TALLER DE LENGUAJE DE PROGRAMACIÓN I

Entorno de CONSOLA y GRÁFICOS

UNIDAD 2

Programación Orientada a Objetos – POO

Encapsulamiento - Composición

JAVA

OBJETIVO

- Formas de Programación
  - Orientado a Procedimientos (POP)
  - Orientado a Objetos (POO)
- Orientado a Objetos (OO)
  - Naturaleza de los Objetos
    - Estado, Comportamiento, Propiedades
  - Ventajas y Desventajas
- Concepto de Clase y Objeto (POO)
- Ejercicio practico en Eclipse
- Encapsulación
- Composición

Trabajo Independiente

- Ejercicio

Formas de Programar

Orientados a Procedimientos

- Surgen en los años 60's y 70's
- Se definen Procedimientos y Funciones.
- Los primeros lenguajes de programación procedimentales importantes aparecieron alrededor de 1957-1964, incluidos Fortran , ALGOL, Cobo l , PL /I y BASIC .
- Pascal y C se publicaron alrededor de 1970-1972.

Orientados a Objetos

- Concepto de Objeto como estructura.
- Mejora la administración de lo complejo.
- Implementa los conceptos de:
  - Herencia.
  - Cohesión.
  - Abstracción.
  - Polimorfismo.
  - Acoplamiento.
  - Encapsulamiento.
- Popularizados en los 90's

Modularización

Reutilización de código

Evaluación del código según comportamiento de Objetos

Orientación a Objetos (OO)

Orientados a Objetos

- Naturaleza de un Objeto
  - **Estado**.
    - Un computador puede estar encendido, apagado, desconectado
  - **Propiedades**
    - Escritorio, Color, Alto, ancho, temperatura, etc.
  - **Comportamiento** (Qué puede hacer?).
    - Procesar, Enviar (impresión), Almacenar, etc.

Orientación a Objetos (OO)

Ventajas y Desventajas de POO

- Ventajas
  - Descompone un problema
    - Modulariza.
    - Reutiliza (Herencia).
  - Programas más fáciles de manejar.
  - Orden y Legibilidad.
  - Manejo de Errores (Excepciones)
  - Encapsulamiento
    - (El chasis no sabe que hace la board)
- Desventajas
  - No recomendable para tareas fáciles
  - La ejecución de los programas puede ser lenta.
  - Su curva de aprendizaje puede ser desafiante.

Concepto de Clase y Objeto en POO

Clase

Es la descripción que se hace de un conjunto de entidades similares; consta de un **estado** (métodos constructores), **acciones** (métodos de acción) y de **características** (atributos) que resumen la especificación del conjunto de entidades.

Objeto

Es la instanciación o referencia especifica de una entidad de clase, que adquiere todos los estados, acciones y características de la clase que lo define, haciendo identificable la “Entidad”.

Ejemplo Clase

Mid-Tower (ATX)

El Chasis es utilizado por diferentes marcas para ensamblar un computado.

Características:

- Placa base Mini ITX, Micro-ATX y ATX (30cm x 24cm)
- Ranuras 8 para expansión
- Administración de cables
- Alto de 45 cm
- Ancho 60 cm
- Profundidad 83 cm

Concepto de Clase y Objeto en POO, continuación

Ejemplo Objeto

Clase Chasis - Mid-Tower (ATX)

Clase

Computador Marca A

Objeto A

Computador Marca B

Objeto B

Concepto de Clase y Objeto en POO, continuación

Ejemplo Objeto aplicado a código

1. Se definen paquetes para organizar la aplicación en un encarpetado.
2. Se define una clase (chasis) que será utilizada como plantilla para ensamblar un computados.
3. Se define el objeto específico de la clase (chasis).

Concepto de Clase y Objeto en POO, continuación

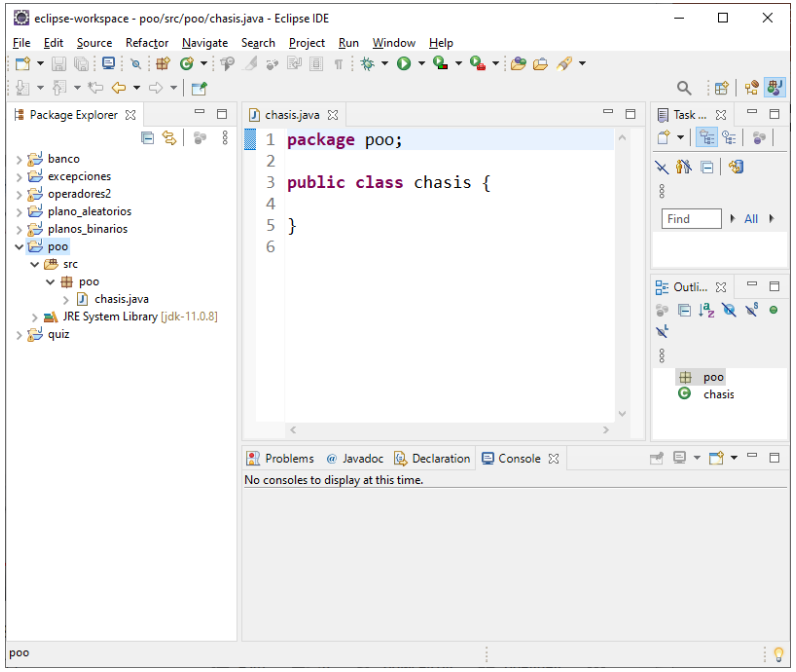
Ejemplo Objeto aplicado a código

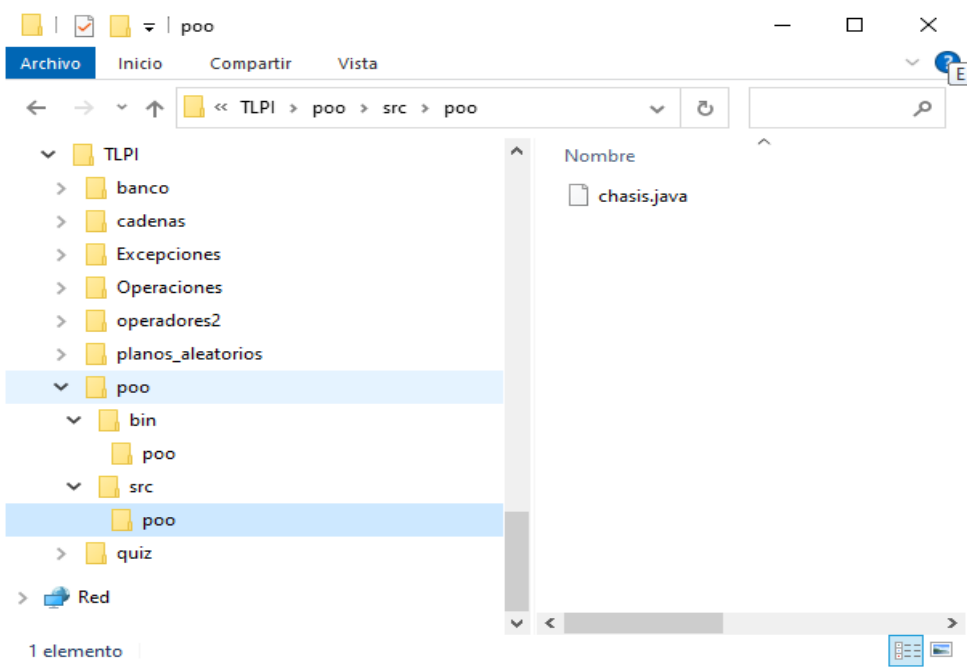
- Se define un proyecto, para el ejemplo llamado poo.

  Siempre debe ser en minúscula y sin espacios
- Se define un paquete, para el caso se llamara poo.
- Se define una clase sin el método main, con nombre Chasis.

Concepto de Clase y Objeto en POO, continuación

Ejemplo Objeto aplicado a código





Concepto de Clase y Objeto en POO, continuación

Ejemplo Objeto aplicado a código

En la clase Chasis, se definen todas las características o atributos que tienen las entidades que se agrupan en la clase.

Concepto de Clase y Objeto en POO, continuación

Ejemplo Objeto aplicado a código

```
package poo;
public class Chasis
{
        int     largoPlaca;
        int     anchoPlaca;
        int     ranuras;
        boolean adminCables;
        int     ancho;
        int     alto;
        int     profundidad;
}
```

La clase queda incluida dentro del paquete poo

Propiedades de la clase, que son las características comunes de las Entidades agrupadas en la clase

Concepto de Clase y Objeto en POO, continuación

Ejemplo Objeto aplicado a código

Se define el estado inicial de las características o atributos de la clase, para esto se define un método de tipo constructor. Con este tipo de métodos se definen los posibles estados de la clase.

Concepto de Clase y Objeto en POO, continuación

Ejemplo Objeto aplicado a código

```
package poo;
public class Chasis
{
    int         largoPlaca;
    int         anchoPlaca;
    int         ranuras;
    boolean     adminCables;
    int         ancho;
    int         alto;
    int         profundidad;

    public Chasis ()
    {
        largoPlaca          =30;
        anchoPlaca          =24;
        ranuras             =8;
        adminCables         =true;
        ancho               =60;
        alto                =45;
        profundidad         =83;
    }
}
```

Debe tener el mismo nombre de la clase

Estado inicial de los atributos de la clase, datos iniciales que no deberían poder ser modificados desde otra clase.

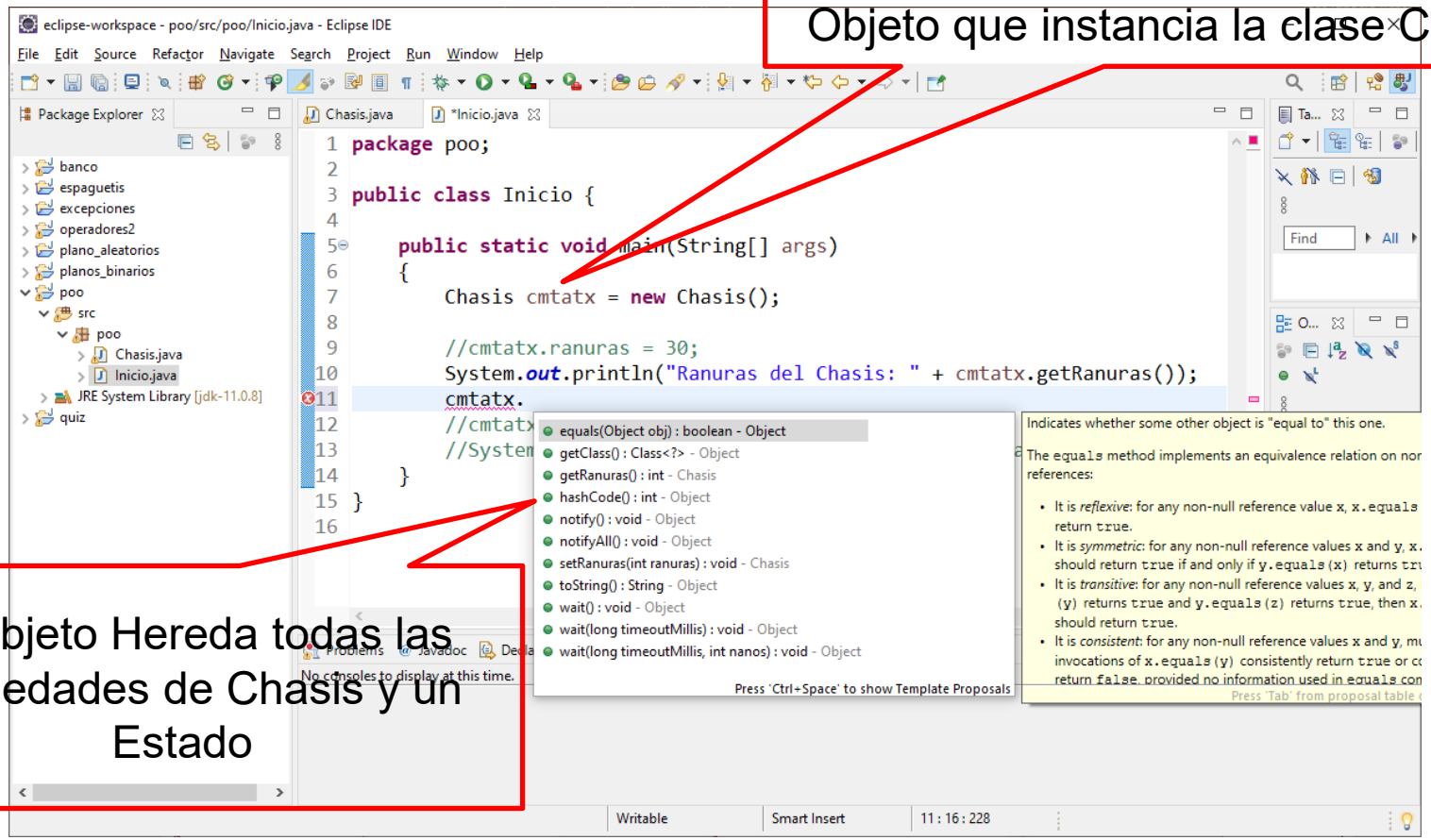
Concepto de Clase y Objeto en POO, continuación

Ejemplo Objeto aplicado a código

Se define una clase con el método main, que instancie la clase Chasis definiendo un Objeto, que Hereda todas los **atributos** de Chasis, un **Estado** inicial y que podrá **comportarse** como un Chasis.

Concepto de Clase y Objeto en POO, continuación

Ejemplo Objeto aplicado a código



Concepto de Clase y Objeto en POO, continuación

Ejemplo Objeto aplicado a código

```
package poo;

public class Inicio {

        public static void main(String[] args)
        {
                Chasis cmtatx = new Chasis();

                cmtatx.ranuras = 30;
                System.out.println("Ranuras del Chasis: " + cmtatx.ranuras);
        }
}
```

Definición del Objeto

Violación de Datos o Integridad de Información

La violación de Datos o Violación de Integridad de Información, se soluciona con el ENCAPSULAMIENTO de los atributos

Encapsulación en POO

Encapsulación

Consiste en la ocultación de los atributos que describen al grupo de Entidades (Clase), de forma que sólo es posible modificar los mismos mediante los métodos (Forma de comunicación entre clase).

Para tal fin, se define el modificador de acceso a los atributos y se crean los métodos necesarios para entregar (get) y recibir (set) datos, desde la clase al objeto.

Encapsulación en POO, continuación

Ejemplo Objeto aplicado a código

```
package poo;

public class Chasis
{
   private int  largoPlaca;
   private int  anchoPlaca;
   private int  ranuras;
   private boolean adminCables;
   private int  ancho;
   private int  alto;
   private int  profundidad;

   public Chasis ()
   {
      largoPlaca          =30;
      anchoPlaca          =24;
      ranuras             =8;
      adminCables         =true;
      ancho               =60;
      alto                =45;
      profundidad         =83;
   }
}
```

Se modifica la forma de acceder de público a privado para ENCAPSULAR

Solo es posible acceder a los datos en la misma clase Chasis. Para permitir ver los datos desde otra clase, es necesario definir métodos de acceso Getter y Setter

Encapsulación en POO, continuación

Ejemplo Objeto aplicado a código

```
package poo;
public class Chasis
{
    private int largoPlaca,anchoPlaca, ranuras, ancho, alto, profundidad;
    private boolean adminCables;

    public Chasis ()
    {
      largoPlaca      =30;
      anchoPlaca      =24;
      ranuras         =8;
      adminCables     =true;
      ancho           =60;
      alto            =45;
      profundidad     =83;
    }
    public int getRanuras()
    {
        return ranuras;
    }

    public void setRanuras(int ranuras)
    {
        this.ranuras = ranuras;
    }
}
```

Tener presente que es un parámetro con el mismo nombre del atributo

Composición en POO

Composición

Consiste en crear una clase nueva agrupando objetos de clases que ya existen. Una composición agrupa uno o más objetos en la definición del objeto, de manera que las instancias de esta nueva clase contienen uno o más objetos de otras clases. Normalmente los objetos contenidos se declaran con acceso private y se inicializan en el constructor de la clase

Composición en POO, continuación

Ejemplo composición aplicado a código

```
package poo;
public class DiscoDuro
{
        private String tipoDisco; //Define si es magnetico o SSD
        private String interfaz; //IDE, SATA, SCSI, SAS, SATA Express
        private int      capacidad; //Capacidad de almacenamiento en GigaByte

        public DiscoDuro()
        {
                tipoDisco = "Magnetico";
                interfaz  = "SATA";
                capacidad = 100;
        }
        public String getInterfaz() {
                return interfaz;
        }
        public void setInterfaz(String interfaz) {
                this.interfaz = interfaz;
        }
}
```

Se debe definir la totalidad de los Métodos getters y setters

Encapsulación en POO, continuación

Ejemplo composición aplicado a código

```
package poo;
public class ArmarComputador
{
		private Chasis chasis;
		private DiscoDuro dd;

		public ArmarComputador(Chasis chasis, DiscoDuro dd)
		{
				this.chasis = chasis;
				this.dd = dd;
		}
		public Chasis getChasis() {
				return chasis;
		}
		public void setChasis(Chasis chasis) {
				this.chasis = chasis;
		}
		public DiscoDuro getDd() {
				return dd;
		}
		public void setDd(DiscoDuro dd) {
				this.dd = dd;
		}
}
```

Encapsulación en POO, continuación

Ejemplo composición aplicado a código, parte 1

```
package poo;

public class Inicio {

		public static void main(String[] args)
		{
				Chasis cmtatx = new Chasis();
				DiscoDuro dd = new DiscoDuro();

				ArmarComputador computador = new ArmarComputador(cmtatx, dd);

				System.out.println("El computador tiene la siguiente configuración Básica");

				System.out.println("Chasis con dimensiones: Alto: " + computador.getChasis().getAlto() +
								"cm Ancho: " + computador.getChasis().getAncho() +
								"cm Profundidad: " + computador.getChasis().getProfundidad() +
"cm");

				System.out.println("Ranuras: " + computador.getChasis().getRanuras());

				System.out.println("Didco Duro: " + computador.getDd().getTipoDisco() +
								" con interfaz: " + computador.getDd().getInterfaz()  +
								" de almacenamiento: " + computador.getDd().getCapacidad() + "
GB");
```

Encapsulación en POO, continuación

Ejemplo composición aplicado a código, parte 2

```
                dd.setCapacidad(120);

                computador.setDd(dd);

                System.out.println("Didco Duro: " + computador.getDd().getTipoDisco() + "
con interfaz: " + computador.getDd().getInterfaz() + " de almacenamiento: " +
computador.getDd().getCapacidad() + " GB");

                //cmtatx.ranuras = 30;
                //System.out.println("Ranuras del Chasis: " + cmtatx.getRanuras());
                //cmtatx.setRanuras(10);
                //System.out.println("Ranuras del Chasis: " + cmtatx.getRanuras());
        }
}
```

Trabajo Independiente

- Desarrollar los ejercicios del Taller de modelamiento e identificar y crear los proyectos por cada uno de ellos, definiendo clases, objetos, métodos constructores, métodos modificadores de datos y construir composiciones.