UTNMDP
Regional Mar del Plata

Guia #2
Ejercitación Adicional

Ejercicio 1. Clases abstractas

```
public abstract class A
{
  public abstract void dinamica();
  public void estatica()
  {
    System.out.println( "Método estático de la clase A");
  }
}

public class B extends A
{
  public void dinamica()
  {
    System.out.println( "Método dinámico de clase B");
  }
}

public class C extends A
{
  public void dinamica()
  {
    System.out.println( "Método dinámico de clase C");
  }
}

public class Ligadura
{
  static public void main(String [] ar)
  {
    A a;
    B b = new B();
    C c = new C();
    System.out.print( "Métodos llamados con objeto b desde");
    System.out.println("referencia de la clase A");
    a = b;
    a.dinamica();
    a.estatica();
```

```
System.out.print( "Métodos llamados con objeto c desde");
System.out.println(" referencia de la clase A");
a = c;
a.dinamica();
a.estatica();
```

- ¿Cual es la salida por pantalla?

Ejercicio 2

La empresa XYZ requiere una aplicación informática para administrar los datos de su personal.

Del personal se conoce: número de DNI, nombre, apellidos y año de ingreso.

Existen dos categorías de personal: el personal con salario fijo y el personal a comisión. Los empleados con salario fijo tienen un sueldo básico y un porcentaje adicional en función del número de años que llevan: menos de dos años salario base, de 2 a 3 años: 5% más; de 4 a 7 años: 10% más; de 8 a 15 años: 15% más y más de 15 años: 20% más.

Los empleados a comisión tienen un salario mínimo que será constante para todos los empleados de este tipo e igual a 750.00$, un número de clientes captados y un monto por cada cliente captado.

El salario se obtiene multiplicando los clientes captados por el monto por cliente, si el salario por los clientes captados no llega al salario mínimo, cobrará esta cantidad.

Se contará con una clase padre Empleado de la cual no se podrán crear objetos y de la que heredan las clases EAsalariado y EComision. En todas las clases debe haber un constructor con parámetros para todos los atributos y otro vacío. En todos deben crearse los getters and setters correspondientes. Empleado contará con un método imprimir() y un método obtenerSalario().

Se creará una clase gestora y en el método main se creará un arreglo con los siguientes objetos:

- Javier Gómez, DNI: 569587A, desde 2008, salario fijo base = 1225.00$.
- Eva Nieto, DNI: 695235B, desde 2010, 179 clientes captados a 8.10$ cada uno.
- José Ruiz, DNI: 741258C, desde 2012, 81 clientes captados a 7.90$ cada uno.
- María Núñez, DNI: 896325D, desde 2013, salario fijo base = 1155.00$

Los dos primeros se crearán utilizando el constructor con todos los parámetros y los dos últimos con el constructor vacío y utilizando los setters adecuados.
Desde el método main se llamará a estos otros dos métodos:

- sueldoMayor(): Dado un array de objetos Empleado muestra el nombre, apellido y salario del que más cobra.
- mostrarTodos(): Dado un array de objetos Empleado lo recorre imprimiendo los datos de todos ellos.

Ejercicio 3: TestFlujo

En la siguiente jerarquía de herencia existen llamadas a métodos en los que se debe tener claro el concepto de polimorfismo. Escribe cuál sería el resultado de la ejecución del siguiente código:

```
abstract class Flujo {

   abstract public void escribe(char c);

   public void escribe(String s) {
      for (int i = 0; i < s.length(); i++) {
         System.out.println("Escribe de Flujo ....");
         escribe(s.charAt(i));
      }
   }

   public void escribe(int i) {
      escribe("" + i);
   }
}

class Disco extends Flujo {

   public void escribe(char c) {
      System.out.println("Escribe de disco " + c);
   }
}

class DiscoFlexible extends Disco {
   public void escribe(String s) {
      System.out.println("Escribe de Disco Flexible...");
      super.escribe(s);
   }
}

public class TestFlujo {

   public static void main(String a[]) {
      DiscoFlexible dc = new DiscoFlexible();
      Flujo f = dc;
      f.escribe("ab");
   }
}
```

UTNMDP
Regional Mar del Plata

Ejercicio 4

Detecta los dos errores que presenta este código y explica por qué no es correcto:

```
abstract class A {

   int i;

   A(int i)
{
      this.i = i;
   }
}

class B extends A {

   void metodo() {
      System.out.println(i);
   }
}

class Cuestion {

   public static void main(String[] args) {
      A[] v = new A[10];
      for (int i = 0; i < 10; i++)
{

         v[i] = new B();
         v[i].metodo();
      }
```