
   I. **insertar(v: Vehiculo, i: entero)** Requiere que exista lugar en la tabla, *v* ligada y que el subíndice *i* sea válido (0 =< i< a.length).
   II. **eliminar(v: Vehiculo)** Requiere *v* ligado. Pueden existir en la tabla vehículos repetidos. Este método elimina todos los vehículos de la tabla que son equivalentes a *v*.
   III. **fisionomiaMayorAn(n: entero): Tabla** Retorna una nueva tabla con todos los vehículos que tengan una fisionomía mayor a *n*. Requiere *n* mayor a 0. En la tabla resultante los elementos ligados deben estar adelante y los nulos al final.
   IV. **nConPotnciax(n: entero, x: real): Boolean** Retorna verdadero si en la tabla existen exactamente *n* vehículos con potencia igual a *x*.

HOJA ①

EJERCICIO 1.





✗ Instrucción 1: Error Comp. Uso const. clase Abstracta

INSTRUCCIÓN 3. ERROR DE COMPILACIÓN. SE PRODUCE PORQUE A LA VARIABLE r1 DECLARADA DE TIPO BETA, SE LE INTENTA ASIGNAR UN NUEVO OBJETO DE TIPO GAMMA, Y AMBOS TIPOS TIENEN DISTINTA JERARQUÍA DE HERENCIA. ✓

INSTRUCCIÓN 5. ERROR DE COMPILACIÓN. SIMILAR AL CASO ANTERIOR, A LA VARIABLE v1 DECLARADA DE TIPO GAMMA, SE LE INTENTA ASIGNAR UNA REFERENCIA A UN OBJETO DE TIPO BETA. ✓

a en gamma t: 10 n: 1

INST. 9. a EN ALFA n: 1 ✗ b EN ALFA n: 1 f() EN GAMMA ✗

INST. 10. ERROR DE COMPILACIÓN. V2 ES DE TIPO ESTÁTICO GAMMA. NO HAY DEFINIDO UN MÉTODO LLAMADO b QUE RECIBA COMO PARÁMETRO UN STRING Y DEVUELVA UN STRING EN GAMMA, Y TAMPOCO LO HEREDA DE ALFA. ✓

INST. 11. b EN ALFA x: 3 n: 1 ✓

INST. 12. b EN BETA x: TRES t: IPOO n: 1 ✓

INST. 13. a EN ALFA n: 1 b EN ALFA n: 1 ✓

INST. 14. ERROR DE COMPILACIÓN. C3 ES DE TIPO ESTÁTICO ALFA, Y NO TIENE DEFINIDO NINGÚN MÉTODO LLAMADO b QUE RECIBA COMO PARÁMETRO UN STRING. AL SER LA CLASE MÁS GENERAL, TAMPOCO LO HEREDA. (X)

a x

INST. 15. (b) EN BETA x: 3 t: IPOO n: 1 (X)

n: 1 a en ALFA n: 1

INST. 16. ERROR EN EJECUCIÓN. SE INTENTA ENVIAR UN MENSAJE A UNA VARIABLE QUE TIENE UNA REFERENCIA NULA.

INST. 17. a EN GAMMA t: 10 n: 1

EJERCICIO 2.

Ⓐ
```
Publ abstract class Vehiculo {
        protected int cantRuedas;
        protected float peso;
        public Vehiculo(int r, float p){
            cantRuedas = r;
            peso = p;
        }
        public int obtenerRuedas(){
            return cantRuedas;
        }
        public float obtenerPeso(){
            return peso;
        }
        public float fisionomia(){
            return peso/cantRuedas;
        }
Public  abstract float potencia();
Public  abstract boolean equals(Vehiculo v);
    }

public class Camión extends Vehiculo {
    protected int caballosDeFuerza;
    public Camión(int r, float p, int c) {
        super(r, p);
        caballosDeFuerza = c;
    }
    public int obtenerCaballosF(){
        return caballosDeFuerza;
    }
}
```

NOTA

```
    public float fisionomia(){
        return super.fisionomia() * caballosDeFuerza;
    }
    public float potencia(){
        return fisionomia() * caballosDeFuerza;
    }
    public boolean equals(Vehiculo v) {
        boolean e = false;
        if (v != null && this.getClass() == v.getClass()){
            Camión c = (Camión) v;
            e = cantRuedas == c.obtenerRuedas() && peso == c.obtenerPeso()
                && caballosDeFuerza == c.obtenerCaballosF();
        }
        return e;
    }

public class Tractor extends Vehiculo {
    protected int empuje;
    public Tractor(int r, float p, int e) {
        super(r, p);
        empuje = e;
    }
    public int obtenerEmpuje(){
        return empuje;
    }
    public float fisionomía(){
        return peso/empuje;
    }
    public float potencia(){
        float p = empuje * cantRuedas;
        return p;
    }
```

HOJA 3

```
public boolean equals(Vehiculo v){
    boolean e = false;
    if(v != null && this.getClass() == v.getClass()){
    Tractor t = (Tractor) v;
    e = cantRuedas == t.obtenerRuedas() && peso == t.obtenerPeso()
      && empuje == t.obtenerEmpuje();
    }
    return e;
}
```

(B)

```
public class Tabla{
    protected Vehiculo [] a;
    public void insertar(Vehiculo v, int i){
        a[i] = v;
    }
    public void eliminar(Vehiculo v){
        for(int i=0; i< a.lenath; i++){
            if(a[i] != null && a[i].equals(v)){
                a[i] = null;
            }
        }
    }
    public Tabla fisionomiaMayorAn(int n){
        Tabla t = new Tabla(a.length);
        int pos = 0;
        for(i=0; i< a.length; i++){
            if(a[i]!=null && a[i].fisionomia() >n){
                t.insertar(a[i], pos);
                pos = pos + 1;
            }
        }
        return t;
    }
```

NOTA

```
public boolean nConPotnciax(int n, float x){
    int contador = 0;
    for(int i=0; i< a.length && contador<=n; i++){
        if(a[i] != null && a[i].potencia() == x ){
            contador = contador + 1;
        }
    }
    return (contador == n);
}
```