UTNMDP
Regional Mar del Plata

# Guia #2
# Modelado e Implementación

## Sobre esta guía

Lea atentamente cada uno de los enunciados y antes de llevar a código cada ejercicio realice en papel o con la ayuda de algún software, el correspondiente diagrama de UML, detallando los atributos, métodos y constructores que considere necesarios. Una vez terminado escriba el código y ejecute las pruebas que se detallan.

## Enunciados

1. Modela un sistema que gestione empleados de una empresa. Cada empleado tiene un nombre, un salario y un método calcularPago() que debe ser implementado de manera específica para cada tipo de empleado:

    a. Implementa las siguientes clases: Empleado (clase base abstracta), EmpleadoTiempoCompleto y EmpleadoPorHoras. Para los empleados a tiempo completo, el método calcularPago() debe retornar el salario mensual, mientras que para los empleados por horas, debe retornar el salario calculado según las horas trabajadas. Utiliza polimorfismo para invocar el método calcularPago() adecuado para cada empleado.

    b. Ahora, agrega la clase "EmpleadoContratista" que representa a empleados contratados por proyectos específicos. Esta nueva clase de empleado tiene un salario basado en el número de horas trabajadas y una tarifa por proyecto.

    c. Permite al usuario realizar las operaciones necesarias mediante un menú.

2. Es necesario modelar el objeto de tipo Libro con las siguientes caracteristicas: titulo, precio, stock y Autor, este último posee las características de: nombre, apellido, email y genero ( ‘M’ o ‘F’ ). Para este ejercicio vamos a asumir que un libro tiene un único autor.

Ejecute las siguientes instrucciones:

a. Inicialice un objeto de tipo Autor, “Joshua”, “Bloch”, “joshua@email.com”, ’M’.

b. Imprima por pantalla al autor previamente instanciado.

c. Inicialice el libro “Effective Java” del Autor “Joshua Bloch” que cuesta 450 pesos con una cantidad de 150 copias.

d. Imprima por pantalla el libro instanciado.

e. Modifique el precio del libro “Effective Java” a 500 pesos y aumente la cantidad en 50 copias.

f. Imprima por pantalla los atributos del Autor Joshua, accediendo desde el Libro “Effective Java”.

g. Agregue un método a la clase Libro que posibilite imprimir en pantalla el siguiente mensaje: “El libro, {título} de {nombre del autor}. Se vende a {precio} pesos.

3. Nos contratan para hacer un programa que lleve el control de las ventas de un local. Para esto es necesario modelar la clase Cliente, que posee un atributo id como identificador del cliente, el mismo debe ser un valor compuesto por letras y números aleatorios que se generan automáticamente al crear un Cliente. El Cliente también posee un nombre, un email y un porcentaje de descuento.
Por otro lado vamos a tener el objeto Factura que representa una venta del local, cada Factura posee un identificador de las mismas características usadas en Cliente. A su vez cada factura posee un monto total, una fecha y el Cliente que generó la compra. Para la fecha de la venta se le va a asignar la fecha y hora al momento de creación del objeto Factura. El tipo Factura debe contar con un método que calcule el monto final luego de aplicarle el descuento que posee el cliente.

   a. Investigue la clase UUID y sus métodos estáticos para la generación de los ids.

   b. Investigue la clase LocalDateTime y sus métodos estáticos para la generación de la fecha.

   c. Cree un objeto de tipo Cliente, imprima sus detalles por pantalla. Para esto haga uso de un método que facilite la

impresión del mismo. Cliente[id=?, nombre=?, email=?, descuento=?]
d. Cree un objeto de tipo Factura que posea al Cliente anteriormente creado. Una vez hecho esto, imprima por pantalla el monto total de esta Factura y el monto total luego de aplicarle el descuento.
e. Cree un método que facilite la impresión del objeto de tipo Factura y que siga el siguiente formato: Factura[id=?, fecha=?, monto=?, montoDesc=?, Cliente[id=?, nombre=?, email=?, descuento=?]]

4. Definir la clase Círculo, que posee como atributo un radio cuyo valor por defecto al ser inicializado sin valores es 1.0. Además el tipo Círculo posee un atributo color, por defecto rojo, un método para calcular el área y otro para imprimir sus características.
Por otro lado tenemos un tipo Cilindro, que extiende a la clase Circulo, por ende se convierte en subclase de Círculo. El Cilindro a diferencia del Círculo posee un atributo altura, que también se inicializa en 1.0 cuando no le pasamos un valor a su constructor. Y un método para calcular el volumen. Codifique ambas clases y realice las siguientes pruebas:
   - Inicializar un Cilindro y debuguear analizando los constructores a los que va llamando para inicializar la cadena de herencia. Imprimir por pantalla, el radio del cilindro, la altura, el área de la base y el volumen.
   - Inicializar un segundo CIlindro esta vez especificando la altura y radio. Imprimir por pantalla el radio, la altura, el área de la base y el volumen.
   - Sobreescribir el método calcular area declarado en Círculo, para que nos permita calcular el área del cilindro.
     (2π×radius×height + 2×areaBase)
   - Imprimir por pantalla el área y el volumen del cilindro. Vamos a notar que el cálculo del volumen difiere, esto es porque nosotros sobreescribimos el método calcular área y ahora nos calcula el área de un cilindro. Modificar el método calcular volumen en la clase cilindro para que llame al método de la SUPER clase que calcula el área.

- Modificar el método correspondiente de la clase cilindro para que imprima por pantalla lo siguiente: Cilindro: subclase de + {método que muestra info de Círculo} +altura= {alturaCilindro};

5. Vamos a diseñar un programa que nos permita gestionar el personal que concurre a una universidad. Para no hacerlo muy extenso vamos a limitarnos a estudiantes y miembros de staff que a diferencia de los estudiantes, estos perciben una remuneración. Ambos tipos comparten la característica de Persona que posee los atributos de dni, nombre, apellido, email y dirección.
   Por otro lado tenemos al Estudiante que posee las características de Persona y las de un estudiante, que en este caso son, año de ingreso, cuota mensual y carrera. Y finalmente tenemos al miembro de Staff que también es una Persona pero con características propias de alguien que trabaja para una institución, por ejemplo salario y turno, este puede ser mañana o noche.

   Diagramar el UML identificando superclase y subclases, crear los constructores necesarios, los getters y setters, y los métodos de ayuda que consideres necesarios. Luego, en el main:
   - Inicializar 4 estudiantes diferentes.
   - Inicializar 4 miembros de staff con características diferentes.
   - Crear un array que permita almacenar juntos los tipos anteriores y almacenar las 8 instancias creadas anteriormente.
   - Investigar el uso de la palabra reservada instanceof.
   - Recorrer el array y mostrar por pantalla la cantidad de estudiantes y la cantidad de miembros de staff.
   - Recorrer el array y sumar el total de ingresos que percibe la institución por parte de la cuota de estudiantes.