Clase prueba this

public class PruebaThis {

    public static void main(String[] args) {

        // Crea un objeto TiempoSimple con hora, minuto y segundo

        TiempoSimple tiempo = new TiempoSimple(15, 30, 19);

        // Imprime el resultado del método crearString() del objeto tiempo

        System.err.println(tiempo.crearString());

    }

}

// La clase TiempoSimple declara la referencia 'this'

class TiempoSimple {

    private int hora; // Atributo privado hora

    private int minuto; // Atributo privado minuto

    private int segundo; // Atributo privado segundo

    // Constructor que inicializa los atributos

    public TiempoSimple(int hora, int minuto, int segundo) {

        this.hora = hora; // Asigna el valor de hora al miembro de la clase

        this.minuto = minuto; // Asigna el valor de minuto al miembro de la clase

        this.segundo = segundo; // Asigna el valor de segundo al miembro de la clase

    }

    // Método que devuelve una cadena con el formato de tiempo

    public String crearString() {

        return String.format("%02d:%02d:%02d", this.hora, this.minuto, this.segundo);

    }

    // Método que devuelve una cadena en un formato universal

    public String aStringUniversal() {

        return String.format("%04d-%02d-%02d", this.hora, this.minuto, this.segundo);

    }

}

Clase tiempo 1

public class Tiempo1 {

    private int hora; // Atributo privado hora

    private int minuto; // Atributo privado minuto

    private int segundo; // Atributo privado segundo

    // Método para establecer el tiempo

    public void establecerTiempo(int hora, int minuto, int segundo) {

        // Validación de los valores de hora, minuto y segundo

        if (hora < 0 || hora > 24 || minuto < 0 || minuto > 59 || segundo < 0 || segundo > 59) {

            throw new IllegalArgumentException("Hora, minuto y segundo estaban fuera de rango");

        }

        this.hora = hora; // Asigna el valor de hora al miembro de la clase

        this.minuto = minuto; // Asigna el valor de minuto al miembro de la clase

        this.segundo = segundo; // Asigna el valor de segundo al miembro de la clase

    }

    // Método que devuelve una representación en formato de 24 horas

    public String aStringUniversal() {

        return String.format("%02d:%02d:%02d", this.hora, this.minuto, this.segundo);

    }

    // Método que devuelve una representación en formato de 12 horas

    public String toString() {

        return String.format("%d:%02d:%02d", this.hora, this.minuto, this.segundo);

    }

}

Clase pruebatiempo 1

public class PruebaTiempo1 {

    public static void main(String[] args) {

        // Crea un objeto Tiempo

        Tiempo tiempo = new Tiempo();

        // Muestra el estado del objeto tiempo

        mostrarTempo("Después de crear el objeto tiempo:", tiempo);

        System.out.println();

        try {

            // Intenta establecer un tiempo inválido

            tiempo.establecerTiempo(99, 99, 99);

        } catch (IllegalArgumentException e) {

            // Imprime el mensaje de excepción si ocurre

            System.out.println("Excepción: " + e.getMessage());

        }

    }

    // Método estático para mostrar el tiempo

    private static void mostrarTempo(String mensaje, Tiempo tiempo) {

        System.out.println(mensaje + " " + tiempo.toString());

    }

}

Clase pruebaaccesomiembros

public class PruebaAccesoMiembros {

    public static void main(String[] args) {

        // Crea un objeto Tiempo

        Tiempo tiempo = new Tiempo();

        // Intento de acceder a los atributos privados, lo que generará errores de compilación

        tiempo.hora = 7; // Error: 'hora' tiene modificador de acceso privado

        tiempo.minuto = 15; // Error: 'minuto' tiene modificador de acceso privado

        tiempo.segundo = 30; // Error: 'segundo' tiene modificador de acceso privado

    }

}

**Tiempo 2**

// Declaración de la clase Tiempo2 con constructores sobrecargados.

public class Tiempo2 {

    private int hora;    // 0 - 23

    private int minuto;  // 0 - 59

    private int segundo; // 0 - 59

    // Constructor de Tiempo2 sin argumentos:

    // Inicializa cada variable de instancia a cero

    public Tiempo2() {

        this(0, 0, 0); // Invoca al constructor de Tiempo2 con tres argumentos

    }

    // Constructor de Tiempo2: se suministra hora, minuto y segundo con valor predeterminado de 0

    public Tiempo2(int hora) {

        this(hora, 0, 0); // Invoca al constructor con tres argumentos

    }

    // Constructor de Tiempo2: se suministra hora y minuto, segundo con valor predeterminado de 0

    public Tiempo2(int hora, int minuto) {

        this(hora, minuto, 0); // Invoca al constructor con tres argumentos

    }

    // Constructor de Tiempo2: se suministra hora, minuto y segundo

    public Tiempo2(int hora, int minuto, int segundo) {

        // Valida que la hora esté en el rango correcto (0-23)

        if (hora < 0 || hora >= 24)

            throw new IllegalArgumentException("hora debe estar entre 0 y 23");

        // Valida que el minuto esté en el rango correcto (0-59)

        if (minuto < 0 || minuto >= 60)

            throw new IllegalArgumentException("minuto debe estar entre 0 y 59");

        // Valida que el segundo esté en el rango correcto (0-59)

        if (segundo < 0 || segundo >= 60)

            throw new IllegalArgumentException("segundo debe estar entre 0 y 59");

        // Si todas las validaciones son correctas, asigna los valores a los atributos

        this.hora = hora;

        this.minuto = minuto;

        this.segundo = segundo;

    }

}

// Constructor de Tiempo2: se suministra otro objeto Tiempo2

public Tiempo2(Tiempo2 tiempo) {

    // Invoca al constructor con tres argumentos de la misma clase

    // Pasa los valores de hora, minuto y segundo del objeto recibido como parámetro

    this(tiempo.obtenerHora(), tiempo.obtenerMinuto(), tiempo.obtenerSegundo());

}

// Métodos Establecer (Setters)

// Establece un nuevo valor de tiempo usando la hora universal; valida los datos

public void establecerTiempo(int hora, int minuto, int segundo) {

    // Valida que la hora esté en el rango correcto (0-23)

    if (hora < 0 || hora >= 24)

        throw new IllegalArgumentException("hora debe estar entre 0 y 23");

    // Valida que el minuto esté en el rango correcto (0-59)

    if (minuto < 0 || minuto >= 60)

        throw new IllegalArgumentException("minuto debe estar entre 0 y 59");

    // Valida que el segundo esté en el rango correcto (0-59)

    if (segundo < 0 || segundo >= 60)

        throw new IllegalArgumentException("segundo debe estar entre 0 y 59");

    // Si todas las validaciones son correctas, asigna los valores a los atributos

    this.hora = hora;

    this.minuto = minuto;

    this.segundo = segundo;

}

// Valida y establece la hora

public void establecerHora(int hora) {

    // Valida que la hora esté en el rango correcto (0-23)

    if (hora < 0 || hora >= 24)

        throw new IllegalArgumentException("hora debe estar entre 0 y 23");

    // Asigna el valor validado al atributo hora

    this.hora = hora;

}

// Valida y establece el minuto

public void establecerMinuto(int minuto) {

    // NOTA: Hay un error en esta condición, debería ser "||" en lugar de "&&"

    // La condición actual es incorrecta y siempre será falsa

    if (minuto < 0 && minuto >= 60) // ERROR - debería ser: if (minuto < 0 || minuto >= 60)

        throw new IllegalArgumentException("minuto debe estar entre 0 y 59");

    // Asigna el valor al atributo minuto (sin validación adecuada debido al error)

    this.minuto = minuto;

}

// Valida y establece el segundo

public void establecerSegundo(int segundo) {

    // NOTA: Hay un error en esta condición, está invertida

    // La condición actual verifica si el segundo es válido, pero luego lanza excepción

    if (segundo >= 0 && segundo < 60) // ERROR - debería ser: if (segundo < 0 || segundo >= 60)

        throw new IllegalArgumentException("segundo debe estar entre 0 y 59");

    // Faltaría la asignación: this.segundo = segundo;

}

**Prueba tiempo 2**

// Uso de constructores sobrecargados para inicializar objetos Tiempo2

public class PruebaTiempo2 {

    public static void main(String[] args) {

        // Crea un objeto Tiempo2 usando el constructor sin parámetros (00:00:00)

        Tiempo2 t1 = new Tiempo2(); // 00:00:00

        // Crea un objeto Tiempo2 usando un constructor con solo la hora (02:00:00)

        Tiempo2 t2 = new Tiempo2(2); // 02:00:00

        // Crea un objeto Tiempo2 usando un constructor con hora y minuto (21:34:00)

        Tiempo2 t3 = new Tiempo2(21, 34); // 21:34:00

        // Crea un objeto Tiempo2 usando un constructor con hora, minuto y segundo (12:25:42)

        Tiempo2 t4 = new Tiempo2(12, 25, 42); // 12:25:42

        // Crea un objeto Tiempo2 usando el constructor de copia (copia de t4: 12:25:42)

        Tiempo2 t5 = new Tiempo2(t4); // 12:25:42

        // Imprime encabezado para mostrar los tiempos construidos

        System.out.println("Se construyó con:");

        // Llama al método mostrarTiempo para cada objeto, con una descripción diferente

        mostrarTiempo("t1: todos los argumentos predeterminados", t1);

        mostrarTiempo("t2: se especificó hora; minuto y segundo predeterminados", t2);

        mostrarTiempo("t3: se especificaron hora y minuto; segundo predeterminado", t3);

        mostrarTiempo("t4: se especificaron hora, minuto y segundo", t4);

        mostrarTiempo("t5: se especificó el objeto Tiempo2 llamado t4", t5);

        // Intento de inicializar t6 con valores inválidos usando un bloque try-catch

        try {

            // Intenta crear un objeto con valores inválidos (27 horas, 74 minutos, 99 segundos)

            Tiempo2 t6 = new Tiempo2(27, 74, 99); // valores inválidos

        } catch (IllegalArgumentException e) {

            // Captura la excepción y muestra un mensaje de error

            System.out.printf("%nExcepción al inicializar t6: %s%n", e.getMessage());

        }

    }

    // Método auxiliar que muestra un objeto Tiempo2 en formatos de 24 y 12 horas

    private static void mostrarTiempo(String encabezado, Tiempo2 t) {

        // Imprime el encabezado, luego el tiempo en formato universal (24h) y formato estándar (12h)

        System.out.printf("%s%n %s%n %s%n", encabezado, t.aStringUniversal(), t.toString());

    }

}

// Métodos Obtener (Getters)

// Obtiene el valor de la hora

public int obtenerHora() {

    return hora;

}

// Obtiene el valor del minuto

public int obtenerMinuto() {

    return minuto;

}

// Obtiene el valor del segundo

public int obtenerSegundo() {

    return segundo;

}

// Convierte a String en formato de hora universal (HH:MM:SS)

public String aStringUniversal() {

    return String.format(

        "%02d:%02d:%02d", obtenerHora(), obtenerMinuto(), obtenerSegundo());

}

// Convierte a String en formato de hora estándar (H:MM:SS AM o PM)

public String toString() {

    return String.format("%d:%02d:%02d %s",

        ((obtenerHora() == 0 || obtenerHora() == 12) ? 12 : obtenerHora() % 12),

        obtenerMinuto(), obtenerSegundo(), (obtenerHora() < 12 ? "AM" : "PM"));

}

// Fin de la clase Tiempo2