**SEMANA DOS**

* Función Main:
* Paradigmas = Estructurada, POF, POO (dependiendo de la forma en que se decida programar se usa un lenguaje funcional, POO=java-python, POF/estructurada=Python)
* Herramientas (IDE (herramienta donde se escribe el código, Lenguaje y su lógica)

Todo objeto en el universe se distingue por dos razones:

1. Atributo
2. Acciones/comportamiento

Metodología UML = Unify Modeler language -> modelación de software. Se modela de la misma manera para poder globalizar la forma de modelar

**¿Como imaginar un objeto en programación?**

* Módulos/métodos: lo que permite hacer las tareas
* DC = datos compartidos para que los módulos puedan realizar las tareas, hay datos que todos o varios módulos necesitan

**UML + documentación ----> MOLDE -----------🡪Objeto------------------🡪 Tarea**

Atributos Clases Usando el molde

Metodos

**¿CÓMO SE CONSTRUYE UNA CLASE?**

* Referencia al paquete = package 🡪 se incluye automáticamente gracias a los IDE
* Importaciones de clase propias de java necesarias para el trabajo de la clase
* Public class NombreClase 🡪 modificador de acceso (public/private/protected) 🡪variables, metodos, clases exigen modificadores de acceso
* Variables de calse ( cualquier método de la clase puede acceder) o atributos. Constantes
* Modificador de acceso, tipo de dato, variable 🡪 private int x;
* Nadie puede acceder directamente al valor cuando esta private, al menos que se haga un método especial para hacerlo.

**TIPOS DE METODOS**

* **Método principal** (si lo hay) 🡪 public static void main (String []) – obligatorio 🡪 es la puerta de entrada a todo, la primera función que busca el copilador es el metodo main. Se pone en la clase que inicia el proyecto
* Se puede acceder de cualquier parte
* Static = me deja entrar a trabajar con la clase sin haber creado objetos todavia
* **Métodos constructores** – obligatorio 🡪 permite la construcción del obeto / pueden tener parámetros o no tenerlos
* inicializa el objeto, pone valores iniciales a la variables y los atributos
* llamar metodos locales
* es la puerta de entrada al objeto
* public NombreClase (){
* **Métodos de construcción de la interfas**
* **Métodos analizadores 🡪** analizan el estado de un objeto, obtiene el valor de un atributo, se programa uno por cada atributo que se necesita acceder, son los únicos en capacidad de acceder a los atributos privados
* Deben ser públicos para poder accederlos desde cualquier parte del proyecto
* Tipo de retorno
* Get (atributo) 🡪 atributo que esta buscando
* Retorna el atributo
* Public int get(atributo) {

return atributo;

}

* **Metodos modificadores**
* Metodos de servicio 🡪 son los encargados de hacer las tareas
* Tiene un modificador de acceso 🡪 public void setAtributo (n
* Tipo de retorno
* Nombre del metodo
* **Public void nombre (parámetros) {**

Variables locales

Programación de la tarea

Valor de retorno

* **Metodos de manejo de eventos 🡪**  cuando el usuario hace un click tiene que ocurrir algo o alguna acción

**¿Cuándo se deja publico y cuando se deja privado el metodo?**

* Cuando el metodo va a ser invocado dentro del mismo objeto se deja privado, si va a ser invocado desde otro objeto se deja publico

**ARQUITECTURA DE CAPAS**

* 3 capas

**3.** Vista 🡪 lo que ve el usuario y con lo que interactúa, interfaz gráfica FX

**2.** Lógica 🡪 donde se pone todo el código, backend, elección de paradigma y lenguaje

**1.** datos, guarda datos 🡪 sistema de archivos de texto

Documentación:

* Mapa de objeto: pintar módulos – robot 🡪es anonimo
* Diagrama de clases (UML 🡪 start uml): lo que esta subrayado es sttatic; + publico, - privado

**SEMANA 3**

Clase elemental: solo tiene tres metodos: main, constructor y un metodo de servicio. Se usa cuando el proceso involucra tareas sencillas que se pueden acomodar en un mismo objeto y metodo.

* El main ni el constructor tienen porque hacer tareas.
* Determinación de cuantos y que tipo de objetos: se aconseja realizar el mapa de objetos
* Programación de plantillas para los objetos
  + Public clase Suma{

}

* + Public static void main

notación camelcase

* No es conveniente que un método haga más de una tarea, facilitando el mantenimiento de los sistemas.

**NIVEL DOS.**

Comunicación entre metodos:

* + Envío entre datos
  + Retorno de datos

Invocación a los metodos

* + Llamamiento escalonado

Para poner a retornar un metodo

* Poner el retunr
* poner tipo de dato que retorn

**NIVEL TRES = ESPECIALIZACIÓN DE OBJETOS**

Se hacen grupos de tareas que sean parecidas y así mismo se crean los objetos. Por es mimsmo se pueden necesitar más de una clase.

HAY QUE TENER CUIDADO CON LA COMUNICACIÓN ENTRE OBJETOS

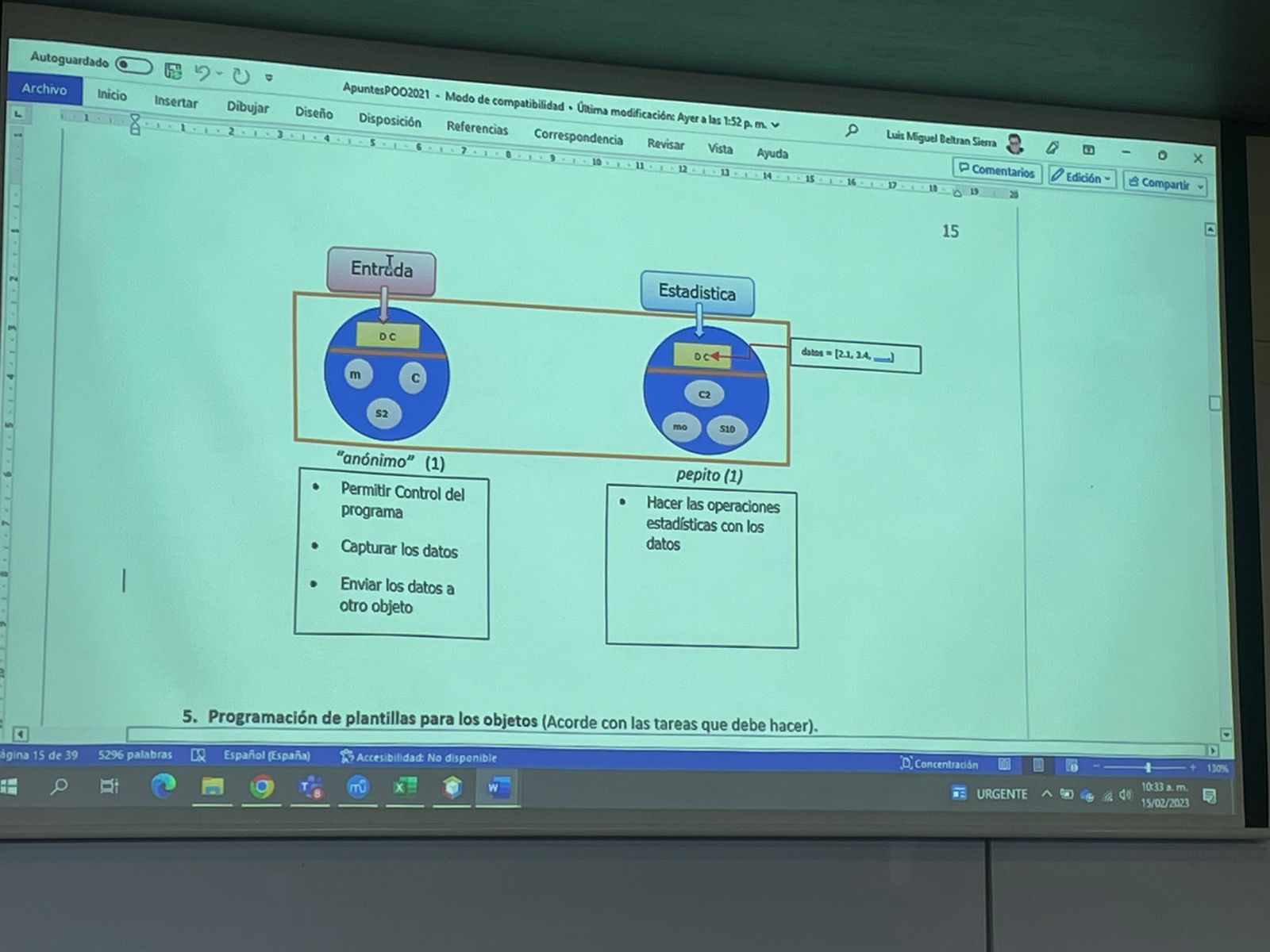
* Envío y retorno de información

1. Metodo constructor
2. Metodo de modificación
3. Cualquier metodo

* Responder: ¿Dónde y cuando crear los nuevos objetos?

*EJERCICIO NIVEL3*

* Ver archivo manejoDatos2.bmp
* Identificaciones de trabajo: 12 tareas ( las 11 relacionadas con manejo de información y una con darle el control al usuario)



* Si se necesita llamar un objeto desde otro, a huevo el que se llama tiene que tener nombre

1. Estructura del objeto

* Programación de plantillas para los objetos

Graphical user interface, application, Teams

Description automatically generatedCuando un objeto A tiene dentro de sus atributos o variables de clase a otros objeto B, existe entre A y B una relación de tenencia.

**SEGUNDO CORTE**

Nivel 4: Soporte a negocios

1. Entidad de negocio: atributos (información que se requiere guardar), parte del negocio. A lo que se dedica el negocio (producto/servicio) ¿Qué quiere guardar de las entidades negocio? (atributos)

* Habrá múltiples entidades, ES TODO DE LO QUE UNA EMPRESA NECESITE GUARDAR INFORMACION. Producto/Proveedor/Cliente: los atributos les pertenecen
* Representan al cliente en el programa

1. Operaciones de negocio: procesos que se realizan con la entidad. Ventas, compras, reservas, detalles de ventas

* Ejemplo Agenda
* Estudiante 🡪 entidad de negocio (tributos, constructores, analizadores, modificadores, ver objeto)

1. Código
2. Nombre
3. Genero
4. Edad
5. Notas

Operaciones CRUD= Create, read, update, delete (SE USAN EN TODOS LOS NEGOCIOS)

Los objetos se crean en la memoria RAM (cada vez que se sale del código se pierden los datos guardados)

* COLECCIONES:

Para las colecciones de objetos no se recomiendan estructuras estáticas como los arreglos, son mejores las dinámicas.

PARA MANEJAR COLECCIONES SE REQUIERE DE CONTENEDORES:

1. Estático: primitivos, objetos (no se recomienda)
2. Dinámicos: dependen del lenguaje de programación: list, arraylist, vector,

* ArrayList:

Gestores: gestionan colecciones, es otro objeto. Administran los contenedores; arrayList

NIVEL 5.

* Arquitectura básica
* Vista
* Lógica (java, POO), objetos de negocio, objetos de gestión, objetos de interacción
* Datos (bases de datos, Excel, txt) guarda datos y no se pierden

Que lo mande a un archivo de txt plano.

Estructura de datos: variables: simples, arreglos.

* Variables de tipo objeto: atributos

Todo se guardaba en la RAM y por eso la información se borraba

Disco: se guarda en tipos de archivos (txt, pdf, rar). Se usara txt para guardar datos de objetos

Exportación de datos en la RAM a el DISCO (escritura, exportación) y viceversa (lectura, importación)

Ruta relativa: arranca donde esta el proyecto, pero como no se sabe donde se va a colocar el proyecto el cliente se pone “ ./ “

Ruta absoluta: solo fu

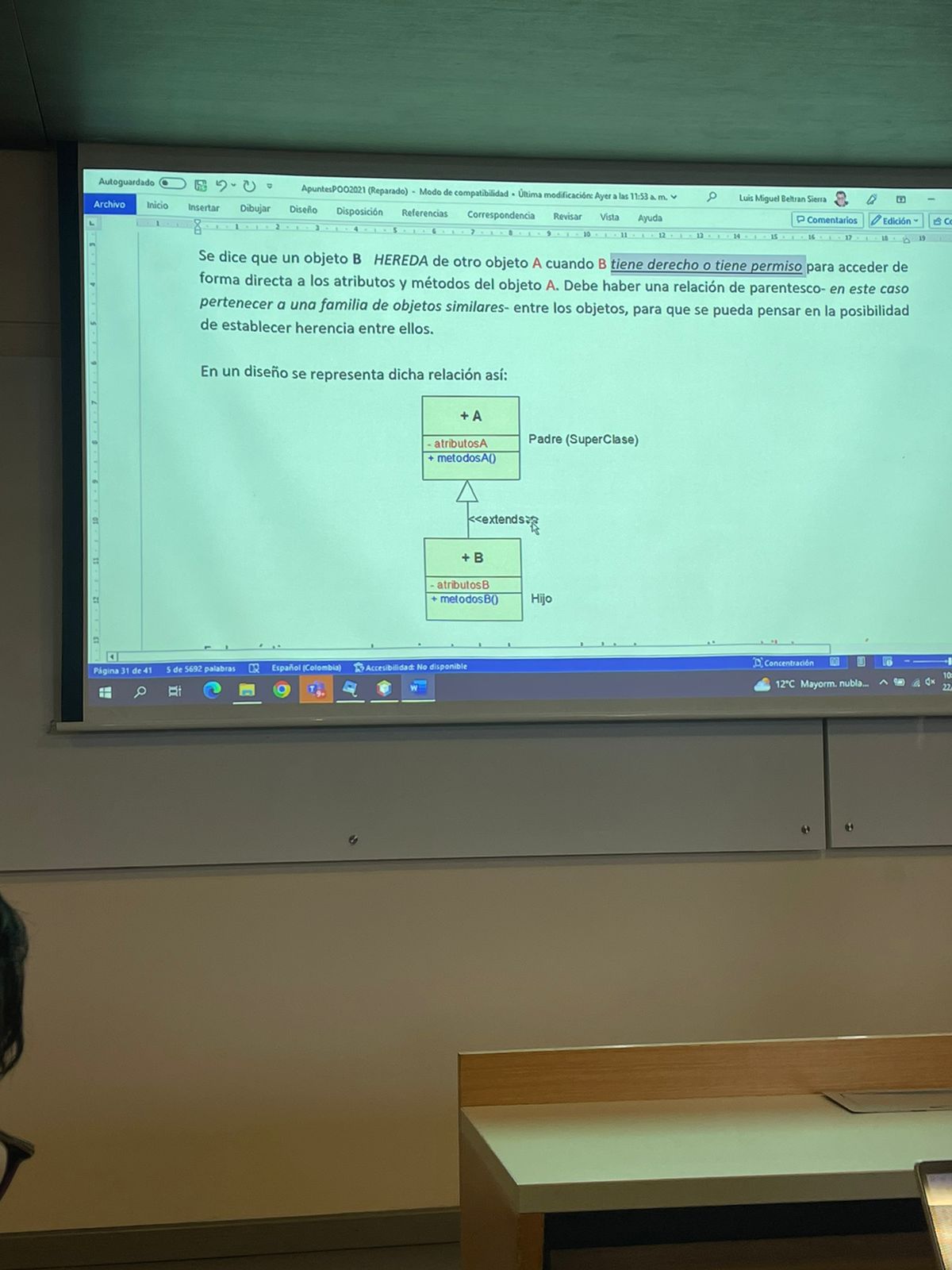
RETO:

* Estudiante (id,nombre,edad)
* Asignaturas(cod,nombre,créditos)
* Inscripciones(idEst, codAsig, arreglo notas)

OPTIMIZACIÓN DE CODIGO:

* Herencia
* Clases abstractas
* Interfaces

HERENCIAS: un objeto B(hijo, subclase) hereda de un objeto A (padre, superclase)cuando B tiene derecho o permiso para acceder directamente a todo lo que tenga el objeto A (metodos y atributos)



* Tienen cosas en común, son como de la misma familia.
* Se dejan los atributos y/o metodos que ambos compartan se dejan en la superclase y los que son únicos de clases se dejan en las clases hijos. No se deja que una clase tenga mas de una superclase.

QUE UNA CLASE TENGA LO NECESARIO PARA PODER SER LA CLASE PADRE DE TODAS LA FIGURAS

FIGURA:

* X
* Y
* Dimen1
* Dimen2
* Dimen3

Polimorfismo= que adquiere diferentes formas

INTERFASES: hay metodos que no puedo implementar porque dependen de cada hijo. Esos metodos se mandan a una interface, implementacion

RELACIONES

* Usabilidad
* Tenacidad
* Implementación