GUÍA DE LABORATORIO N° 09

Escuela Profesional: Ingeniería de Sistemas. **Asignatura:** Prog. Orientada a Objetos
Ciclo y Turno: Cuarto –Noche **Semestre Académico:** 2023-1

Sesión 09: Polimorfismo y Tratamiento de excepciones

INTRODUCCION

El polimorfismo es una característica de los objetos, y permite obtener muchas múltiples formas con la finalidad de acceder a los diferentes métodos y atributos a través de una solo declaración. El tratamiento de excepciones permite manejar los errores generados en tempo de ejecución.

I. OBJETIVOS

1. Codificar, compilar y ejecutar los ejercicios de aplicación.
2. Conocer la estructura de un programa visual en Java.

II. EQUIPOS Y MATERIALES

- Computadora personal.
- Programa NetBeans IDE 6.9.1 correctamente instalado.

III. METODOLOGIA Y ACTIVIDADES

- Codificar los ejercicios desarrollados en el aula.
- Presentar avances y ejecución de cada uno de los ejercicios al docente o jefe de práctica encargado para la calificación correspondiente.

IV. IMPORTANTE

- Antes de comenzar a manipular el NetBeans 6.9.1 crear siempre, una carpeta, donde se guardarán todas las aplicaciones realizadas en el presente laboratorio con el nombre LABORAT-09.

V. MANEJO DEL SOFTWARE

Implemente una clase padre abstracta de nombre **Empleado** con los siguientes atributos: DNI, apellidos, nombres, así como su constructor correspondiente que actualice su información y considere un método para mostrar la información de los atributos así como sus métodos getter y setter (Encapsulamiento). Así como también defina métodos abstractos para los ingresos, bonificación y descuentos, y también métodos no abstractos para el sueldo y la impresión de la información del empleado.

Luego construya una clase hija de nombre **EmpleadoVendedor** con los siguientes atributos: monto vendido y la tasa de comisión y su respectivo constructor y los métodos setter y getter (encapsulamiento). Para esta clase desarrolle los métodos abstractos heredados con las siguientes características y que permita hacer lo siguiente:

- Calcular sus ingresos basados en montovendido*tasacomision.
- Calcular su bonificación de la siguiente manera: si el monto vendido es menor que 1000 no tendrá bonificación, si estas entre 1000 y 5000 será el 5% de sus ingresos, si supera los 5000 su bonificación será del 10% de sus ingresos.
- Calcular su descuento de la siguiente manera: si sus ingresos es menor a 1000 su descuento será del 11% de sus ingresos sino será el 15% de sus ingresos.
- Calcular su sueldo neto como: ingresos + bonificaciones - descuentos

Luego construya otra clase hija de nombre **EmpleadoPermanente** con los siguientes atributos: sueldo base y afiliación (AFP y SNP) e incluya su constructor y los métodos setter y getter (encapsulamiento). Para esta clase desarrolle los métodos abstractos

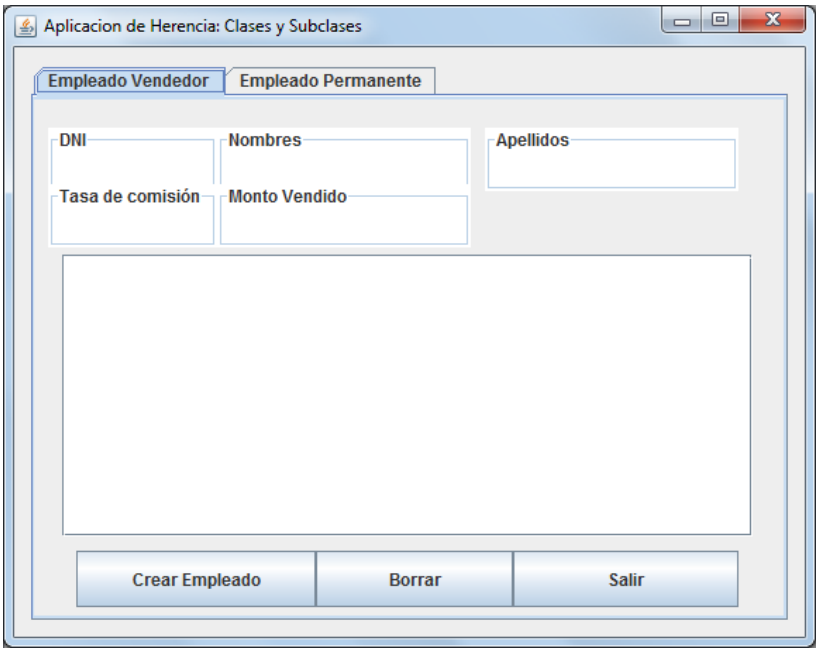
heredados con las siguientes características y que permita hacer lo siguiente:

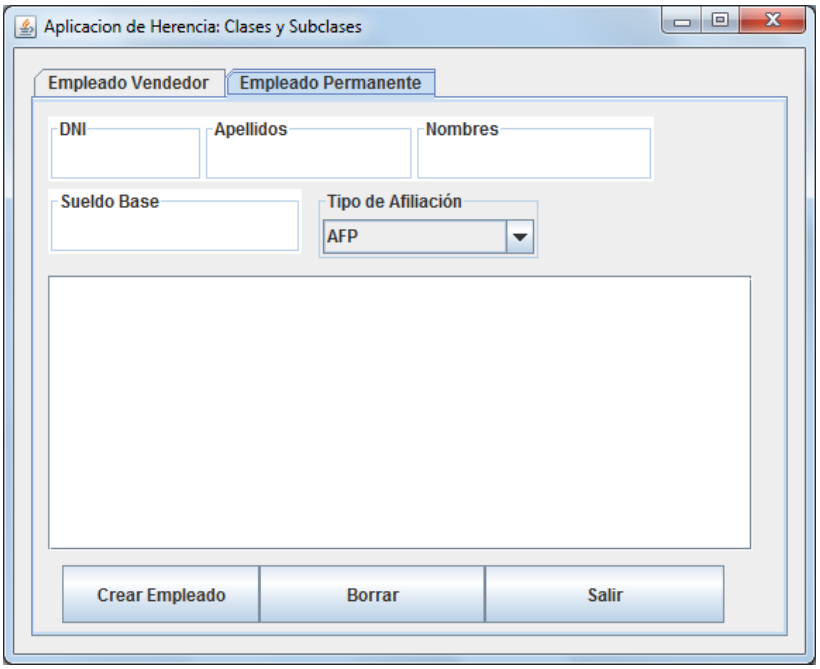
- Retornar el sueldo base, método que llamara ingresos
- Calcular el descuento basado en: si la afiliación es AFP será el 15% del sueldo base sino será el 11% del sueldo base.
- No hay bonificación es decir la bonificación es 0.
- El sueldo neto se calculara como: ingresos – descuentos
- Además recibirá asignación de movilidad si sus ingresos son menores de 1000 soles recibirá 50 soles sino recibirá 40 soles.

Considere los objetos necesarios para utilizar la funcionalidad de la clase

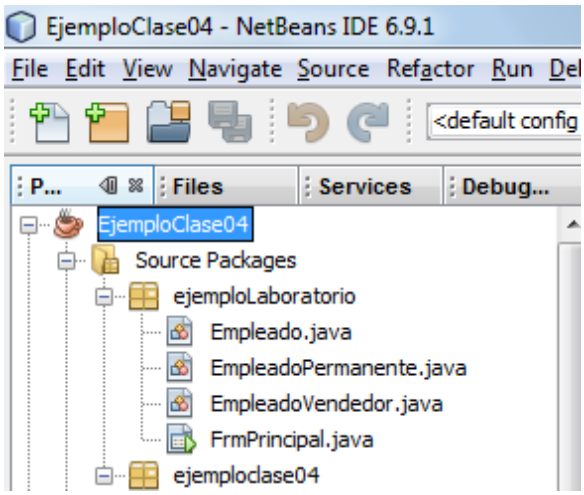
SOLUCION

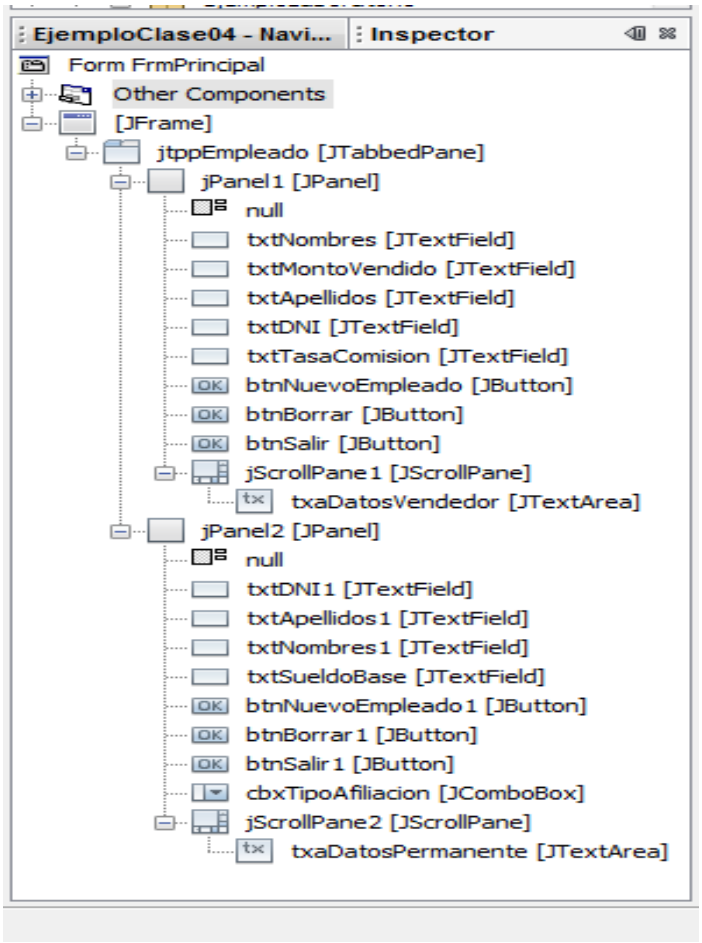
Utilice la paleta de componentes y construya el siguiente diseño:





Observe la gráfica y su proyecto quedara de la siguiente forma que constara de 3 clases y un formulario. Observe al lado derecho el Inspector de Objetos para guiarse de los nombres de los objetos





Del enunciado la clase **Empleado** tiene la siguiente información:

```
1  package clase03;
2  import java.text.DecimalFormat;
   public abstract class Empleado {
4      // ATRIBUTOS
5      private String dni;
6      private String apellidos;
7      private String nombres;
8      // CONSTRUCTORES
9      public Empleado(){ }
10     public Empleado(String dni, String apellidos, String nombres){
11         this.dni=dni;
12         this.apellidos=apellidos;
13         this.nombres=nombres;
14     }
15     //ENCAPSULAMIENTO DE DATOS - Metodos Setter y Getter
16     public String getDni()                    {return dni;                }
17     public void setDni(String dni)            {this.dni = dni;            }
18     public String getApellidos()              {return apellidos;          }
19     public void setApellidos(String apellidos) {this.apellidos = apellidos; }
20     public String getNombres()                {return nombres;            }
21     public void setNombres(String nombres)    {this.nombres = nombres;    }
```

```
22      // METODOS ADICIONALES
        public String MostrarInfo(){
24          DecimalFormat df=new DecimalFormat("##0.00");
25          return "\nNombres \t\t: "       +getNombres()+
26                 "\nApellidos \t\t: "    +getApellidos()+
27                 "\nD.N.I.   \t\t: "     +getDni()+
28                 "\nIngresos\t\t: "      +df.format(ingresos())+
29                 "\nBonificaciones\t\t: "+df.format(bonificaciones())+
30                 "\nDescuentos\t\t: "    +df.format(descuentos())+
31                 "\nSueldo Neto\t\t: "   +df.format(sueldoneto());
32      }
33      public double sueldoneto()
34      {
35          return ingresos()+bonificaciones()-descuentos();
36      }
37  // METODOS ABSTRACTOS
        public abstract double ingresos();
        public abstract double bonificaciones();
        public abstract double descuentos();
41  }
```

La clase hija **EmpleadoVendedor** tiene la siguiente información

```
1   package clase03;
2   public class EmpleadoVendedor extends Empleado{
3      private double montovend;
4       private double tasacomis;
5       // CONSTRUCTORES
6       public EmpleadoVendedor(String dni, String nombres, String apellidos,
7                               double monvend, double tasacom){
8           super(dni,nombres,apellidos);
9           this.montovend=monvend;
10          this.tasacomis=tasacom;
11      }
12      //ENCAPSULAMIENTO DE DATOS - Metodos Getter y Setter
13      public double getMontovendido()                 { return montovend;             }
14      public void setMontovendido(double montovend)   { this.montovend = montovend; }
15      public double getTasacomision()                 { return tasacomis;             }
        public void setTasacomision(double tasacomision){ this.tasacomis = tasacomis; }
17      // CONSTRUCCION DE METODOS ABSTRACTOS HEREDADOS
        public double ingresos(){
19          return getMontovendido()*getTasacomision();
20      }
        public double bonificaciones(){
22          if(getMontovendido()<1000)
23              return 0;
24          else if(getMontovendido()<5000)
25              return ingresos()*0.05;
26          else
27              return ingresos()*0.10;
28      }
        public double descuentos(){
30          if(ingresos()<1000)
31              return ingresos()*0.11;
32          else
33              return ingresos()*0.15;
34      }
35       // METODOS REUTILZADOS - SOBRECARGA DE METODOS
        public String MostrarInfo(){
37           return "EMPLEADO PERMANENTE: \n"+super.MostrarInfo();
38      }
39  }
```

La clase hija **EmpleadoPermanente** tiene la siguiente información

```
1  package clase03;
2  public class EmpleadoPermanente extends Empleado{
3      private double sueldobase;
4      private String afiliacion;
5      // CONSTRUCTOR
6      public EmpleadoPermanente(String dni, String apellidos, String nombres,
7                                double sueldobase, String afiliacion){
8          super(dni,apellidos,nombres);
9          this.sueldobase=sueldobase;
10         this.afiliacion=afiliacion;
11     }
12    //ENCAPSULAMIENTO DE DATOS - Metodos Setter y Getter
13    public double getSueldobase()                { return sueldobase;             }
14    public void setSueldobase(double sueldobase) { this.sueldobase = sueldobase; }
15    public String getAfiliacion()                { return afiliacion;             }
16    public void setAfiliacion(String afiliacion) { this.afiliacion = afiliacion; }
17     //CONSTRUCCION DE LOS METODOS ABSTRACTOS HEREDADOS
18     public  double ingresos(){
19         return getSueldobase();
20     }
21     public double descuentos(){
22         if(getAfiliacion().equalsIgnoreCase("AFP"))
23             return getSueldobase()*0.15;
24         else  return getSueldobase()*0.11;
25     }
26     public double bonificaciones(){
27          return 0;
28     }
29     // METODO REUTILIZADO - SOBRECARGA DE METODO
30     public String MostrarInfo(){
31
32         return "EMPLEADO PERMANENTE: \n"+super.MostrarInfo();
33     }
34 }
```

```
34
35      //Metodo Propio que retorna la movilidad segun sus ingresos
36      public double movilidad()
37      {  if(ingresos()<1000) return 50;
38             else return 40;
39      }
40  }
41
```

A continuación complete lo siguientes códigos en el **Editor de Código**

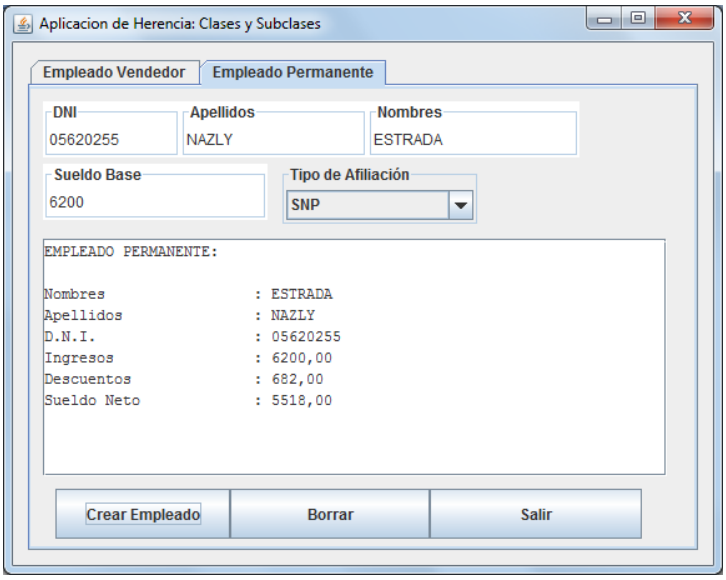
```
1   package ejemploLaboratorio;
2   import java.awt.Font;
3   import javax.swing.JOptionPane;
4   import javax.swing.JTextArea;
5   public class FrmPrincipal extends javax.swing.JFrame {
6       /** Creates new form FrmPrincipal */
7       public FrmPrincipal() {
8           initComponents();
9           this.setLocationRelativeTo(null);
10      }
11      @SuppressWarnings("unchecked")
12      Generated Code
185
186     private void jbtnCrearVendedorActionPerformed(java.awt.event.ActionEvent evt) {
187     //capturando los datos de entrada
188         double mv=0,tc=0;
189          String dni=txtDNI.getText();
190         String nom=txtNombres.getText();
191         String ape=txtApellidos.getText();
192         //verificando si el monto vendido es un número real
193         if(leeMontoVendido()==-1)     mensaje("Error: Ingrese Monto Vendido correcto");
194         else   mv=leeMontoVendido();
195         //verificando si el monto vendido es un número real
196         if(leeTasaComision()==-1)   mensaje("Error: Ingrese Tasa Comisión correcta");
197         else tc=leeTasaComision();
198         //verificando que los datos correctos númericos
199         if(leeMontoVendido()!=-1 && leeTasaComision()!=-1)
200         {  //generando objeto EmpleadoVendedor
201            EmpleadoVendedor ev = new EmpleadoVendedor(dni,ape,nom,mv,tc);
202            //Mostrando los datos generados del objeto
203            Listar(ev,txaDatosVendedor);
204         }
205     }
206
```

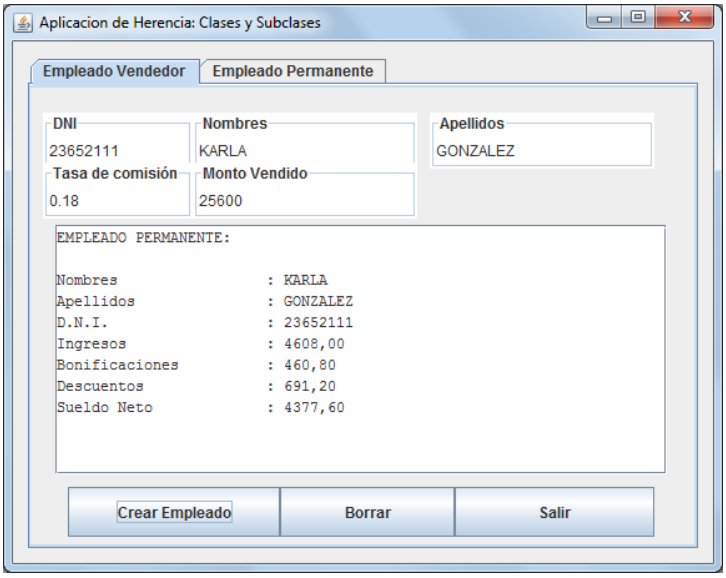
METODOS PARA EL MANEJO DE EXCEPCIONES

```
206
207     //METODOS QUE EVALUAN EXCEPCIONES
208     public double leeMontoVendido()
209     {
210         try  {  return Double.parseDouble(txtMontoVendido.getText());    }
211         catch(Exception ex){  return -1;    }
212     }
213
214     public double leeTasaComision()
215     {
216         try {   return Double.parseDouble(txtTasaComision.getText());     }
217         catch(Exception ex){ return -1; }
218     }
219     public void mensaje(String s)
220         {    JOptionPane.showMessageDialog(this,s);
221     }
222
223     public double leeSueldoBase()
224     {
225         try { return Double.parseDouble(txtSueldoBase.getText());     }
226         catch(Exception ex){ return -1; }
227     }
228     // metodo que recibe un empleado e imprime su informacion en un TextArea recibido
229     // como parametro
230     public void Listar(Empleado e,JTextArea txa)
231     {  txa.append(e.MostrarInfo());
232        if(e instanceof EmpleadoPermanente)
233            txa.append("\nMovilidad\t\t:"+((EmpleadoPermanente)e).movilidad());
234     }
235
248
249     private void jbtnCrearPermanenteActionPerformed(java.awt.event.ActionEvent evt) {
250         //capturando los datos de entrada
251         double s_base=0;
252         String dni=txtDNI1.getText();
253         String nom=txtNombres1.getText();
254         String ape=txtApellidos1.getText();
255         String tipo_af=cbxTipoAfiliacion.getSelectedItem().toString();
256         //verificando que el sueldo este correcto
257         if(leeSueldoBase()==-1) mensaje("Error: Ingrese un sueldo correcto..");
258         else s_base = leeSueldoBase();
259          //generando objeto EmpleadoPermanente
260         if(leeSueldoBase()!=-1)
261         {   EmpleadoPermanente ep = new EmpleadoPermanente(dni,ape,nom,s_base,tipo_af);
262             //Mostrando los datos generados del objeto
263             Listar(ep,txaDatosPermanente);
264         }
265     }
266
```

```
221
222     private void btnBorrar1ActionPerformed(java.awt.event.ActionEvent evt) {
223         //limpiando las entradas
224         txtDNI.setText("");
225         txtNombres.setText("");
226         txtApellidos.setText("");
227         txaSalida.setText("");
228         txtSueldoBase.setText("");
229         cbxTipoAfiliacion.setSelectedIndex(0);
230         txtNombres.requestFocus();
231     }
232
233     private void btnSalir1ActionPerformed(java.awt.event.ActionEvent evt) {
234         System.exit(0);
235     }
236
237     private void formWindowOpened(java.awt.event.WindowEvent evt) {
238
239      txaSalida.setFont(new Font("monospaced",Font.PLAIN,12));
240      txaSalida1.setFont(new Font("monospaced",Font.PLAIN,12));
241
242     }
243
196     private void btnBorrarActionPerformed(java.awt.event.ActionEvent evt) {
197         txtDNI.setText("");
198         txtNombres.setText("");
199         txtApellidos.setText("");
200         txaSalida.setText("");
201         txtMontoVendido.setText("");
202         txtNombres.requestFocus();
203     }
204
205     private void btnSalirActionPerformed(java.awt.event.ActionEvent evt) {
206      System.exit(0);
207     }
208
```

Presione Shift+F6 y el aplicativo mostrará la siguiente ventana





EJERCICIO PROPUESTO

Construya una clase abstracta de nombre **Estudiante** que tiene los siguientes atributos: código, apellidos, nombres, semestre de ingreso (2010-I o 2011-II) e incluya sus constructores. Además construya sus métodos getter y setter (Encapsulamiento) y también declare los **métodos abstractos** como pago parcial de pensión y descuentos, además construya **métodos no abstractos** como cálculo del pago final de pensión que será igual a: pago parcial de pensión - descuentos. Implemente dicha clase con la interface Serializable.

Construya una interface que permita manejar las siguientes constantes el descuento de porcentajes del 19% del IGV. Que será implementada en la clase Estudiante

Luego construya una subclase **Estudiante de PreGrado** de la clase Estudiante que tendrá los atributos como: Categoría (A o B), Promedio ponderado, colegio de Procedencia (Estatal o Particular) y contador de atributo estático. Construya su constructor que actualizará los datos de las clase padre así como actualizara el atributo código, heredado de la Clase Estudiante con un código autogenerado y sus métodos getter y setter. Para esta clase desarrolle los métodos abstractos heredados y que permitan hacer lo siguiente:

- ③ Calcular el pago parcial de pensiones en función a la categoría y el colegio de procedencia tal y como se muestra en la tabla

Categoría	Pago parcial de Pensiones	
	Estatal	Particular
A	320	450
B	300	420

- ③ Calcular el descuento sobre el pago parcial de pensiones de acuerdo al promedio ponderado como sigue a continuación

Rango del Prom Ponderado	% de descuento sobre el pago de pensión
Menor a 13	0%
Mayor 13	8%

- ③ Tendrá su metodo propio que permita generar un código con la siguiente característica: EPRE0001, EPRE0002…

Luego construya la subclase **Estudiante de Postgrado** de la clase Estudiante que tiene los siguientes atributos: grado académico (Bachiller Titulado) y contador con atributo estático. Construya su constructor que actualizará los datos de las clase padre así como actualizar el atributo código, heredado de la Clase Estudiante con un código autogenerado y sus métodos getter y setter. Para esta clase desarrolle los métodos abstractos heredados y que permitan hacer lo siguiente:

- ③ El sueldo base se calcula de la siguiente tabla

Grado académico	Pago parcial de Pensiones
Bachiller	400
Titulado	350

- ③ Los Estudiantes de Postgrado no gozan de descuentos.
- ③ Tendrá su método propio que permita generar un código con la siguiente característica: EPOS0001, EPOS0002…

NOTA: *Ambas clases tendrán métodos recargados para mostrar la información de sus datos adicionales.*

Construya el aplicativo con los objetos necesarios para utilizar la funcionalidad de las clases y subclases implementadas utilice el tratamiento de excepciones para verificar los datos de entrada del tipo numérico.