Ejercicio 2.1. Definición de clases

Las clases son modelos del mundo real que capturan la estructura y comportamiento compartidos por una colección de objetos de un mismo tipo (Seidl *et al*., 2015). Una clase está conformada por sus atributos y métodos. Una clase se define en Java como:

```
class NombreClase {
        lista de atributos
        lista de constructores
        lista de métodos
}
```

Un objeto se considera la instancia de una clase. Para crear un objeto se debe invocar a su constructor, el cual coincide con el nombre de la clase y se debe utilizar la palabra reservada *new*.

```
Clase objeto = new Clase();
```

Los constructores, además de permitir la instancia de objetos, realizan la inicialización de los atributos del objeto. Esto se logra pasando los valores de los atributos como parámetros en la invocación del constructor:

```
Clase nombreClase {
        tipo atributo;
        nombreClase(int parámetro, ...) { // Constructor
            this.atributo = parámetro;
            …
        }
}
```

De otro lado, la palabra *this* se utiliza para referirse a los atributos de la clase y en particular, para diferenciar cuando los parámetros del constructor tienen el mismo nombre que los atributos.

El operador . (punto) permite acceder a los distintos atributos y métodos de una clase. El formato de la operación punto es:

```
objeto.atributo;
objeto.metódo();
```

Objetivos de aprendizaje

Al finalizar este ejercicio, el lector tendrá la capacidad para:

- Entender los conceptos: clase, objeto, constructor y la referencia *this*.
- Comprender el significado de los atributos de un objeto.
- Realizar la instanciación de objetos.

Enunciado: clase Persona

Se requiere un programa que modele el concepto de una persona. Una persona posee nombre, apellido, número de documento de identidad y año de nacimiento. La clase debe tener un constructor que inicialice los valores de sus respectivos atributos.

La clase debe incluir los siguientes métodos:

- Definir un método que imprima en pantalla los valores de los atributos del objeto.
- En un método *main* se deben crear dos personas y mostrar los valores de sus atributos en pantalla.

Instrucciones Java del ejercicio

Tabla 2.1. Instrucciones Java del ejercicio 2.1.

Instrucción	Descripción	Formato
this	Palabra clave que se puede usar dentro de un método o constructor una clase. Funciona como una referencia al objeto actual.	*this*.atributo = parámetro;
void	Palabra clave que especifica que un método no tiene un valor de retorno.	*void* nombreMétodo() { }

Solución

Clase: Persona

```
/**
 * Esta clase define objetos de tipo Persona con un nombre, apellidos,
 * número de documento de identidad y año de nacimiento.
 * @version 1.2/2020
 */
public class Persona {

    String nombre; // Atributo que identifica el nombre de una persona
    String apellidos; // Atributo que identifica los apellidos de una persona
```

```
/* Atributo que identifica el número de documento de identidad de
    una persona */
String númeroDocumentoIdentidad;
int añoNacimiento; /* Atributo que identifica el año de nacimiento
    de una persona */

/**
* Constructor de la clase Persona
* @param nombre Parámetro que define el nombre de la persona
* @param apellidos Parámetro que define los apellidos de la persona
* @param númeroDocumentoIdentidad Parámetro que define el
* número del documento de identidad de la persona
* @param añoNacimiento Parámetro que define el año de nacimiento
* de la persona
*/
Persona(String nombre, String apellidos, String númeroDocumento
    Identidad, int añoNacimiento) {
    this.nombre = nombre;
    this.apellidos = apellidos;
    this.númeroDocumentoIdentidad = númeroDocumentoIdentidad;
    this.añoNacimiento = añoNacimiento;
}

/**
* Método que imprime en pantalla los datos de una persona
*/

void imprimir() {
    System.out.println("Nombre = " + nombre);
    System.out.println("Apellidos = " + apellidos);
    System.out.println("Número de documento de identidad = " +
        númeroDocumentoIdentidad);
    System.out.println("Año de nacimiento = " + añoNacimiento);
    System.out.println();
}

/**
* Método main que crea dos personas e imprime sus datos en pantalla
*/
```

```
        public static void main(String args[]) {
            Persona p1 = new Persona("Pedro","Pérez","1053121010",1998);
            Persona p2 = new Persona("Luis","León","1053223344",2001);
            p1.imprimir();
            p2.imprimir();
        }
    }
```

Diagrama de clases

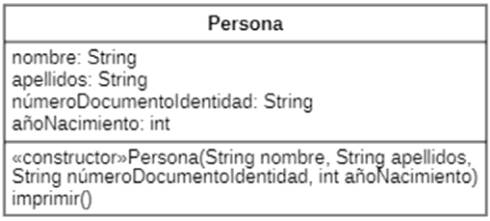

Figura 2.1. Diagrama de clases del ejercicio 2.1.

Explicación del diagrama de clases

Se ha definido una sola clase denominada Persona. El nombre de la clase se ubica en el primer compartimiento de la clase. En el segundo compartimiento se han definido los cuatro atributos junto con su tipo (nombre, apellidos y añoNacimiento de tipo *int* y númeroDocumentoIdentidad de tipo *String*). En el tercer compartimiento se han definido dos métodos, comenzando con el constructor, el cual tiene la etiqueta <<constructor>> como un estereotipo UML (brinda información adicional y personalizada) para su correcta identificación. El otro método es imprimir, el cual no contiene parámetros ni valor de retorno.

Diagrama de objetos

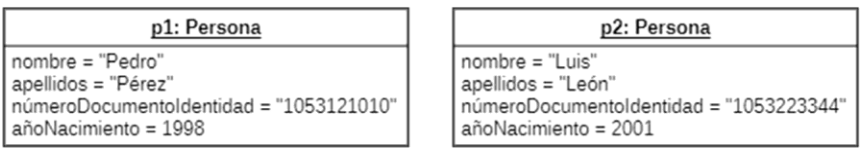

Figura 2.2. Diagrama de objetos del ejercicio 2.1.

Ejecución del programa

```
Nombre = Pedro
Apellidos = Pérez
Número de documento de identidad = 1053121010
Año de nacimiento = 1998

Nombre = Luis
Apellidos = León
Número de documento de identidad = 1053223344
Año de nacimiento = 2001
```

Figura 2.3. Ejecución del programa del ejercicio 2.1.

Ejercicios propuestos

- **Agregar dos nuevos atributos a la clase Persona. Un atributo que represente el país de nacimiento de la persona (de tipo *String*) y otro que identifique el género de la persona, el cual debe represen-tarse como un *char* con valores 'H' o 'M'.**
- **Modificar el constructor de la clase Persona para que inicialice es-tos dos nuevos atributos.**
- **Modificar el método imprimir de la clase Persona para que mues- tre en pantalla los valores de los nuevos atributos.**

Ejercicio 2.2. Definición de atributos de una clase con tipos primitivos de datos

También existen los datos enumerados que representan un grupo de constantes con valores predefinidos. Se debe utilizar la palabra clave *enum* y separar las constantes con una coma. Los elementos enumerados deben estar en letras mayúsculas. El formato de los datos enumerados es:

enum variable {ELEMENTO1, ELEMENTO2, ELEMENTO3}

En este caso, la variable es un valor enumerado que puede asumir los valores ELEMENTO1, ELEMENTO2 o ELEMENTO3.

Objetivos de aprendizaje

Al finalizar este ejercicio el lector tendrá la capacidad para:

- Definir atributos de una clase con un tipo de primitivo de dato.

- Definir los valores iniciales de los atributos de una clase.
- Definir atributos de tipo enumerado.

Enunciado: clase Planeta

Se requiere un programa que modele el concepto de un planeta del sistema solar. Un planeta tiene los siguientes atributos:

- Un nombre de tipo *Srping* con valor inicial de *null*.
- Cantidad de satélites de tipo *int* con valor inicial de cero.
- Masa en kilogramos de tipo *double* con valor inicial de cero.
- Volumen en kilómetros cúbicos de tipo *double* con valor inicial de cero.
- Diámetro en kilómetros de tipo *int* con valor inicial de cero.
- Distancia media al Sol en millones de kilómetros, de tipo *int* con valor inicial de cero.
- Tipo de planeta de acuerdo con su tamaño, de tipo enumerado con los siguientes valores posibles: GASEOSO, TERRESTRE y ENANO.
- Observable a simple vista, de tipo booleano con valor inicial *false*.

La clase debe incluir los siguientes métodos:

- La clase debe tener un constructor que inicialice los valores de sus respectivos atributos.
- Definir un método que imprima en pantalla los valores de los atributos de un planeta.
- Calcular la densidad un planeta, como el cociente entre su masa y su volumen.
- Determinar si un planeta del sistema solar se considera exterior. Un planeta exterior está situado más allá del cinturón de asteroides. El cinturón de asteroides se encuentra entre 2.1 y 3.4 UA. Una unidad astronómica (UA) es la distancia entre la Tierra y el Sol= 149 597 870 Km.
- En un método *main* se deben crear dos planetas y mostrar los valores de sus atributos en pantalla. Además, se debe imprimir la densidad de cada planeta y si el planeta es un planeta exterior del sistema solar.

Instrucciones Java del ejercicio

Instrucción	Descripción	Formato
return	Finaliza la ejecución de un método y puede utilizarse para devolver el valor de un método.	*return*; *return* variable;

Tabla 2.3. Instrucciones Java del ejercicio 2.2.

Diagrama de clases

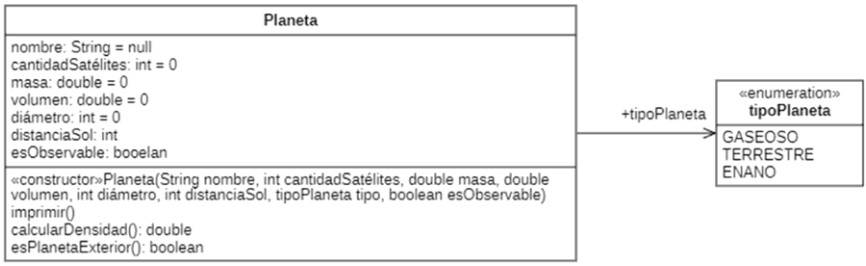

Figura 2.4. Diagrama de clases del ejercicio 2.4.

Explicación del diagrama de clases

Se ha definido una clase denominada Planeta con sus respectivos atributos que representan el nombre del planeta, cantidad de satélites que tiene, su masa, volumen, diámetro, distancia al Sol y si es observable o no. Para cada atributo, además de su tipo, se ha agregado su correspondiente valor inicial o por defecto. También se han definido sus métodos respectivos: un constructor que inicializa los valores de sus atributos, un método para imprimir los valores de sus atributos en pantalla, otro para calcular la densidad (que devuelve un valor de tipo *double*) y un último método para determinar si es un planeta exterior (que devuelve un valor booleano).

El atributo para identificar el tipo de planeta se expresa como una asociación entre la clase Planeta y la clase TipoPlaneta cuyo nombre es el nombre del atributo. En UML, las variables *enum* se identifican con el estereotipo <<enumeration>>, que representan un conjunto de valores constantes identificados como atributos en su segundo compartimiento.

Diagrama de objetos

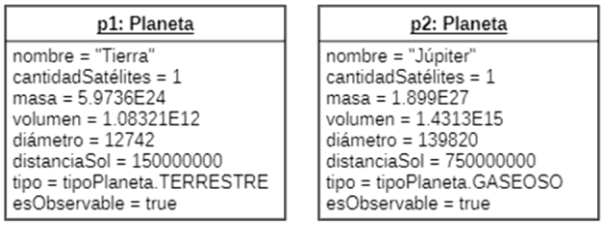

Figura 2.5. Diagrama de objetos del ejercicio 2.2.

Ejecución del programa

```
Nombre del planeta = Tierra
Cantidad de satélites = 1
Masa del planeta = 5.9736E24
Volumen del planeta = 1.08321E12
Diámetro del planeta = 12742
Distancia al sol = 150000000
Tipo de planeta = TERRESTRE
Es observable = true
Densidad del planeta = 5.514720137369484E12
Es planeta exterior = false

Nombre del planeta = Júpiter
Cantidad de satélites = 79
Masa del planeta = 1.899E27
Volumen del planeta = 1.4313E15
Diámetro del planeta = 139820
Distancia al sol = 750000000
Tipo de planeta = GASEOSO
Es observable = true
Densidad del planeta = 1.3267658771745964E12
Es planeta exterior = true
```

Figura 2.6. Ejecución del programa del ejercicio 2.2.

Ejercicios propuestos

- Agregar dos atributos a la clase Planeta. El primero debe representar el periodo orbital del planeta (en años). El segundo atributo representa el periodo de rotación (en días).
- Modificar el constructor de la clase para que inicialice los valores de estos dos nuevos atributos.
- Modificar el método imprimir para que muestre en pantalla los valores de los nuevos atributos.

Ejercicio 2.3. Estado de un objeto

Se denomina estado de un objeto al conjunto de pares *{atributo = valor de un objeto}*. El estado de un objeto puede cambiar a lo largo de su vida a medida que se vayan ejecutando sus métodos (Arroyo-Díaz, 2019a).

Las clases tienen dos tipos de métodos: *get* y *set*.

- Los métodos *get* permiten obtener el valor de un atributo de un objeto. Los métodos *get* se definen con el siguiente formato:

getNombreAtributo

- Los métodos *set* permiten asignar o cambiar el valor de un atributo de un objeto y, por lo tanto, cambian su estado. Los métodos *set* se definen con el siguiente formato:

setNombreAtributo(tipo parámetro)

Los métodos *get* y *set* tienen una estructura que se repite para cada uno de los atributos. Por ello, los entornos de desarrollo actuales permiten generar automáticamente dichos métodos.