

GUÍA DE LABORATORIO N° 09**Escuela Profesional:** Ingeniería de Sistemas.
Ciclo y Turno: Cuarto –Noche**Asignatura:** Prog. Orientada a Objetos
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 sola declaración. El tratamiento de excepciones permite manejar los errores generados en tiempo 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

- ③ Computadora personal.
- ③ 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 $\text{montovendido} * \text{tasacomision}$.
- 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: $\text{ingresos} + \text{bonificaciones} - \text{descuentos}$

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:

Aplicacion de Herencia: Clases y Subclases

Empleado Vendedor Empleado Permanente

DNI Nombres Apellidos

Tasa de comisión Monto Vendido

Crear Empleado Borrar Salir

Aplicacion de Herencia: Clases y Subclases

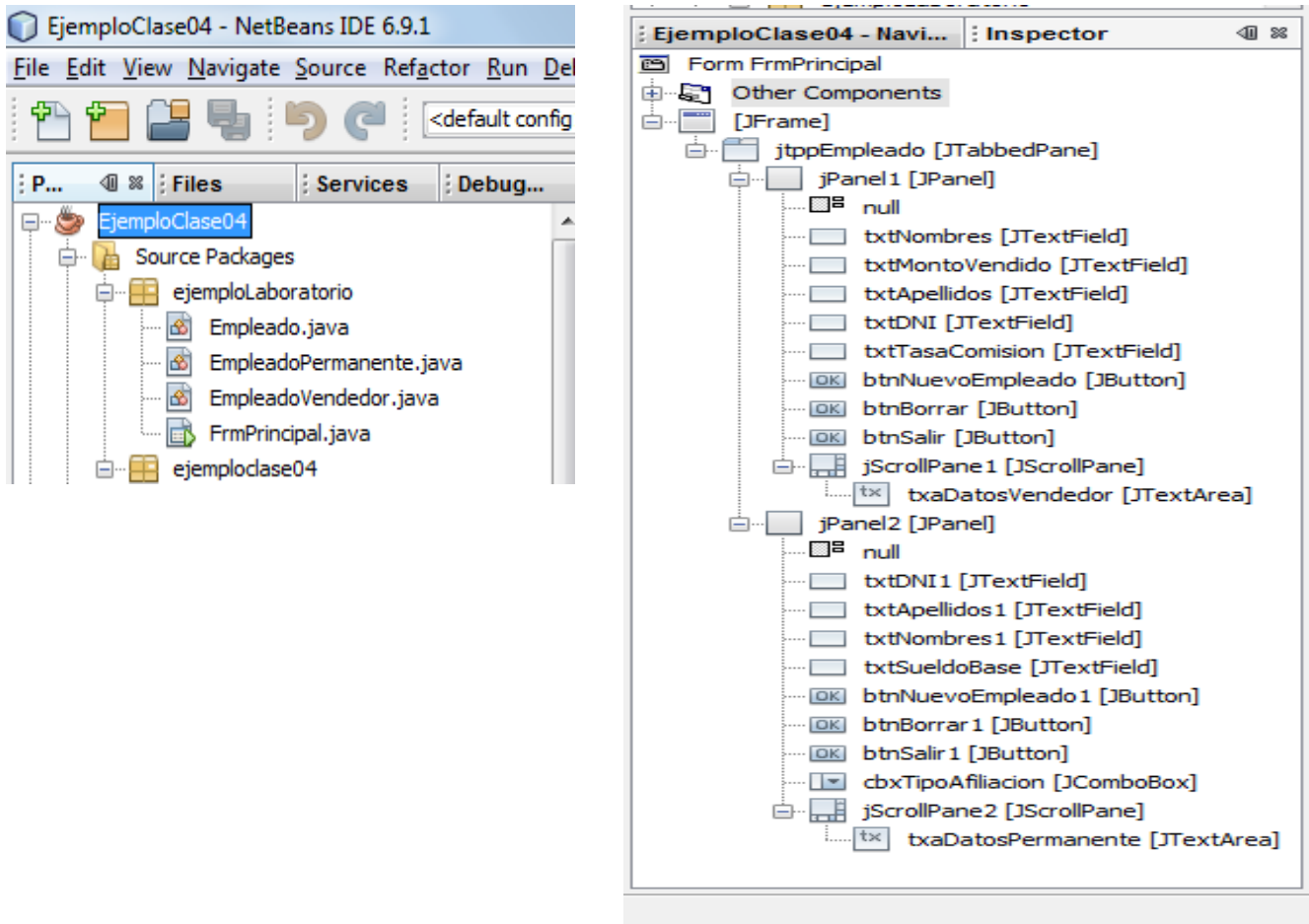
Empleado Vendedor Empleado Permanente

DNI Apellidos Nombres

Sueldo Base Tipo de Afiliación
AFP

Crear Empleado Borrar Salir

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;
3 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
23 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
38 public abstract double ingresos();
39 public abstract double bonificaciones();
40 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; }
16    public void setTasacomision(double tasacomision){ this.tasacomis = tasacomis; }
17    // CONSTRUCCION DE METODOS ABSTRACTOS HEREDADOS
18    public double ingresos(){
19        return getMontovendido()*getTasacomision();
20    }
21    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    }
29    public double descuentos(){
30        if(ingresos()<1000)
31            return ingresos()*0.11;
32        else
33            return ingresos()*0.15;
34    }
35    // METODOS REUTILIZADOS - SOBRECARGA DE METODOS
36    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 getSuelldobase() { return sueldobase; }
14    public void setSuelldobase(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 getSuelldobase();
20    }
21    public double descuentos(){
22        if(getAfiliacion().equalsIgnoreCase("AFP"))
23            return getSuelldobase()*0.15;
24        else return getSuelldobase()*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 }
35
36 //Metodo Propio que retorna la movilidad segun sus ingresos
37 public double movilidad()
38 { if(ingresos()<1000) return 50;
39   else return 40;
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 jButtonCrearVendedorActionPerformed(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 numéricos
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
236
237
238
239 private void jbtnCrearPermanenteActionPerformed(java.awt.event.ActionEvent evt) {
240     //capturando los datos de entrada
241     double s_base=0;
242     String dni=txtDNI1.getText();
243     String nom=txtNombres1.getText();
244     String ape=txtApellidos1.getText();
245     String tipo_af=cbxTipoAfiliacion.getSelectedItem().toString();
246     //verificando que el sueldo este correcto
247     if(leeSueldoBase() == -1) mensaje("Error: Ingrese un sueldo correcto..");
248     else s_base = leeSueldoBase();
249     //generando objeto EmpleadoPermanente
250     if(leeSueldoBase() != -1)
251     { EmpleadoPermanente ep = new EmpleadoPermanente(dni,ape,nom,s_base,tipo_af);
252       //Mostrando los datos generados del objeto
253       Listar(ep,txaDatosPermanente);
254     }
255 }
256
```

```

221
222 private void btnBorrar1ActionPerformed(java.awt.event.ActionEvent evt) {
223     //limpiando las entradas
224     txtDNI.setText("");
225     txtNombres.setText("");
226     txtApellidos.setText("");
227     txtaSalida.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     txtaSalida.setFont(new Font("monospaced", Font.PLAIN, 12));
240     txtaSalida1.setFont(new Font("monospaced", Font.PLAIN, 12));
241
242 }
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999

```


Presione Shift+F6 y el aplicativo mostrará la siguiente ventana

Aplicacion de Herencia: Clases y Subclases

Empleado Vendedor Empleado Permanente

DNI: 05620255 Apellidos: NAZLY Nombres: ESTRADA

Sueldo Base: 6200 Tipo de Afiliación: SNP

EMPLEADO PERMANENTE:

Nombres : ESTRADA
Apellidos : NAZLY
D.N.I. : 05620255
Ingresos : 6200,00
Descuentos : 682,00
Sueldo Neto : 5518,00

Crear Empleado Borrar Salir

Aplicacion de Herencia: Clases y Subclases

Empleado Vendedor Empleado Permanente

DNI: 23652111 Nombres: KARLA Apellidos: GONZALEZ

Tasa de comisión: 0.18 Monto Vendido: 25600

EMPLEADO PERMANENTE:

Nombres : KARLA
Apellidos : GONZALEZ
D.N.I. : 23652111
Ingresos : 4608,00
Bonificaciones : 460,80
Descuentos : 691,20
Sueldo Neto : 4377,60

Crear Empleado Borrar Salir

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 (Estatad o Particular) y contador de atributo estático. Construya su constructor que actualizará los datos de las clase padre así como actualizará 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	
	Estatad	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.