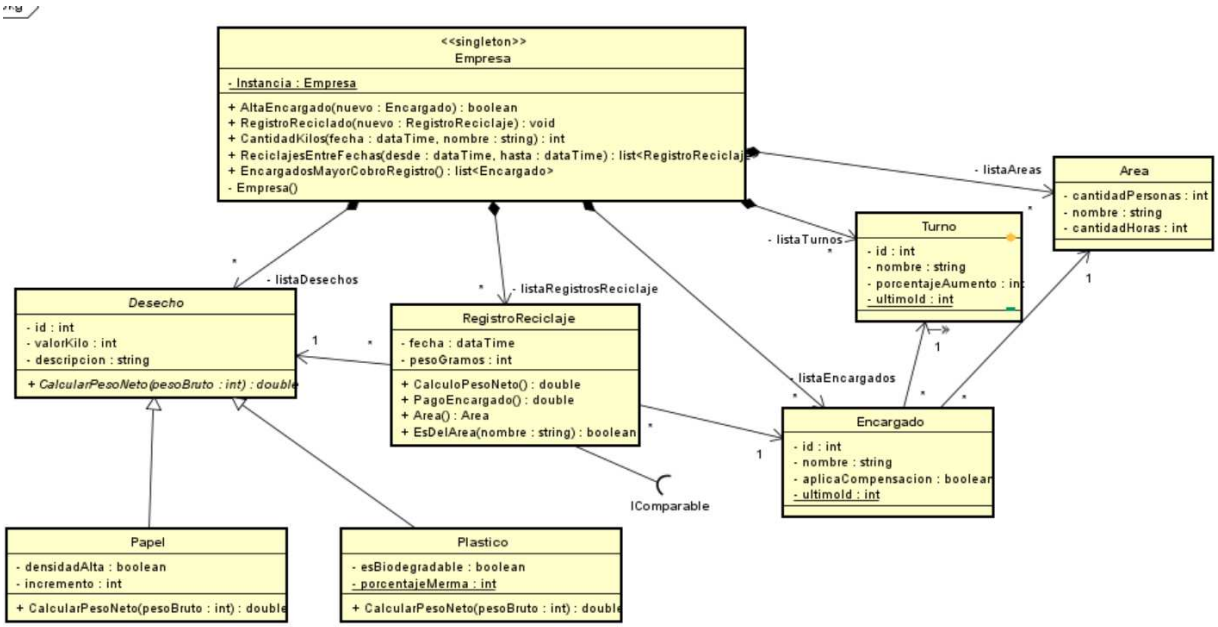
UNIVERSIDAD ORT
Uruguay

EVALUACIÓN	**SOLUCION EXAMEN**	**GRUPO**	TODOS	**FECHA**	04/05/2018
MATERIA	PROGRAMACIÓN 2				
CARRERA	AP/ATI				
CONDICIONES	**- Puntos: 100 puntos** **- Duración: 3 Horas** **- Sin material** **- Otros : No escribir en la hoja de la letra** Consultas solamente sobre interpretación de letra.				

NOTA: Se presenta idea de solución no detallada

PARTE 1



UNIVERSIDAD ORT
Uruguay

PARTE 2

REQUERIMIENTO C

En clase Empresa:

```
publicdoubleCantidadKilos(DateTime fecha, string nombre) {
double kilos=0;
foreach(RegistroReciclajeunRegistroinListaRegistrosReciclaje)
            {
if (unRegistro.Fecha == fecha &&unRegistro.EsDelArea(nombre))
                {
                    kilos += unRegistro.CalculoPesoNeto();
                }
            }
return kilos;
}
```

En clase RegistroReciclaje:

```
       publicboolEsDelArea(stringnombre)
        {
returnArea().Nombre == nombre;
        }

publicAreaArea() {
returnEncargado.Area;
        }

publicdoubleCalculoPesoNeto()
        {
returnDesecho.CalcularPesoBruto(PesoGramos);
        }
```

En clase Desecho:

```
       publicvirtualdoubleCalcularPesoBruto(intpesoNeto)
        {
return (double)pesoNeto / 1000;
        }
```

UNIVERSIDAD ORT
Uruguay

En clase Plastico (hereda de Desecho):

```
       publicoverridedoubleCalcularPesoBruto(intpesoNeto)
        {
double peso = base.CalcularPesoBruto(pesoNeto);

if (Biodegradable) {
                peso -= peso * 0.15;
            }

            peso -= peso * Plastico.PorcentajeMerma/100;
return peso;
        }
```

En clase Papel (hereda de Desecho):

```
       publicoverridedoubleCalcularPesoBruto(intpesoNeto)
        {
double peso = base.CalcularPesoBruto(pesoNeto);

if (DensidadAlta)
            {
                peso -= peso * Incremento;
            }

return peso;
        }
```

REQUERIMIENTO D

En clase Empresa;

```
publicList<RegistroReciclaje>ReciclajesEntreFechas(DateTime desde, DateTime hasta)
{

List<RegistroReciclaje> resultado = newList<RegistroReciclaje>();

foreach (RegistroReciclaje registro inListaRegistrosReciclaje) {
if (registro.Fecha>=desde &&registro.Fecha<=hasta) {
if (registro.Area().CantidadPesonas>= 10) {
resultado.Add(registro);
                    }
                }
            }
resultado.Sort();
return resultado;

}
```

En clase RegistroReciclaje (implementa interfaz IComparable<RegistroReciclaje>):

```
        publicintCompareTo(RegistroReciclaje other)
         {
returnother.Fecha.CompareTo(this.Fecha);
}
```

FUNCIONALIDAD E

En clase Empresa:

```
        publicList<Encargado>EncargadosMayorCobroRegistro()
         {
List<Encargado> resultado = newList<Encargado>();
doublemaximo = 0;
foreach (RegistroReciclajeunRegistroinListaRegistrosReciclaje) {

double pago = unRegistro.PagoEncargado();
if (pago >maximo)
                {
resultado.Clear();
maximo = pago;
resultado.Add(unRegistro.Encargado);
                }
else {
if (pago == maximo) {
if(!resultado.Contains(unRegistro.Encargado))
                        {
resultado.Add(unRegistro.Encargado);
                        }
                    }
                }

            }
return resultado;
        }
```

En clase RegistroReciclaje:

```
        publicdoublePagoEncargado()
         {
double resultado = this.CalculoPesoNeto() * this.ValorKilo;

if (Encargado.AplicaCompensacion) {
                resultado += 500;
            }

            resultado += resultado * Encargado.Turno.PorcentajeAumento/100;

return resultado;
        }
```