Universidad Autónoma
del Estado de México

Licenciatura en Informática Administrativa

Unidad de aprendizaje: Programación orientada a objetos

Clave	Horas teoría	Horas práctica	Total de horas	Créditos
L30027	2	4	6	8

Unidad de competencia I

"Conceptos Programación orientada a objetos"

L.I.A. Cecilia Bibiana Ramírez Waldo

Septiembre 2019.

Tipo de Unidad de Aprendizaje	Carácter de la Unidad de Aprendizaje	Núcleo de formación	Modalidad
Curso Teórico-Práctico	Obligatoria	Sustantivo	Presencial

Índice

Guion explicativo

- La programación orientada a objetos es considerada como una metodología más de desarrollo en aplicaciones de cómputo necesarias en el entorno de trabajo actual, por lo que este programa de estudios, pretende dar las bases de los lenguajes orientados a objetos para que el alumno aprenda a realizar aplicaciones mediante la comprensión de este paradigma.

Objetivo de la unidad de aprendizaje

- El estudiante comprenderá la programación orientada a objetos y será capaz de elaborar aplicaciones de cómputo mediante este estilo de programación.

Competencias genéricas

- Capacidad de análisis.- Habilidad para la evaluación de datos y líneas de actuación, así como para tomar decisiones lógicas de una manera imparcial desde el punto de vista racional.
- Análisis y resolución de problemas.- Eficacia para identificar problemas, buscar datos pertinentes al respecto, reconocer la información relevante y encontrar posibles causas del mismo.
- Creatividad.- Capacidad para identificar alternativas y proponer soluciones.
- Conocimientos de informática relativos al ámbito de estudios.- Habilidad y eficiencia para desarrollar algoritmos de computadora y para identificar y manipular estructuras de datos. Habilidad en el manejo de equipo de cómputo

Estructura de la unidad de aprendizaje

1. Comprender los conceptos, entorno y elementos básicos de la Programación Orientada a Objetos.
2. Conocer la metodología de programación, las herramientas y las estructuras básicas necesarias para construir aplicaciones mediante la Programación Orientada a Objetos.
3. Desarrollo de aplicaciones que luzcan como página Web.
4. Desarrollo de programas que permitan el acceso a base de datos.

Secuencia didáctica

- Unidad 1. Conceptos Programación orientada a objetos
- Unidad 2. Aplicaciones básicas
- Unidad 3. Diseño de interfaces gráficas
- Unidad 4. Aplicaciones para web
- Unidad 5. Acceso a datos

Contenido

Unidad de competencia I:
Conceptos Programación orientada a objetos

1. Comprender conceptos básicos de POO.
2. Identificar las herramientas de desarrollo.
3. Identificar los elementos del lenguaje y su Entorno de Desarrollo Integrado (IDE).

1. Conceptos básicos de la POO

- La programación orientada a objetos (POO), se define como un paradigma que permite realizar una abstracción de la realidad, que se puede implementar en una aplicación de *software,* que permite el desarrollo de aplicaciones robustas (Flórez, 2012).

- La programación orientada a objetos (POO), esta constituida por objetos con ciertas características y métodos, los cuales pertenecen a una clase para poder satisfacer necesidades especificas que brinden funcionabilidad al programa (Osorio, 2007).

1.1. Paquete

- Se define como un contenedor de clases. Se utiliza para ordenar el código de acuerdo a los servicios que se implementarán en el programa (Flórez, 2012).

- Su característica principal es que el nombre de ese paquete contendrá el código.

Sintaxis:

```
package Mipaquete;
```

1.2. Importación de clases

- Los POO, se desarrollan mediante el uso de clases predefinidas, en Java estas clases se agrupan en paquetes (Duran, 2007).

Sintaxis:

import *java.util,*;*

- Para usar elementos dentro de un programa se puede escribir su nombre completo, o en su defecto, si se requiere de una clase profundamente anidado, es necesario usar puntos (.)

Sintaxis:

import *java.util;*

import *java.util.Scanner;*

1.3. Clase

- Una clase describe a un conjunto de objetos que comparten una estructura y un comportamiento en común (García, 2010).
- Para poder definir una clase en Java, se requiere de la palabra reservada ***class***, además de la especificación de un nombre para dicha clase.

Sintaxis:

public class Nombredelaclase {

1.4 Relación entre clases

Las relaciones entre clases, como indica Booch (1994), se deben a dos razones

- Primera, una relación de clases puede indicar algún tipo de compartición. (por ejemplo, margaritas y rosas son ambas tipos de flores).
- Segunda, una relación entre clases puede indicar algún tipo de conexión semántica; por ejemplo, las rosas rojas y amarillas son más parecidas entre si que las margaritas y rosas.

Existen tres tipos de relaciones entre clases son:

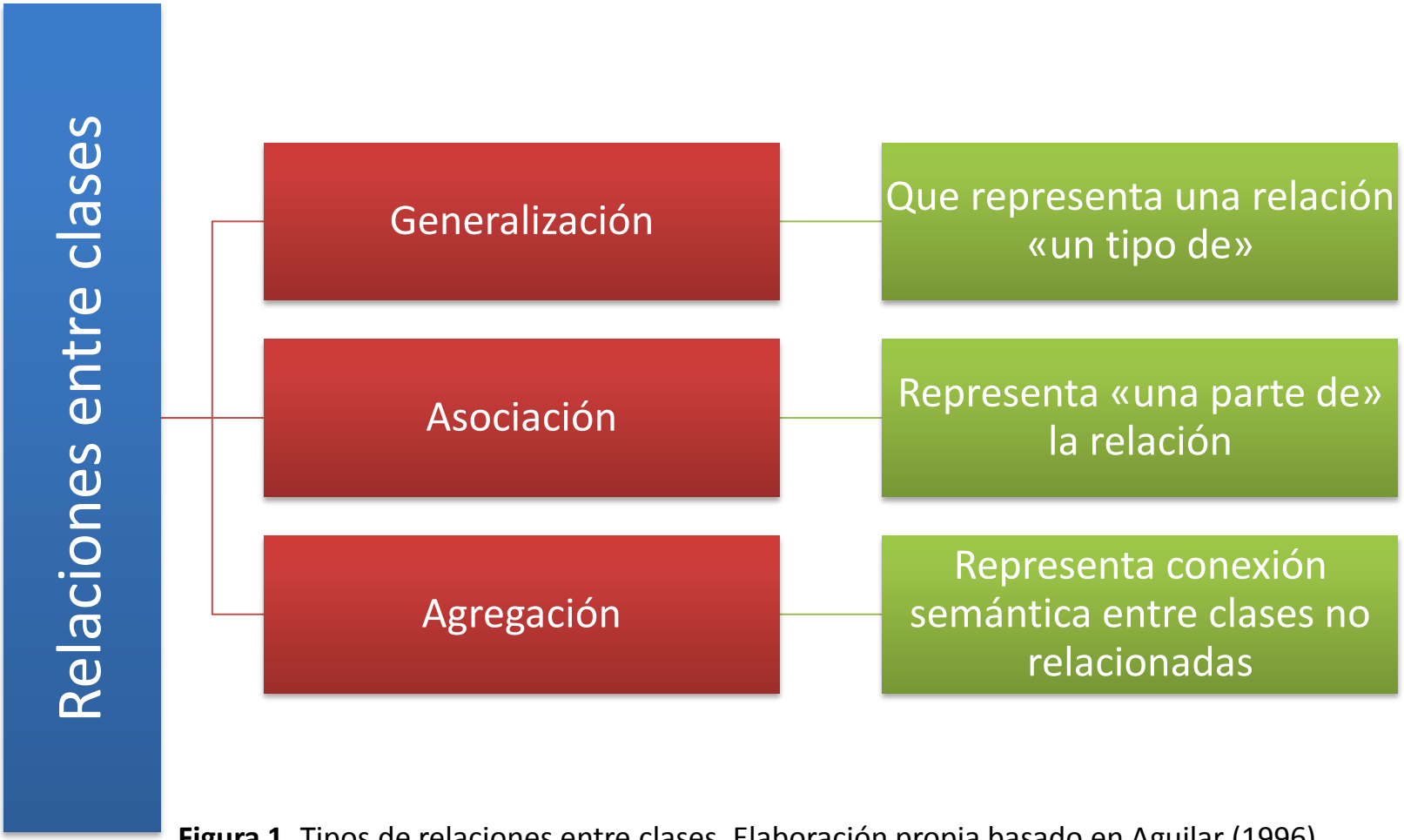

Figura 1. Tipos de relaciones entre clases. Elaboración propia basado en Aguilar (1996).

- Por ejemplo:
- La ***generalización,***

  Una rosa es **un tipo de** flor.

- La ***asociación,***

  Un pétalo no es una clase de flor; es **una parte de** una flor.

- La ***agregación,***

  Las rosas y velas son clases independientes, aunque ambas representan cosas que se pueden utilizar para decorar una mesa para cenar.

1.5. Objeto

- Según Osorio (2007), un objeto es un elemento real o abstracto que tiene un estado, un comportamiento y una identidad.
- Un objeto que pertenece a una clase, se dice que es una **instancia** (acción o particularización específica de determinado objeto) de la clase.

Clase: **Medios de transporte**

Objeto: **Helicóptero**

Figura 2. Representación de un objeto. Elaboración propia

1.6 Atributos

- Son las características del objeto (que tiene, de que consta), en Java se definen a través de tipos de datos o variables, (Hernández, 2001)

Atributos
+ Raza
+ Color
+ Talla
+ Peso

Objeto: Perro

Figura 3. Representación de atributos de un objeto. Elaboración propia basado en Hernández(2001).

- Los atributos describen el estado del objeto Un atributo consta de dos partes: un nombre de atributo y un tipo de dato.

Sintaxis:

Tipo de dato → ***int*** Dato1; ← Nombre del atributo

String Nombre;

- Los objetos simples pueden constar de tipos primitivos, tales como enteros, carácter, boolean, reales, o tipos simples definidos por el usuario.
- Los objetos complejos pueden constar de pilas, conjuntos, listas, array, entre otros.

Sintaxis:

boolean f=true;

int x[5] = {3,2,7,1,8} ;

1.7 Método

Es una acción que determina como debe de actuar un objeto cuando recibe un mensaje, también podría llamarse “función” (Hernández, 2001).

- Un método se invoca mediante una llamada, y cuando el método que se llamo completa su tarea, devuelve un resultado, o simplemente el control al método que lo llamo (Deitel, 2008).

Objeto: Perro

Atributos:
+ Raza
+ Color
+ Talla
+ Peso

Métodos:
Jugar();
Ladrar();
Dormir();
Comer ();

Figura 4. Representación de métodos de un objeto. Elaboración propia basado en Deitel, (2008).

1.8 Diagramas de clase UML

- Un diagrama UML de clase es una técnica diagramática para describir clases, objetos y las relaciones entre ellos.

- Un cuadro de diagrama UML de clase está dividido en tres partes:
  - nombre de clase hasta arriba,
  - *atributos* en la parte de en medio y
  - *operaciones* al final. (Dean, 2009)

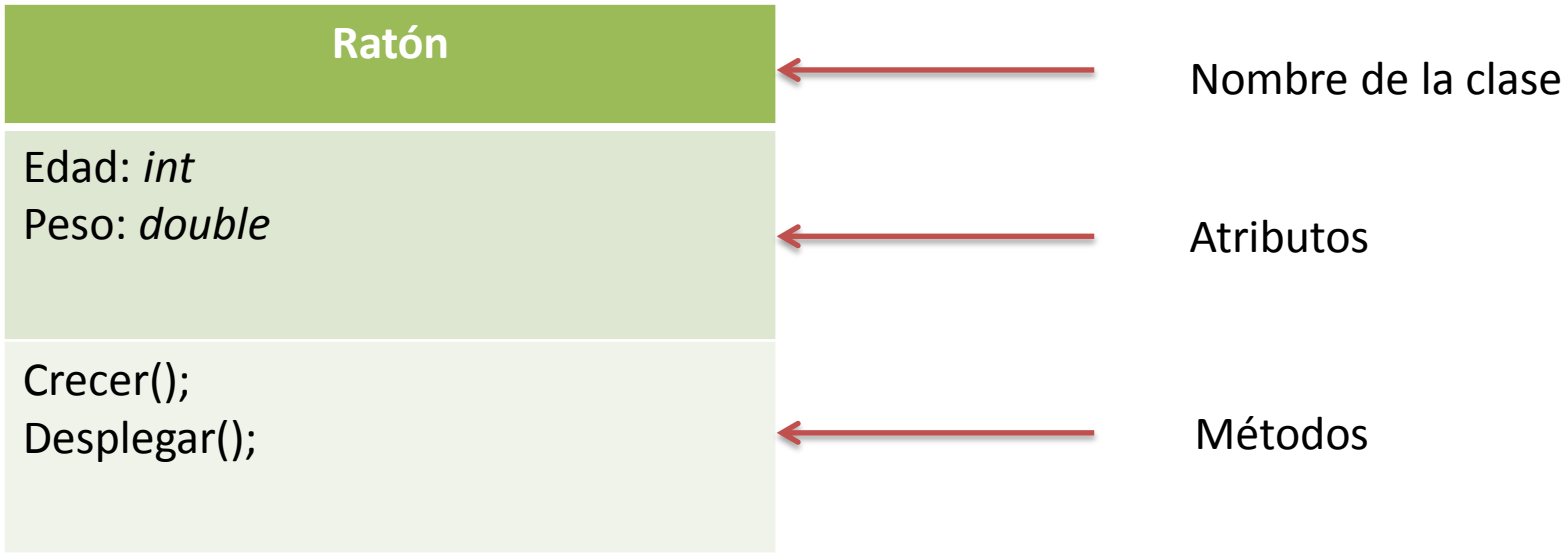

Figura 5. Partes de un diagrama de clases UML. Elaboración propia basado en Dean, (2009).

1.9 Especificadores de acceso

- Son aquellos que permiten controlar el acceso a los miembros de una clase, se clasifican en:

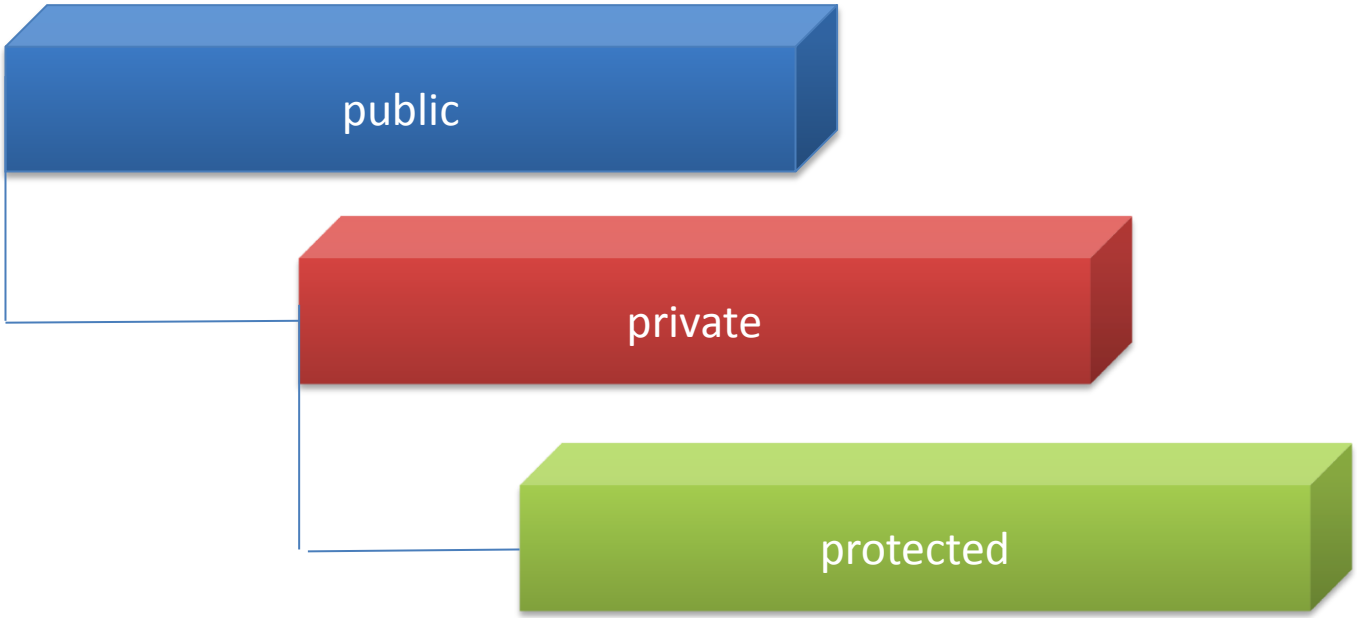

Figura 6. Clasificación de los especificadores de acceso. Elaboración propia basado en Dean, (2009).

- *public*, pueden ser accedidos en cualquier parte (dentro de los miembros de clase y también fuera de ellos).

- Se debe declarar un método como public cuando se quiera que sea un portal a través del cual el mundo pueda acceder a los datos del objeto. (Dean, 2009)

- *Private*, el miembro puede ser accedido sólo desde dentro de la clase del miembro y no desde "fuera del mundo" (esto es, por código que esté fuera de la clase en la cual reside el miembro).
- Las variables de instancia son declaradas casi siempre con el modificador de acceso private porque siempre se querrá que los datos del objeto estén ocultos. (Dean, 2009)

- *protected,* estos miembros de la clase (ya sean datos o métodos) son visibles desde dentro del la clase y desde cualquier otra clase heredada, es decir, clases hijas o subclases, como prefiera llamarlas (Morero, 2000).

1.10 Constructor

- Un constructor es una entidad semejante a un método que es llamado automáticamente cuando se instancia un objeto. (Dean, 2009)

```
public Carro(String m, int y, String c)
{
   this.marca = m;
   this.año = y;
   this.color = c;
}
```

- El constructor es una función como cualquier otra, salvo por un par de particularidades: se llama como la clase y no tiene tipo de retorno. (Rodríguez, 1998)

1.11 Herencia

- La herencia es el medio para compartir en forma automática los métodos y los datos entre las clases y subclases de los objetos. Los objetos heredan las propiedades y el comportamiento de todas las clases a las que pertenecen (Hernández, 2001).

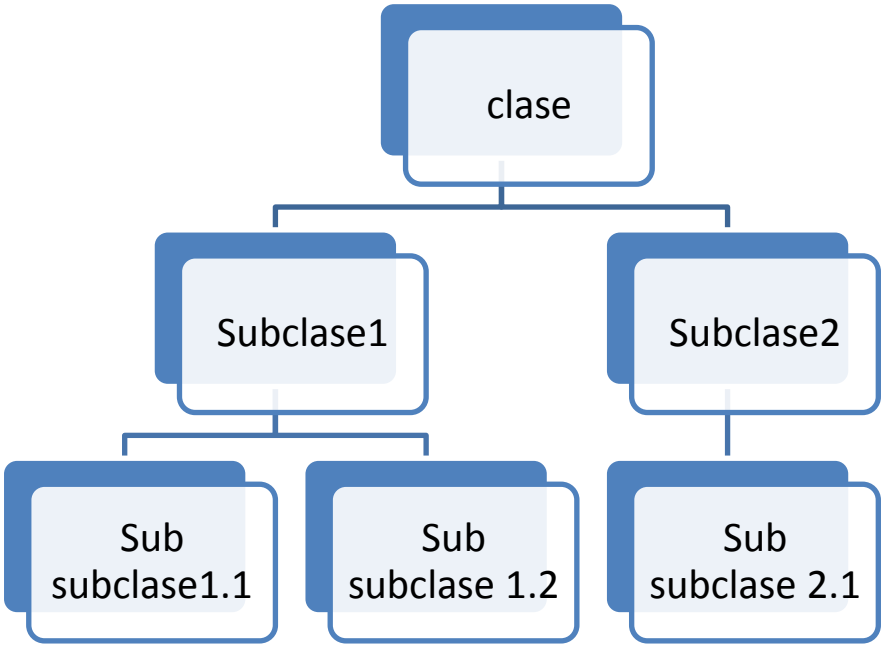

Figura 7. Herencia de clases.
Elaboración propia basado en Hernández, (2001).

Una clase hereda sus características (datos y funciones) de otra clase Esta característica proporciona claramente un soporte poderoso para reutilización y extensibilidad, dado que la definición de nuevos objetos se pueden basar en clases existentes (Aguilar, 1996).

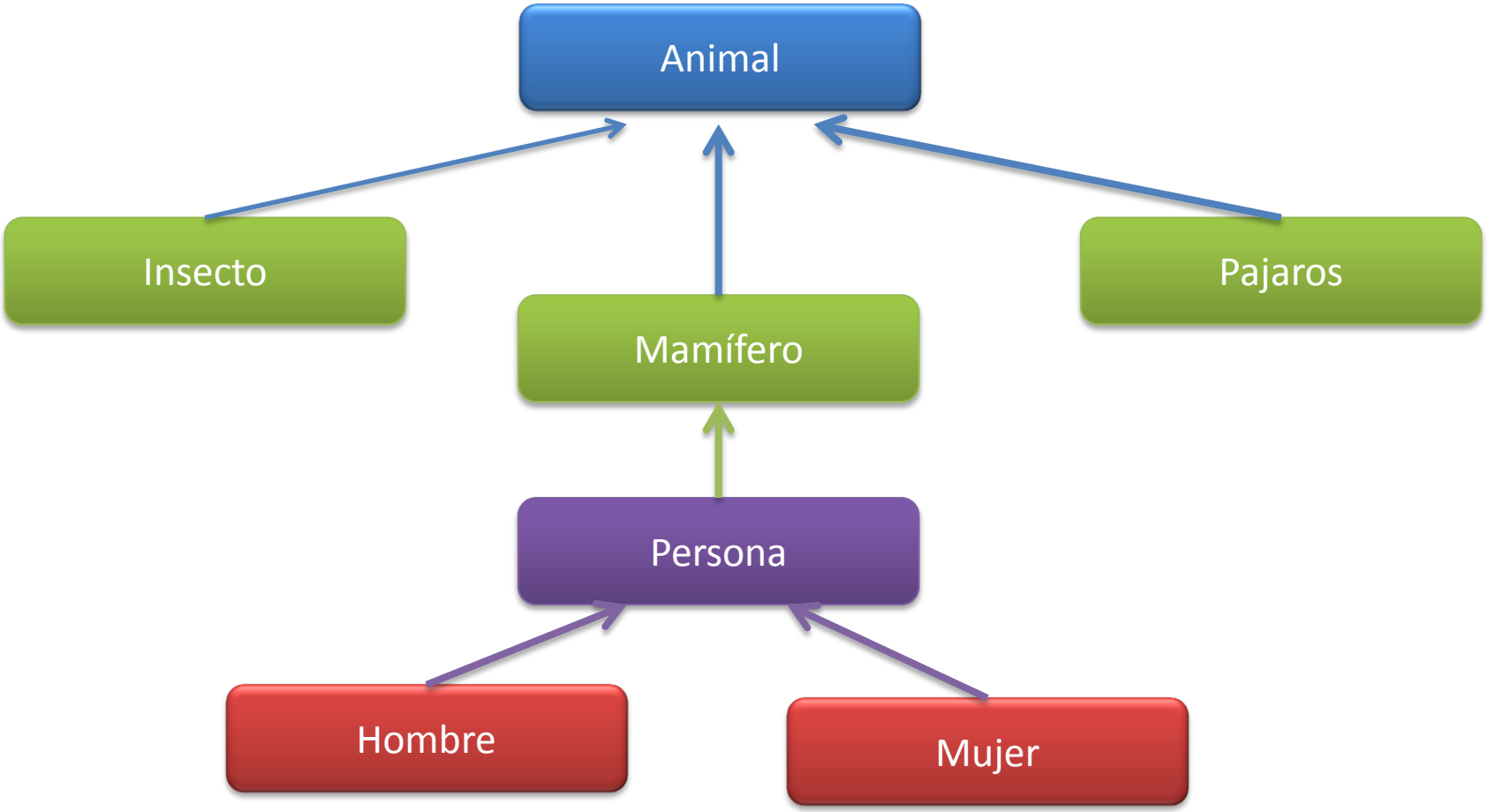

Figura 8. Ejemplo de jerarquía de clases. Elaboración propia basado en Aguilar, (1996).

2. Identificar herramientas de desarrollo

- La máquina virtual de Java se denomina al procesador o entorno virtual que se utiliza para interpretar los bytecodes de los binarios de Java, ya que como sabemos Java se hizo para correr en cualquier plataforma sin recompilar los binarios.

De esta manera este entorno virtual se puede obtener para nuestra arquitectura y sistema operativo sin modificaciones a nuestro programa original (Pérez, 2008).

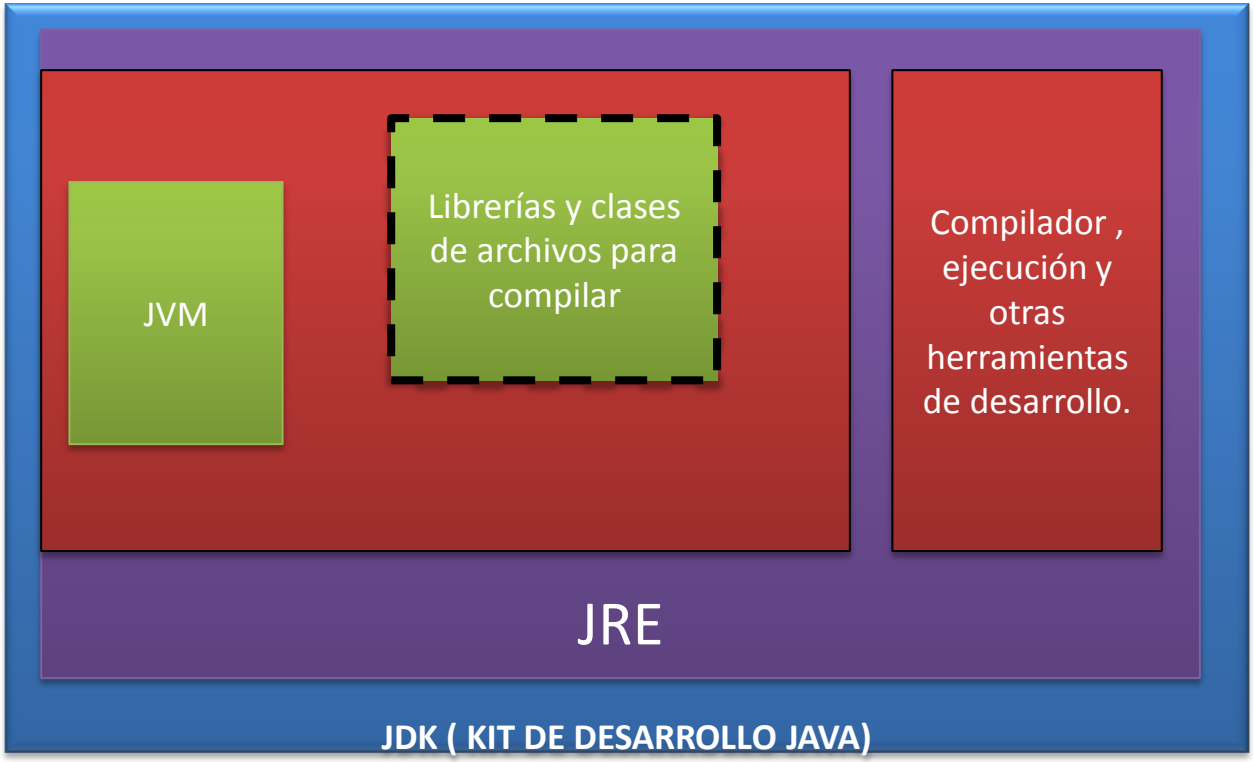

Figura 9. Máquina virtual de java. Basado en Pérez, (2008).

2.1 JDK

- El Kit de desarrollo conocido como JDK (Java Development Kit) provee de un compilador, un mecanismo para comprimir un proyecto en un solo archivo de tipo JAR (que es compatible con ZIP) y un entorno de ejecución para nuestros binarios. (Pérez, 2008)

- El JDK (Java Development Kit) o Equipo de Desarrollo de Java, en español, contiene las herramientas que permiten a los usuarios crear aplicaciones en Java.

- Simplemente se trata de un conjunto de programas y librerías que permiten desarrollar, compilar y ejecutar programas en Java.

- Los JDK incorporan una herramienta de Debugger lo que significa que se puede detener la ejecución de un programa en la línea que se desee para poder conocer el valor de las variables en ese momento.

2.2 JRE

- El JRE (Java Runtime Environment) Entorno en Tiempo de Ejecución de Java, en español, consta de la Máquina Virtual de Java, bibliotecas, y todos los demás componentes necesarios para ejecutar aplicaciones Java y applets.

- Viene incluido en el JDK pero también puede instalarse por separado y también es de libre distribución.
- Es un entorno de ejecución que es más pequeño en cuestiones sólo de espacio en disco (Pérez, 2008).

2.3 Comandos que inician el JDK, JRE

Comando	Descripción
java	Inicia el entorno de ejecución recibiendo como argumento el nombre del binario ejecutable en formato ByteCodes sin la extensión de archivo .class que identifica de manera visual un binario java. Este comando es parte de JDK y JRE
javac	Inicia el compilador Java recibiendo como argumento todos los archivos de código fuente cuya terminación es .java incluida dicha extensión. Este comando no es parte de JRE.
jar	Por medio de este comando iniciamos el empaquetador de clases y archivos de Java que nos permiten fabricar un único archivo contenedor de nuestras aplicaciones, multimedia y gráficos. Este comando es parte sólo de JDK.

Figura 10. Comandos de ejecución en java. Basado en Pérez, (2008).

3. Identificar los elementos del lenguaje y su Entorno de Desarrollo Integrado (IDE).

- El entorno de desarrollo integrado, será aquel *software* que nos permite trabajar con una máscara que va más allá de un editor de textos.

- Para ello aun se requiere de un JDK, y el IDE complementa con herramientas y facilita la programación.

Entornos de desarrollo integrado

- **1. NetBeans**

Inicialmente desarrollado por Sun y ahora en manos de Oracle, es uno de los IDE para desarrollo Java más completos.

NetBeans tiene una estructura modular fácilmente ampliable mediante complementos, existiendo configuraciones predefinidas para desarrollo Java SE, Java EE y también dirigidas a otros lenguajes de programación, como PHP o C++.

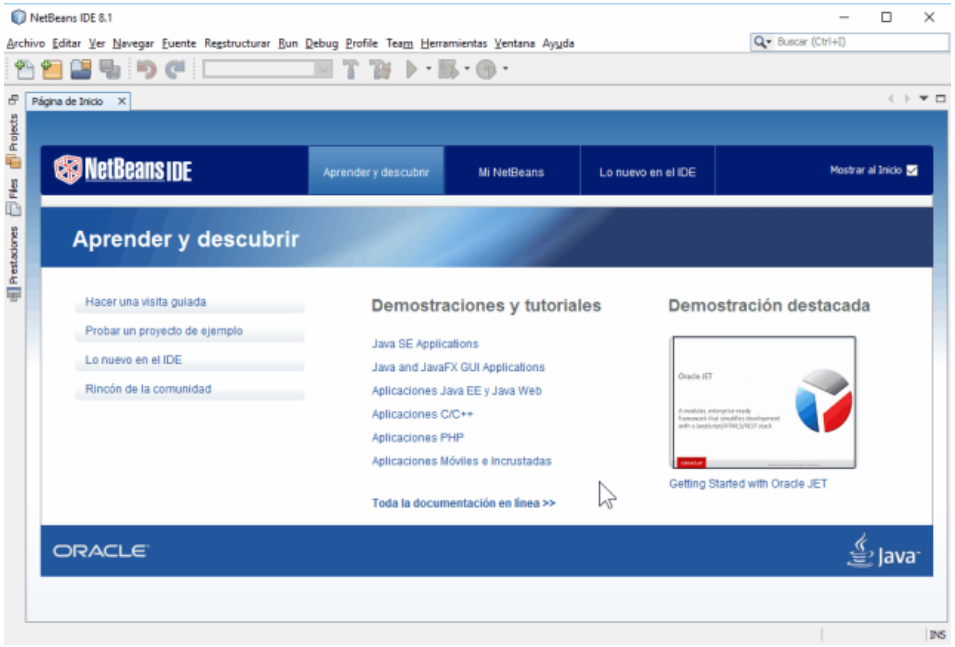

Figura 11. Entorno gráfico de NetBeans. Basado en Oracle (2018).

Entornos de desarrollo integrado

- **2. Eclipse**

Es un proyecto de código abierto y el IDE está disponible para múltiples sistemas operativos. Eclipse es un IDE no solo para Java, sino para muchos otros lenguajes y herramientas de desarrollo. Es considerado por muchos el IDE por excelencia, al incorporar un gran abanico de complementos que facilitan prácticamente todas las tareas relativas al desarrollo de software.

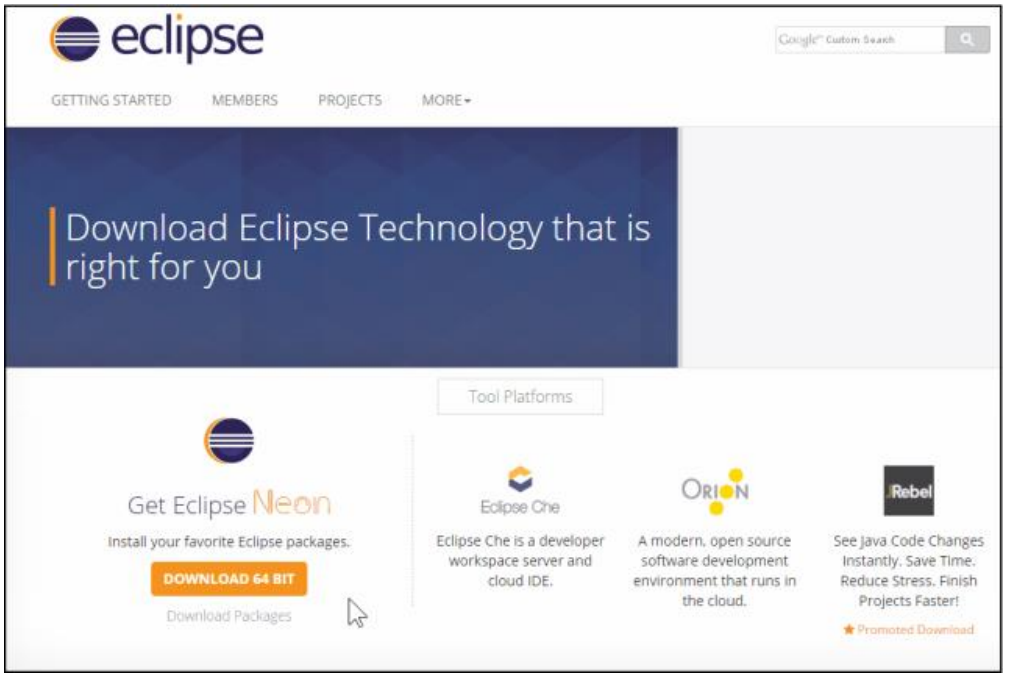

Figura 12. Entorno gráfico de Eclipse. Basado en Oracle (2018).

Entornos de desarrollo integrado

- **3. JCreator**

Es la herramienta de desarrollo para cada programador que le gusta hacer lo que mejor sabe hacer: la programación. Es más rápido, más eficiente y más confiable que otros IDE de Java. Por lo tanto, es la herramienta perfecta para programadores de todos los niveles, desde programadores de aprendizaje hasta especialistas en Java.

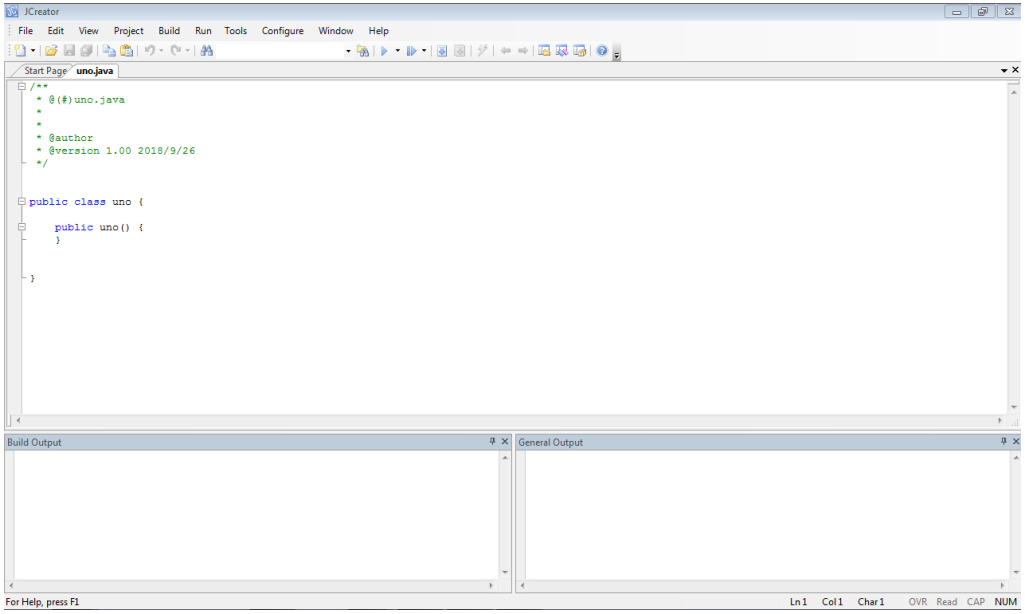

Figura 13. Entorno gráfico de JCreator. Basado en Oracle (2018).

Conclusiones

- En la programación orientada a objetos comprender y manejar los conceptos, e identificar las herramientas que nos permitirán el desarrollo de aplicaciones básicas en un lenguaje de POO.

Bibliografía

- Aguilar, L. J. (1996). *Programación orientada a objetos.* Madrid: McGraw-Hill.
- Dean, R. H. (2009*). Introducción a la programación con Java*. México: McGrawHill.
- Deitel, P. J. (2008). *Java cómo programar*. México: Pearson.
- Durán, F. (2007). *Programación orientada a objetos con Java.* España: Thomson.
- Flórez, H. (2012). *Programación orientada a objetos usando Java.* Bogotá: ECOE EDICIONES.
- Hernández, L. A. (2001). *Programación orientada a objetos en Java.* México: UNAM.
- Morero, F. (2000). *Introducción a la POO.* EIDOS.
- Rodríguez, A. G. (1998). *Introducción a la programación orientada a objetos.* Catalunya: UOC.
- Osorio, F. (2007). *Introducción a la programación en Java: Un enfoque práctico.* Colombia: Instituto Tecnológico Metropolitano.
- Pérez, G. G. (2008). *Aprendiendo Java y programación orientada a objetos*.

Por su atención, gracias.