Objetivos:

- Repaso IO
- Aplicar conceptos de Orientación a Objetos en Java
- El lenguaje y la plataforma Java, Objetos y Clases en Java
- Variables, métodos y constructores. Modificadores de acceso
- Diseño OO

*En esta materia veremos cómo aplicar los conceptos vistos en **Introducción a Objetos** utilizando el lenguaje de programación llamado **JAVA**.*

JAVA es un lenguaje orientado a objetos desarrollado por SUN, el cual implementa muchos de los conceptos que se vieron durante la primera mitad del año. Para más información sobre el mismo pueden dirigirse a www.java.sun.com

En Wikipedia pueden hallar una breve reseña histórica de JAVA, sus orígenes, creadores, etc: http://es.wikipedia.org/wiki/Lenguaje_de_programación_Java

Por otro lado, para poder trabajar con JAVA utilizamos un IDE (Integrated Development Environment) el cual nos facilitara el desarrollo de los trabajos prácticos.

*En este año **utilizaremos dos IDEs diferentes el cual iremos alternando en cada uno de los trabajos prácticos** con el fin de que los alumnos usen ambos:*

1. *Por un lado se utilizará Eclipse, desarrollado originalmente por IBM y liderado actualmente por la Fundación Eclipse. Para más información sobre el mismo pueden dirigirse a www.eclipse.org Eclipse tiene varias distribuciones según el tipo de proyecto que debamos desarrollar. En nuestro caso usaremos el paquete para Desarrollo Java Básico. Lo pueden descargar de: https://www.eclipse.org/downloads/*

2. *Otro de los IDEs que vamos a utilizar a lo largo del año es Netbeans. http://netbeans.org/. Sun MicroSystems fundó el proyecto de código abierto NetBeans en junio de 2000 y continúa siendo el patrocinador principal de los proyectos. Así como Eclipse, Netbean también cuenta con varias distribuciones y en nuestro caso vamos a utilizar la denominada "Java SE".*
   Lo pueden descargar de: http://netbeans.org/downloads/index.html

En éste trabajo práctico vamos a utilizar Eclipse como IDE de desarrollo.

Ejercicio 1. (Repaso)

a) Defina en JAVA la clase **Punto,** que posee un valor x y un valor y, ambos **privados**, es decir, no se pueden acceder desde afuera, e implemente los siguientes métodos:

```
public void sumar(int x){
 /*Suma el mismo valor a x e y*/
}

public void sumar(Punto unPunto){
/*Suma unPunto a los valores que tiene el punto*/
}

public int distancia(Punto unPunto){
/*Calcula la distancia por Pitágoras*/
}
```

*Para poder **probar** si todo funciona correctamente debemos hacer un método especial llamado **main**. Más adelante nos enfocaremos en este método, por ahora úsenlo de la siguiente forma:*

```
/**
 * @En una aplicación JAVA, el main es el método
 * principal que se ejecuta por defecto.
 */

public static void main(String[] args) {
   Punto p1=new Punto(1,3);
   System.out.println("La coordenada del punto es: "+p1.getX()+"@"+p1.getY());
}
```

*Para poder **ejecutar** el código desde el Eclipse haga click con el botón derecho del mouse sobre la clase que tiene el método **main** y seleccione Run As/Java Application (ver imagen 1). El resultado sé vera en la pestaña titulada **consola**.*

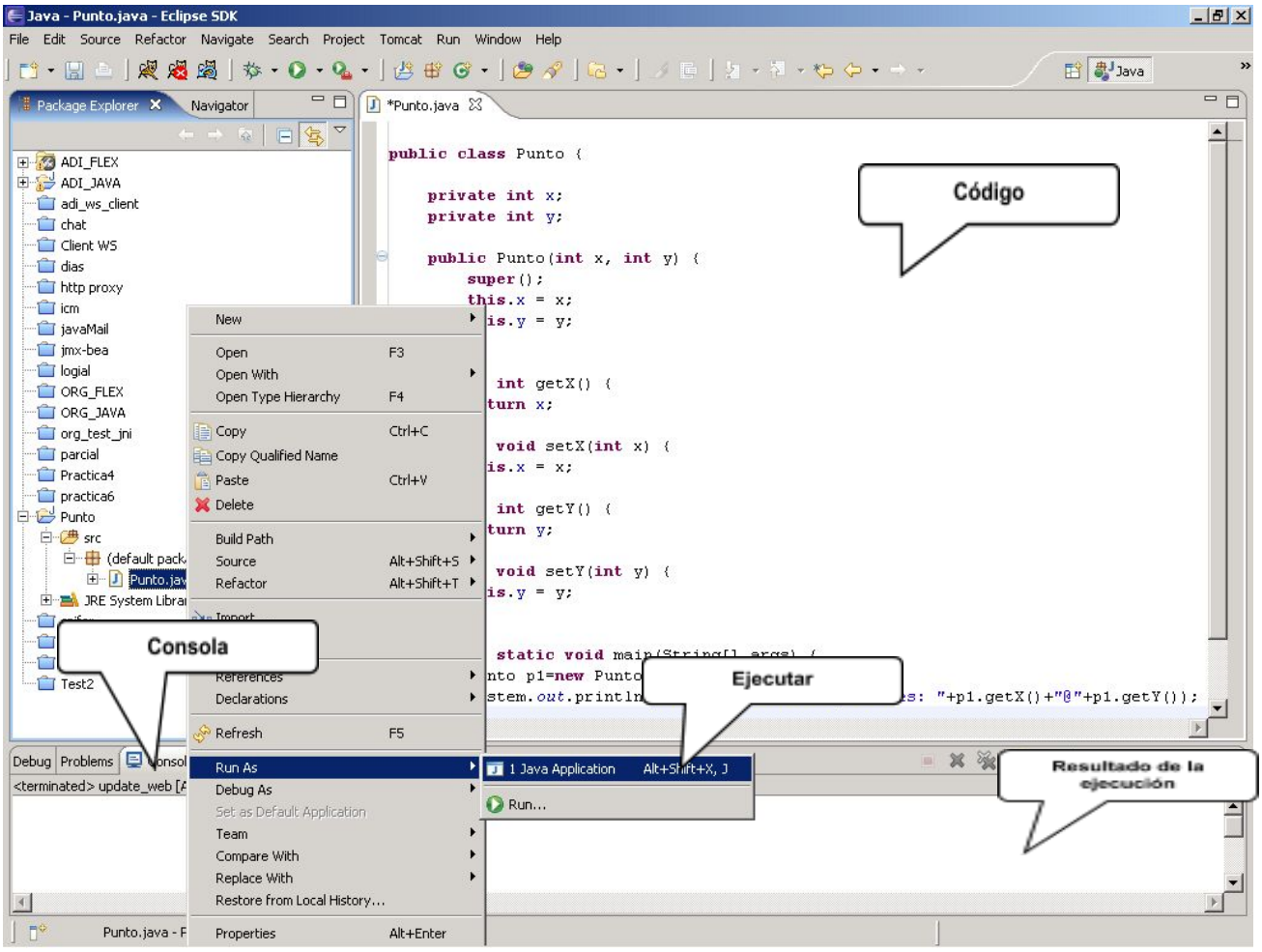

Imagen 1 – Ejecución de una aplicación Java desde Eclipse

b) Basándose en su código y en los métodos que se pide que implemente responda lo siguiente:

   a. ¿Por qué el objeto que se muestra como ejemplo tiene el valor 1@3?
   b. ¿Que es necesario para hacer dicha inicialización del objeto?

Nota: Para poder imprimir resultados en la consola se puede utilizar el siguiente código:

```
System.out.println(valor);
```

Donde ***valor*** puede ser un String, un Entero o algún otro tipo de dato.

Ejercicio 2. (Repaso)

Implemente en JAVA una **CuentaBancaria** con sus respectivas subclases **CajaDeAhorro** y **CuentaCorriente** las cuales permiten extraer y depositar dinero. Ambas poseen un saldo, el cual se puede consultar. Además, ambas poseen un limiteMinimo de extracción, es decir que el saldo no puede quedar menor a eso. Mientras que en la caja de ahorro el limiteMinimo es de 10, en la cuenta corriente es de -150. La caja de ahorro tiene un máximo de extracciones (inicialmente 5 y se "resetea manualmente una vez al mes") y la cuenta corriente no.

Ambas cuentas tienen un titular (uno y solo uno) que es de la clase **Persona** (La cual posee todos los atributos de una persona) y una fecha de apertura de cuenta.

Por otro lado, se desea llevar un control de las transacciones realizada, es por ellos que cada vez que se realice una transacción, la misma se debe guardar en la cuenta. De cada transacción se conoce la fecha, el monto y el tipo de transacción.

Ejercicio 3.

Una pequeña cooperativa de transporte quiere organizar el funcionamiento de sus unidades de colectivos:

- La cooperativa está formada por varios socios, todos chóferes, y cada uno maneja un colectivo diferente.
- La cantidad de pasajeros es diferente entre un colectivo y otro.
- Se registra la patente y el modelo de cada vehículo.

Se pide que implemente los siguientes métodos de la clase *Cooperativa*:

```
public Chofer realizarViaje(int cantPasajeros, float cantKmts) {}
/*Devuelve el chofer de la cooperativa que pueda realizar el viaje para la
cantidad de pasajeros solicitada y registra el kilometraje del viaje.*/

public Colectivo mayorKilometro() {}
 /*Devuelve el colectivo con mayor kilometraje acumulado.*/
```

Ejercicio 4.

Una empresa de logística desea automatizar el control de peso y valor de artículos que envían en sus contenedores. La empresa maneja muchos contenedores. La misma está compuesta por un nombre, una dirección y los contenedores.
El contenedor tiene un número de identificación, un país destino (hacia donde se envía) y un peso máximo de carga.
Los contenedores contienen artículos, los cuales pueden ser o productos simples o cajas de productos que a su vez contienen productos simples.

Los productos simples están compuestos por, un nombre, una fecha de vencimiento y un valor. Las cajas están compuestas por muchos productos simples el valor de la caja está dado por la suma de sus productos simples.

1. Realice el diagrama de clases en UML.
2. Desarrolle en JAVA-like todos las clases y métodos necesarios teniendo en cuenta que el sistema debe tener como mínimo la siguiente funcionalidad:

a) */* Retorna el peso del artículo. */*
public float **pesoArticulo**(Articulo un Articulo)

b) */* Retorna el peso del contenedor. Sumando las cajas y los productos simples. */*
public float **pesoContenedor**(Contenedor unContenedor)

c) */* Retorna la suma de todos los contenedores. */*
public float **pesoContenedores**()

d) */* Retorna el valor de todos los artículos del contenedor. */*
public float **valorDeLosArticulos**(Contenedor unContenedor)

e) */* Retorna la suma de todos los valores de los contenedores */*
public float **valorDeTodosLosArticulo**()

Ejercicio 5.

Se desea modelar un **cliente de correo** electrónico simple símil al *Outlook exprés, Windows mail, Thunderbird*, etc.
El mismo está simplificado, así que solamente maneja una sola **cuenta** de **correos** y **contactos**.

Los correos estar en una carpeta de correo recibido o de correo enviado según sea el caso. De cada correo se conoce el asunto, el texto, si está leído o no, el contacto origen y los contactos destino (uno o varios).

De la cuenta se conoce el nombre del usuario, la dirección de mail, la dirección del servidor de entrada y la de salida. De cada contacto se conoce su nombre y dirección de mail.

El cliente de correo tiene la siguiente funcionalidad:

- Cantidad de correos -> *retorna la cantidad total de correos (recibidos + enviados)*
- Cantidad de correos recibidos -> *retorna la cantidad total de correos recibidos*

- Cantidad de correos enviados -> *retorna la cantidad total de correos enviados*
- Cantidad de correos no leídos (solo de los recibidos) -> *retorna la cantidad total de correos no leídos de la carpeta recibidos*
- Cantidad de contactos - > *retorna la cantidad total de contactos*
- AgregarCorreoRecibido(Correo unCorreo) -> *Agrega un correo a la carpeta de recibidos*

a) Diseñe en UML una solución al enunciado.

b) Implemente en JAVA la solución propuesta en a).

Ejercicio 6.

Se desea modelar el objeto Clima, el mismo está compuesto por los datos del clima actual. Al menos se conoce:

- La ciudad, provincia y país del clima.
- La temperatura actual.
- La mínima y máxima del día.
- La escala (C o F)
- La presión
- El viento y la dirección del mismo
- La Humedad
- La Visibilidad
- La hora de la última actualización

a) Diseñe en UML una solución al enunciado.

b) Implemente en JAVA la solución propuesta en a)

Ejercicio 7

Un banco internacional se encarga de gestionar transacciones de dinero entre diferentes bancos nacionales o internacionales.
Del banco se conoce, el nombre, la ubicación (incluido el país), las transacciones y los clientes del mismo.

Las transacciones pueden ser nacionales o internacionales y están compuestas por, una fecha, un banco destino, un monto y el cliente origen (es decir, el cliente del banco origen).
En el caso de las transacciones nacionales, el monto de la misma esta expresados en peso, pero en el caso de las transacciones internacionales el monto está expresado en la

moneda del banco destino, con lo cual se debe tener el tipo de cambio (la moneda) de dicho banco, por ejemplo: *Dólar (U$D), Euro (€), Yen (¥).*

Se trabaja únicamente con tres monedas y cada moneda tiene el siguiente valor de cambio actual:

Moneda	Pesos
1 Dólar	27.09 Pesos
1 Yen	0.24 Pesos
1 Euro	31.49 Pesos

Recuerde que el valor de la moneda es fijo, es decir, una vez que se crea la transacción y se le asigna la moneda está y su valor al cambio no varía. Es por ello que se debe tener en cuenta como diseña esta parte.

Por cada transacción, el banco cobra un impuesto sobre el monto de la transacción, que varía según la transacción sea nacional o internacional.
En el caso de las transacciones nacionales el impuesto de las mismas es del **8%** en todos los casos. Y en el caso de las internacionales es del **10%**.

- Realice el diagrama de clases en UML
- Desarrolle el sistema en JAVA. El mismo debe permitir:

  c) public void agregarTransaccion(Transaccion unaTransaccion) //Agrega una transacción con todos los datos al sistema
  d) public int cantidadTotalDeTransacciones() //Retorna el número total de transacciones del banco
  e) public int impuestoACobrarDe(Transaccion unaTransaccion) //Retorna el valor (en Pesos) del impuesto que debe cobrar. Por ejemplo: Supongamos que se trata de una transacción en dólares de 100USD. El impuesto a cobrar sería de: (100 * 27.09) * (0.1) = **270.90**
  f) public int impuestosACobrar() // Retorna la suma de todos los impuestos a cobrar en pesos.
  g) public int montoTotalDeTransacciones() // retorna la suma en pesos de todas transacciones realizadas.