

Objetivo: aplicar los conceptos de clases e instanciación de objetos con LDP Java, usando el ocultamiento de datos.

**Preguntas a responder:**

**Pgta 1. ¿Desde dónde es posible acceder a la variable time?**

1. Cualquier otra clase
2. java.util package
3. myUtilities package
4. La clase TodaysDate
5. La clase TodaysDate y sus sub clase

**Pgta 2. ¿Quiénes pueden acceder al atributo day?**

day es accesible desde cualquier otra clase en cualquier paquete. Osea está disponible para todos.

**Pgta 3. ¿Qué atributos de la clase tienen el método de acceso más restrictivo?**

1. day
2. month
3. time
4. year

**Pgta 4. ¿Desde dónde se puede acceder al atributo year?**

1. Sólo dentro del package myUtilities.
2. Sub clases de TodaysDate en cualquier package
3. La clase TodaysDate
4. Sub clases de TodaysDate

## Clase TodaysDate por Catalina Quidel y estructura de paquetes

```
1 package org.example;
2
3 import java.util.Calendar;
4 import java.util.GregorianCalendar;
5
6 public class TodaysDate {
7     // variables
8     String time;
9     public int day;
10    private int month;
11    protected int year;
12
13    // methods
14    public void printDateAndTime() {
15        GregorianCalendar calendar = new GregorianCalendar();
16        time = calendar.get(Calendar.HOUR_OF_DAY) + ":" +
17            + calendar.get(Calendar.MINUTE) + ":" +
18            + calendar.get(Calendar.SECOND);
19        day = calendar.get(Calendar.DATE);
20        month = calendar.get(Calendar.MONTH) + 1;
21        year = calendar.get(Calendar.YEAR);
22
23        System.out.println("Time: " + time);
24        System.out.println("Date: " + month + " " + day + " " + year);
25    }
26 }
```

## Compilación exitosa en con la clase pública TestFecha por Catalina Quidel

```
1 package org.example;
2
3 public class TestFecha {
4     public static void main(String[] args) {
5         // crear un objeto llamado hoy del tipo TodaysDate
6         TodaysDate hoy = new TodaysDate();
7
8         // usar
9         hoy.printDateAndTime();
10    }
11 }
12
```

## Pruebas unitarias por Catalina Quidel

```
package org.example;

import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.*;

import java.util.Calendar;
import java.util.GregorianCalendar;

public class TodayDateTest {

    // Prueba para verificar que el atributo day se actualiza correctamente
    @Test
    void testDayAttributeUpdatesCorrectly() {
        TodayDate todayDate = new TodayDate();
        todayDate.printDateAndTime();

        GregorianCalendar calendar = new GregorianCalendar();
        int currentDay = calendar.get(Calendar.DAY_OF_MONTH);

        assertEquals(currentDay, todayDate.day, "El día debería coincidir con el día actual.");

        // Prueba para verificar que el atributo year se actualiza correctamente
        @Test
        void testYearAttributeUpdatesCorrectly() {
            TodayDate todayDate = new TodayDate();
            todayDate.printDateAndTime();

            GregorianCalendar calendar = new GregorianCalendar();
            int currentYear = calendar.get(Calendar.YEAR);

            assertEquals(currentYear, todayDate.year, "El año debería coincidir con el año actual.");

            // Prueba para verificar que el atributo time se actualiza correctamente
            @Test
            void testTimeAttributeUpdatesCorrectly() {
                TodayDate todayDate = new TodayDate();
                todayDate.printDateAndTime();

                GregorianCalendar calendar = new GregorianCalendar();
                int currentHour = calendar.get(Calendar.HOUR_OF_DAY);
                int currentMinute = calendar.get(Calendar.MINUTE);
                int currentSecond = calendar.get(Calendar.SECOND);

                assertEquals(currentHour, todayDate.time.charAt(0), "La hora debería coincidir con la hora actual.");
                assertEquals(currentMinute, todayDate.time.charAt(2), "Los minutos deberían coincidir con los minutos actuales.");
                assertEquals(currentSecond, todayDate.time.charAt(4), "Los segundos deberían coincidir con los segundos actuales.");
            }
        }
    }
}
```

Run: TodayDateTest (org.example) 83 ms. Tests passed: 4 of 4 tests - 83 ms.

testDayAttributeUpdatesCorrectly() 78 ms  
testTimeAttributeNotNullAndNotEmpty() 2 ms  
testTimeAttributeFormat() 1 ms  
testYearAttributeUpdatesCorrectly() 1 ms

Process finished with exit code 0

## El ocultamiento de datos y cómo afecta a las pruebas

Cómo month es una variable private, no se puede acceder directamente desde fuera de la clase TodayDate, incluyendo la clase de prueba unitaria. Una forma para hacer un Test sobre month sería usar el valor del atributo a través de un método Getter público. Getter que realizó Natalia Perez a continuación.

```
import myutilities.TodayDate;

public class TestFecha {
    public static void main(String[] args) {
        TodayDate hoy = new TodayDate();
        hoy.printDateAndTime();
    }
}
```

Run: TestFecha

Process finished with exit code 0

```
package myutilities;

import java.util.Calendar;
import java.util.GregorianCalendar;

public class TodayDate {
    private int day;
    private int month;
    private int year;
    private String time;

    public TodayDate() {
        Calendar calendar = new GregorianCalendar();
        this.day = calendar.get(Calendar.DAY_OF_MONTH);
        this.month = calendar.get(Calendar.MONTH) + 1;
        this.year = calendar.get(Calendar.YEAR);
        this.time = String.format("%tT", calendar);
    }

    public void printDateAndTime() {
        System.out.println("Time: " + time);
        System.out.println("Date: " + day + " " + month + " " + year);
    }

    // Métodos getter y setter solo si los necesitas
    public int getDay() { return day; }
    public int getMonth() { return month; }
    public int getYear() { return year; }
    public String getTime() { return time; }
}
```

```
TestFecha.java  TodaysDateTest.java  TodaysDate.java

1  import myutilities.TodaysDate;
2  import org.junit.jupiter.api.Test;
3  import static org.junit.jupiter.api.Assertions.*;
4
5  class TodaysDateTest {
6
7      @Test
8      void testObjectNotNull() {
9          TodaysDate date = new TodaysDate();
10         assertNotNull(date, message: "El objeto TodaysDate no debería ser nulo");
11     }
12
13     @Test
14     void testDayIsValid() {
15         TodaysDate date = new TodaysDate();
16         assertTrue(condition: date.getDay() >= 1 && date.getDay() <= 31, message: "El día debe estar entre 1 y 31");
17     }
18
19     @Test
20     void testMonthIsValid() {
21         TodaysDate date = new TodaysDate();
22         assertTrue(condition: date.getMonth() >= 1 && date.getMonth() <= 12, message: "El mes debe estar entre 1 y 12");
23     }
24
25     @Test
26     void testYearIsCurrent() {
27         TodaysDate date = new TodaysDate();
28         int currentYear = java.util.Calendar.getInstance().get(java.util.Calendar.YEAR);
29         assertEquals(currentYear, date.getYear(), message: "El año debe ser el actual");
30     }
31
32     @Test
33     void testTimeFormat() {
34         TodaysDate date = new TodaysDate();
35         assertTrue(date.getTime().matches(regex: "\\d{2}:\\d{2}:\\d{2}"), message: "El formato de hora debe ser HH:MM:SS");
36     }
37 }
```

```
Run  TodaysDateTest  x

146 ms  Tests passed: 5 of 5 tests - 146 ms
✓ testTimeFormat() 131 ms
✓ testYearIsCurrent() 7 ms
✓ testObjectNotNull() 4 ms
✓ testMonthsValid() 2 ms
✓ testDaysValid() 2 ms

C:\Users\natal\jdk\openjdk-24\bin\java.exe ...
Process finished with exit code 0
```