**Repaso:**

**Conceptos Básicos de Objetos**

**Concepto de Clase:**

Representa un tipo de dato abstracto, y la diagramamos en UML como sigue:



Luego, a la hora de codificar el sistema cada clase se plasma en código fuente sobre algún lenguaje que soporte orientación a objetos.

**Casos Ejemplo de Clases:**

|  |  |
| --- | --- |
|  | class **Coordenada** {  private float latitud; //un atributo  private float longitud; //otro atributo    public void mostrar() { // método que muestra los atributos  System.out.println( latitud );  System.out.println( longitud );  }  } |

|  |  |
| --- | --- |
|  | class **Cliente** {  public int codigo;  prívate String nombre;  prívate String direccion;    public void mostrar() { // método que muestra los atributos  System.out.println( Nombre + “ vive en “ + direccion );  }  public void mostrar() { // método que muestra los atributos  System.out.println( “enviando mail a: “ + nombre );  }  } |

|  |  |
| --- | --- |
|  | *class Habitacion {*  *public float largo;*  *public float ancho;*  *private int numero;*    *public Habitacion(float l, float a, int n){ //constructor*  *largo = l; //sirve para inicializar atributos*  *ancho = a;*  *numero =n;*  *}*  *}* |

|  |  |
| --- | --- |
|  | *class NumeroComplejo {*  *private float parteReal;*  *private float parteImaginaria;*  *public float getReal() {*  *return parteReal;*  *}*  *public void setReal(float entrada) {*  *parteReal = entrada;*  *}*  *public float getImaginaria() {*  *return parteImaginaria;*  *}*  *public void setImaginaria(float entrada) {*  *parteImaginaria = entrada;*  *}*  *}* |

|  |  |
| --- | --- |
|  | *class Pelicula {*  *public int legajo;*  *public String nombre;*  *private String direccion;*  *private float dni;*    *public Boolean darAlta(){*    *//...aca viene codigo para dar de alta*  *return true;*  *}*  *}* |

|  |  |
| --- | --- |
|  | *class Profesor {*  *public int legajo;*  *public String nombre;*  *private String direccion;*  *private float dni;*    *public Boolean darAlta(){*    *//...aca viene codigo para dar de alta*  *return true;*  *}*  *}*  *class Alumno {*  *public int legajo;*  *public String nombre;*  *private String direccion;*  *private float dni;*    *public Boolean inscribirEnMateria(){*    *//...aca viene codigo para inscribir*  *return true;*  *}*  *}* |

**Si observamos las clases anteriores, vemos que podemos especificar un poco más lo que es una *dirección*. Por lo tanto creamos el nuevo tipo (clase). Y a continuación lo relacionamos con la clase *Profesor* y *Alumno*.**

|  |
| --- |
|  |
| *class Direccion {*  *public String calle;*  *public int numero;*  *private int localidad;*  *private String provincia;*  *private String pais;*    *public void mostrar(){*  *System.out.println("Calle " + calle + " " + numero);*  *}*  *}*  *class Profesor {*  *public int legajo;*  *public String nombre;*  ***private Direccion direccion;***  *private float dni;*  *public Boolean darAlta(){*  *//...aca viene codigo para dar de alta*  *return true;*  *}*  *}* |

**Por otro lado, también podemos mejorar el modelo, utilizando una superclase (o clase madre) para que Profesor y Alumno hereden los atributos comunes de ella.**

|  |
| --- |
|  |
| ***class Persona {***  ***public int legajo;***  ***public String nombre;***  ***protected float dni;***  ***}***  *class Profesor* ***extends Persona*** *{*  *public Boolean darAlta() {*  *//...aca viene codigo para dar de alta*  *return true;*  *}*  *}*  *class Alumno* ***extends Persona*** *{*  *public Boolean inscribirEnMateria() {*  *//...aca viene codigo para inscribir*  *return true;*  *}*  *}* |

**Luego, utilizar el tipo (clase) Dirección que escribimos anteriormente para relacionarlo con Persona, y que a su vez pueda ser heredado por Profesor y Alumno**

|  |
| --- |
|  |
| ***class Persona {***  ***public int legajo;***  ***public String nombre;***  ***protected float dni;***  ***}***  *class Profesor* ***extends Persona*** *{*  ***private Direccion direccion;***  *public Boolean darAlta() {*  *//...aca viene codigo para dar de alta*  *return true;*  *}*  *}*  *class Alumno* ***extends Persona*** *{*  ***private Direccion direccion;***    *public Boolean inscribirEnMateria() {*  *//...aca viene codigo para inscribir*  *return true;*  *}*  *}* |

|  |
| --- |
|  |
| *class Persona*  *protected int legajo;*  *protected String nombre;*  *protected float dni;*  *}*  class Actor extends Persona  private String Nacionalidad;  public void setNacionalidad(String Nacionalidad) {  this.Nacionalidad = Nacionalidad;  }  public String getNacionalidad(){  return Nacionalidad;  }  class Pelicula  private int Genero;  private boolean Alquilada;  private Actor actor[]; //RELACIÓN ENTRE CLASES  public void alquilar(){  Alquilada = true;  }  public void recibir(){  Alquilada = false;  }  }  //también habría que escribir los métodos get y set para cada atributo privado |

**Preguntas:**

1. **La clase Persona se podría calificar como abstracta (abstract) ? que pasaría ?**
2. **La clase Persona se podría calificar como *final* ?**
3. **La clase Actor se podría calificar como abstracta ?**
4. **La clase Actor se podría calificar como *final* ?**