Ejercicios

1.Proyecto 1: Inicialización a la creación de clases y uso de métodos en éstas.

2. Proyecto 2: Creación de instancias de las clases (objetos) en método Main mediante el uso de constructores. Diferencias entre imprimir valores o referencias. Uso de la depuración para comprobar la modificación de atributos.

3. Proyecto 3: Uso de constructores sobrecargados y constructor por defecto. Pruebas con la visibilidad de los atributos. Construcción y uso de los getters y setters.

4. Proyecto 4: Crear un programa que pida al usuario una hora y un número (n) y muestre la hora introducida y los n segundos siguientes. Para ello creamos la clase Hora.

5. Proyecto 5: Implementar la clase ContadorObjetos con un atributo estático que nos permita llevar la cuenta del número de objetos creados de esta clase.

6. Proyecto 6: Crear un enumerado y añadirlo como atributo a otra clase.

7. Proyecto 7: Creación de una diagrama de clases. Definición de paquetes y clases necesarias para gestionar una empresa ferroviaria.

8. Proyecto 8: Convertir un proyecto anterior a una clase y convertir sus métodos y atributos en estáticos. Generar un exportable y usarlo en un proyecto nuevo.

9. Proyecto 9: Herencia de clases. Métodos sobrescritos (@Override).

10. Proyecto 10: Herencia con sobrescritura de métodos y polimorfismo.

11. Proyecto 12: Uso de clases abstractas y de métodos toString() y equals() de la clase Object sobrescritos en una nueva clase.

12. Proyecto 13: Uso de clases finales, métodos estáticos, constantes y excepciones para la creación de una clase de Utilidades Matemáticas.

13. Proyecto 14: Uso de composición y herencia.

14. Proyecto 15: Diferencias entre el paso por valor (tipos primitivos) y por referencia (tipo objeto) de los argumentos de los métodos.

15. Proyecto 16: Uso de interfaces e implementación de ellas.

16. Proyecto 17: Puesta a prueba de todos los conocimientos de esta unidad. Uso de casting para comparar objetos de clases distintas.

Notas de clase

Regla	Descripción	Ejemplo
Atributos	Datos de la clase, sustantivos. Tienen que llevar delante la visibilidad (que suele ser privada por seguridad) y el tipo.	private int edad; protected String nombre;
Métodos	Son funciones, verbos. También deben llevar delante la visibilidad (suele ser publica para que se puedan usar desde otras clases (main) y el tipo.	public void anotarGol();
Clase	Plantilla para la creación de objetos. Los atributos y los métodos son sus principales miembros.	class Persona{...}
Objeto	Es una instancia de una clase. Hace uso de los constructores disponibles en la clase para su creación.	Coche coche = new Coche (“Seat Ibiza”);
Constructor	Es un método especial que se usa para inicializar un objeto o para crearlo. Puede ser: Por defecto, sin parámetros o con parámetros.	Persona(nombre, apellidos, edad...) Persona()→ Sin parámetros.

this	Palabra reservada con la que podemos acceder a atributos dentro de un método aunque hayan sido ocultados por vv locales con el mismo nombre. Si se usa con paréntesis al final llama al constructor.	this(nombre, apellidos...)
		this();
		this.edad = edad;

Visibilidad	Determina que parte del programa puede acceder a un atributo, método o constructor dentro de una clase. Hay varios tipos:	Public (+), private (-), protected (#) y package (sin nada). protected String dni;

Regla	Descripción	Ejemplo
Visibilidad	Sin modificador(solo accesible desde el mismo paquete), public, private y protected(accesible por subclases)	
Getters y setters	Métodos publicos que se van a referir a los atributos privados y se usan para acceder a ellos o modificarlos.	public void setEdad(){…} public int getEdad(int edad){...}
Estático	Atributos y métodos que pertenecen a la clase (y no a sus instancias) y por tanto, son compartidos por todos los objetos de la clase.	static void hoyEs (int dia){...}
Enumerados	Variables limitadas a una serie de valores	enum Sexo {MASCULINO, FEMENINO, OTRO}
Herencia	Subclases que obtienen los miembros de una superclase. No hay herencia múltiple pero si transitiva. Se usa la palabra extends.	class Coche extends Transporte{...}
super	Palabra clave que funciona igual que this pero para llamar a la superclase.	super(); super.golesAnotados;
Override	Permite a una subclase redefinir el comportamiento de un método heredado. Fundamental para el polimorfismo	@Override public String toString(){...}
Polimorfismo	Los objetos de la subclase son también objetos de la superclase pero puede implementar de forma distinta algunos métodos.	Deportista d = new Futbolista(...);
Object	Es la superclase de todas las clases en Java y todas heredan de ella. Proporciona funcionalidades esenciales que van a ser comunes a todas la clases.	toString(), equals(), getClass()

Regla	Descripción	Ejemplo
Métodos Abstractos	Métodos vacíos pensados para ser sobrescritos. No tienen implementación y terminan en ;. Deben ser implementados por las subclases que los hereden.	`public abstract void hacerSonido();`
Clases Abstractas	Clases que contienen al menos un método abstracto (tb pueden contener métodos concretos) y no pueden ser instanciadas directamente.	`abstract class Animal {...}`
Atributos finales	Constantes (su valor no puede cambiar tras incializarse). Se suele poner en mayúscula.	`final double PI = 3.141592`
Métodos finales	No pueden ser sobrescritos por subclase ni se puede modificar su implementación.	`public final boolean validarUsuario(String contraseña) {...}`
Clases finales	No pueden ser heredadas y sus métodos tienen que ser finales.	`final class ClaseFinal`

Interfaces	Son plantillas que definen un conjunto de métodos abstractos que una clase debe implementar. Permiten herencia múltiple. Se usa la palabra implements.	`public interface Volador{…}`
		
		`public class Pajaro implements Volador {...}`

Argumentos por valor	Se pasa una copia del original y los cambios NO afectarán al original. Afecta a los tipos primitivos	-
Referencia	Se pasa la referencia del original y los cambios SI afectarán al original. Afecta a los tipo Objeto.	-