********

**I N S T I T U T O T E C N O L Ó G I C O D E L V A L L E D E O A X A C A**

**D e s a r r o l l o d e A p l i c a c i o n e s p a r a D i s p o t i v o s M ó v i l e s**



TEMA: PRINCIPIOS SOLID

ACTIVIDAD: BITÁCORA DE DESARROLLO DE APLICACIÓN MÓVIL

FACILITADOR: AMBROSIO CARDOSO JIMENEZ

NOMBRE DEL ESTUDIANTE: DAMIAN MARTINEZ JIMENEZ

GRADO Y GRUPO: 9 A

CARRERA: INGENIERÍA INFORMÁTICA

FECHA DE ENTREGA: 24/09/2021

**APLICACIÓN 1: APP IMPORTE PREDIAL**

15 septiembre 2021

Se desea diseñar una aplicación que permita calcular el importe total que una persona debe pagar por el impuesto predial, considerando que una persona puede tener varios predios. El costo de cada predio está en función a la zona de ubicación y para ello se cuenta con un catálogo de zonas.

**Clave Zona Costo**

MAR Marginado 2.00

RUR Rural 8.00

URB Urbana 10.00

RES Residencial 25.00

El gobierno municipal está implementando el siguiente programa de descuento:

* Para las personas mayores o iguales de 70 años o madres solteras tiene un 70% de descuento si los pagos se realizan en los meses de enero y febrero y de un 50% en los siguientes meses
* Para el resto de la población hay un descuento del 40% en los meses de enero y febrero

**Paso 1.** Identificar las clases que serán creadas para desarrollar en proyecto, se identificaron 5 y una clase main para realizar las pruebas.

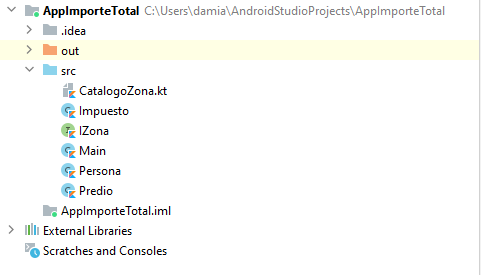


Figura 1. Clases

**Paso 2.** Codificación de la clase persona, con variables propios de una persona como nombre, apellido, fecha de nacimiento, y otros que son necesarios para resolver el problema como: si es madre soltera y el cálculo de la edad.

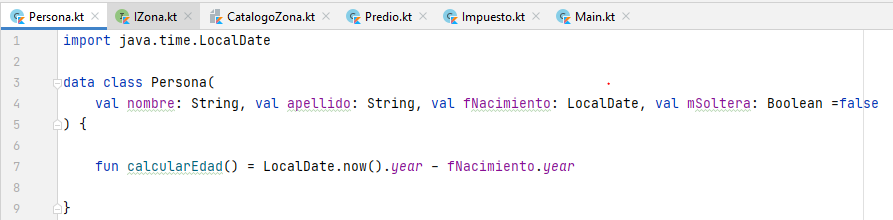


Figura 2. Clase Persona

**Paso 3.** Codificación de una interfaz IZona, contiene las variables: zona, clave y costo.

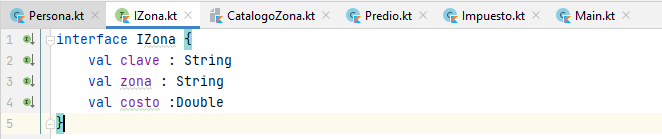


Figura 3. Interfaz IZona

**Paso 4.** Codificacion de un CatalogoZona que contiene información de las zonas.

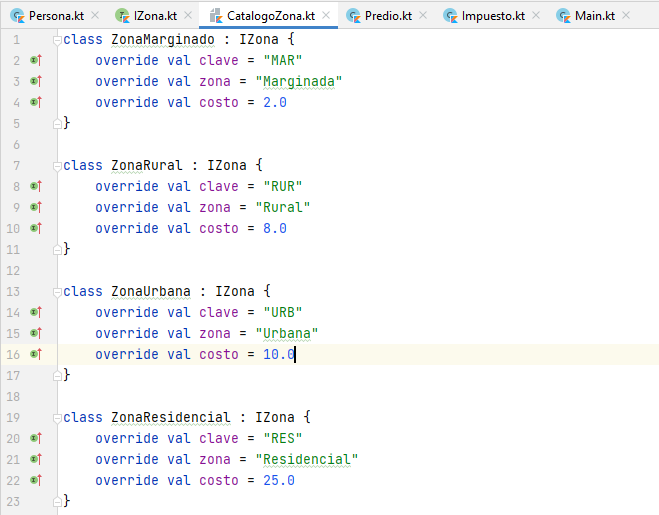


Figura 4. CatagoloZona

**Paso 5.** Codificación de la clase Predio el cual tiene como variables, zona y tamanio, también se encarga de calcular el costo.

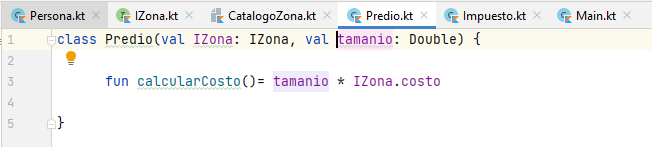


Figura 5. Clase Predio

**Paso 6.** Codificación de la clase Impuesto, tiene variables: descuento, y fecha pago. En esta clase se realizan los cálculos, calculaTotalSinDescuento y calcularImpuestoTotal.

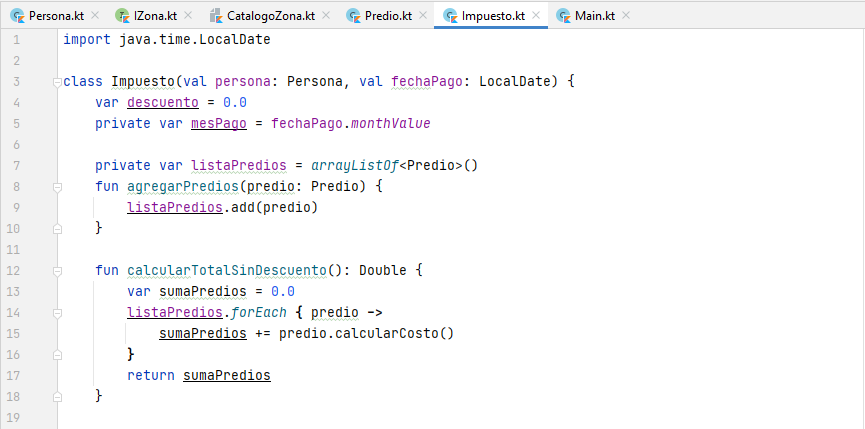


Figura 6. Clase Impuesto

Figura 6. Continuación clase PredialPago

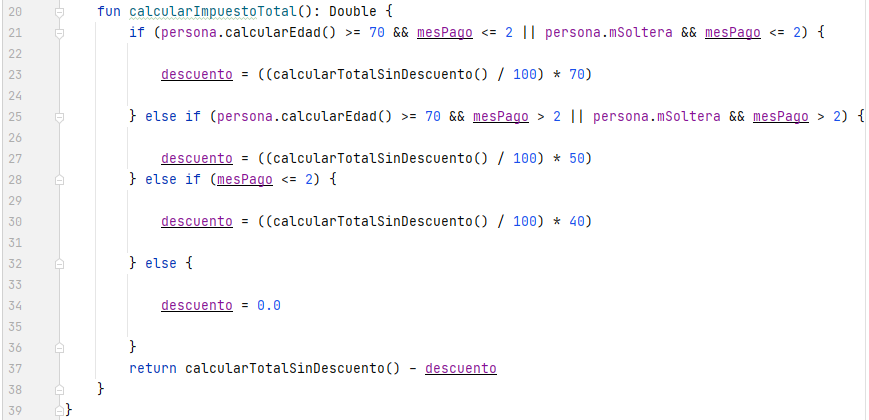


Figura 7. Continuación clase Impuesto

**Paso 7.** Codificación de la clase Main, en el cual se realizaron pruebas, haciendo uso de la librería Junit4

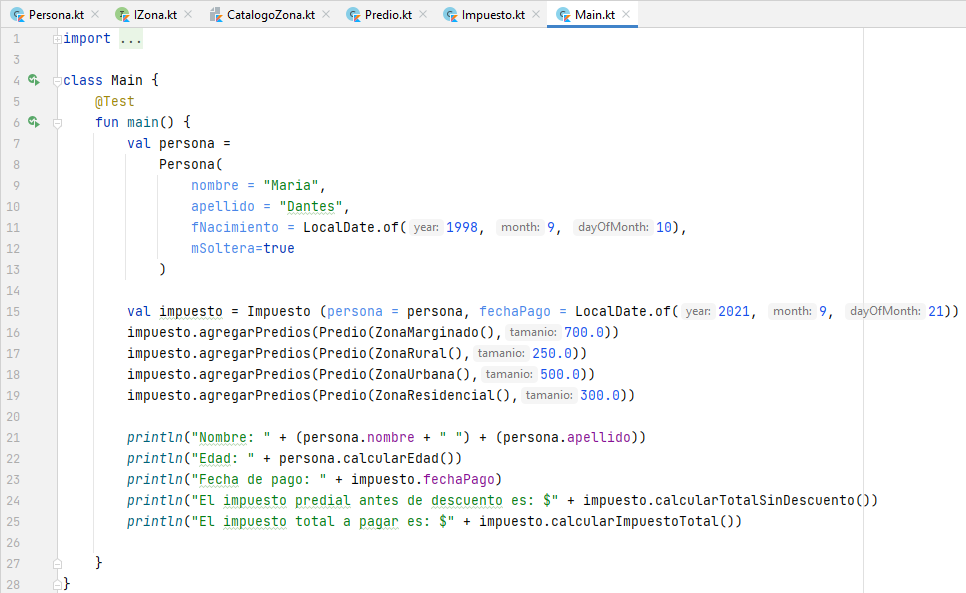


Figura 8. Clase Main para pruebas

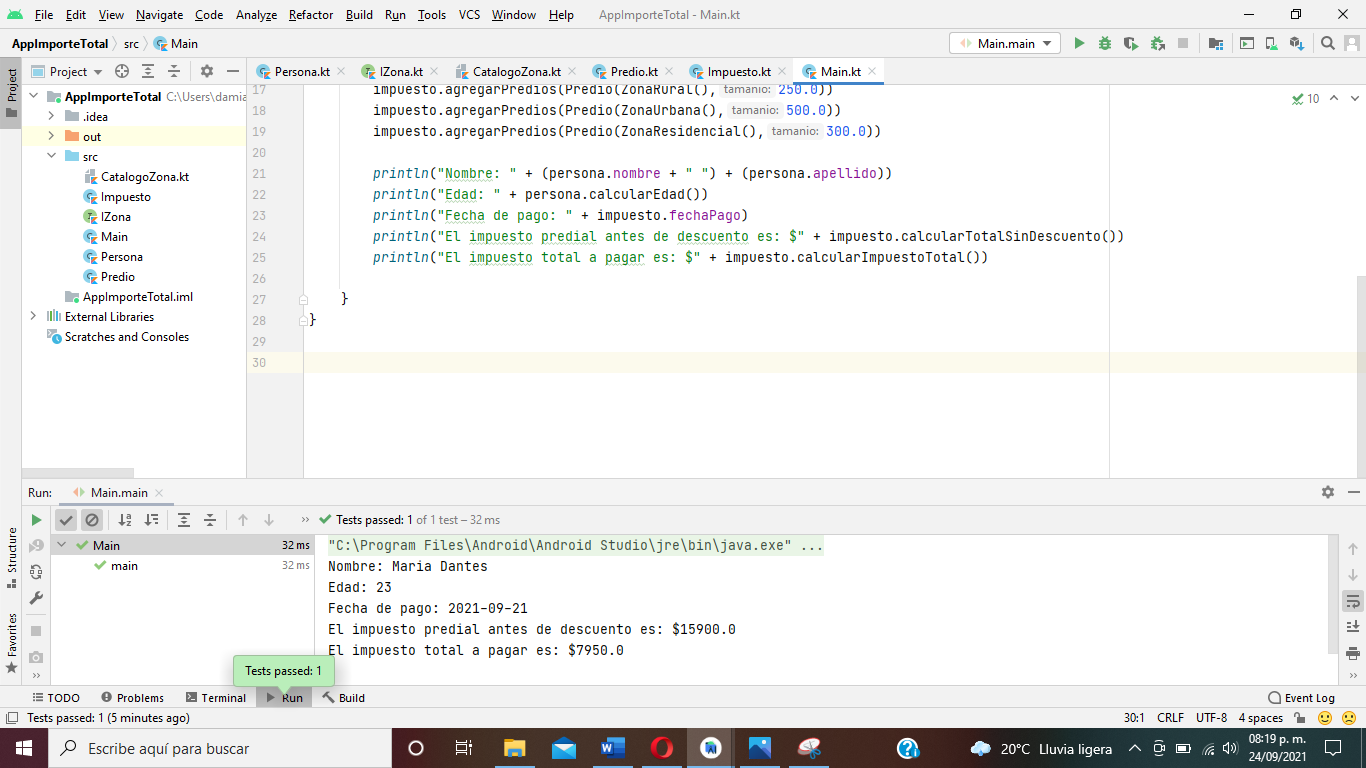
**Resultados**

Figura 9. Prueba 1

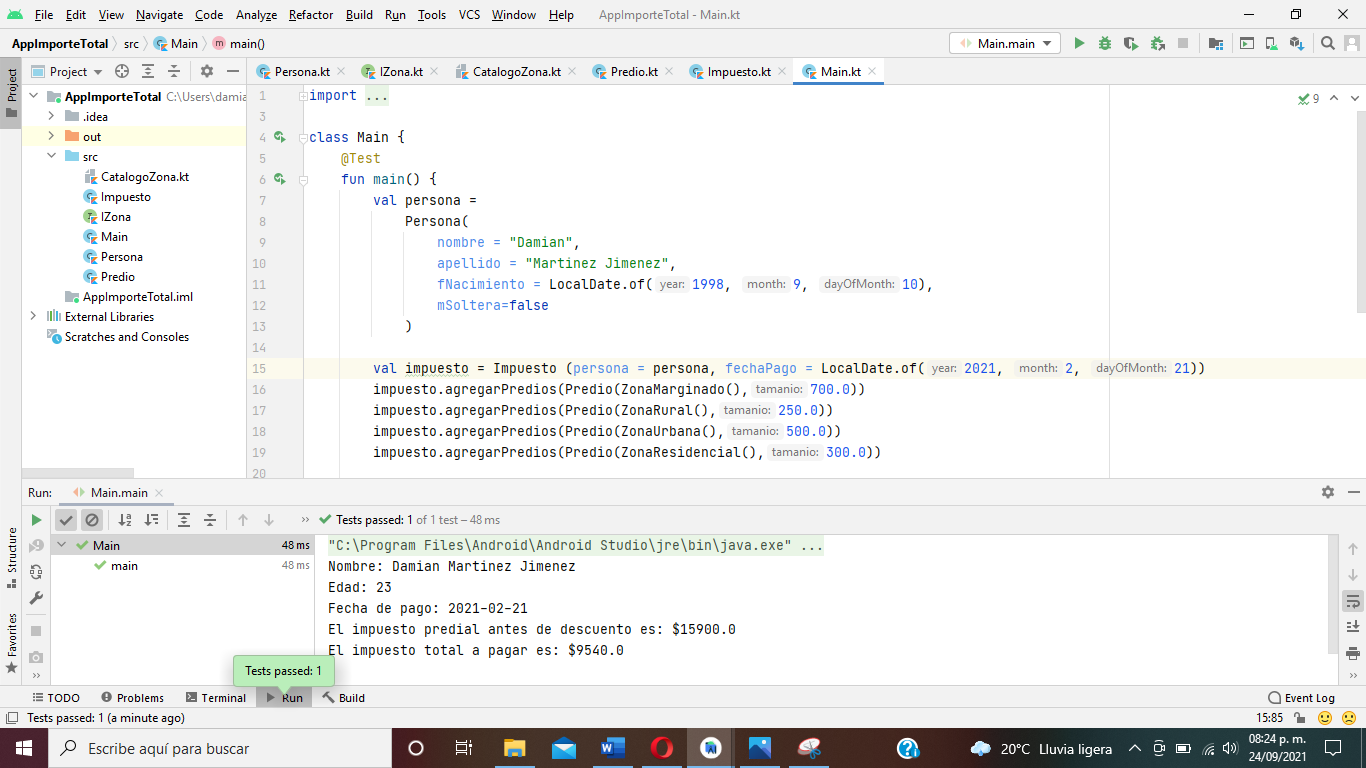


Figura 10. Prueba 2

**Principios SOLID que se cumplen.**

* Principio de responsabilidad única (S):

Cada clase es responsable de realizar una única cosa.

* Principio Open/Closed (O)

La clase Impuesto es la única que se encarga de realizar cálculos, se puede decir que la clase está abierta a nuevas funcionalidades de cálculo, sin afectar el código que ya existe.

* Principio de segregación de interfaces (I)

Ninguna clase depende de métodos que no usa, lo cual ayuda reutilizar el código.

* Principio de inversión de dependencias (D)

La capa de alto nivel no depende de la capa de bajo nivel.

**APLICACIÓN 2: CONTROL TRABAJADORES**

18 septiembre 2021

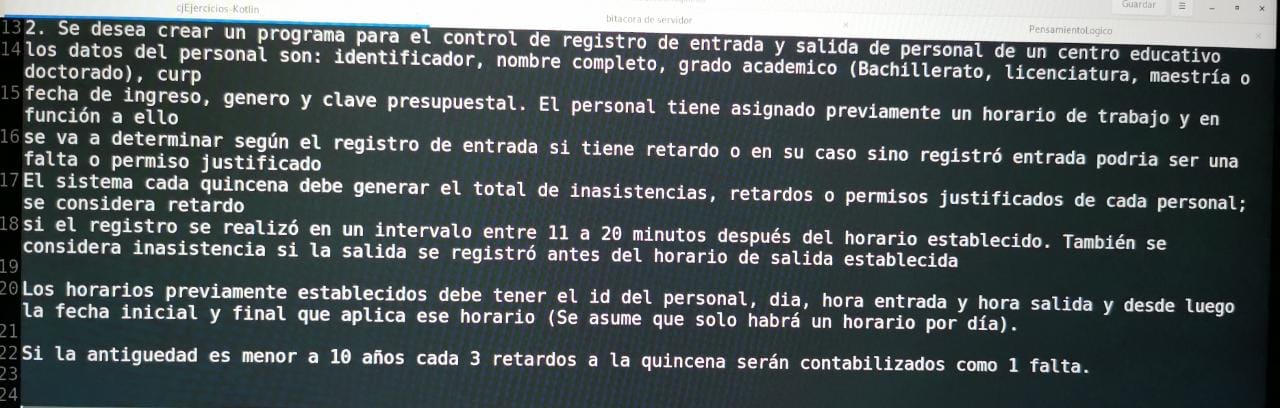


Figura 1. Problema a resolver

**Paso 1.** Se identifican las clases, en este caso, GradoAcademico, Horario, Personal, RangoHoras, Registro, Reporte, y una clase MainControl que ayudara a realizar las pruebas.

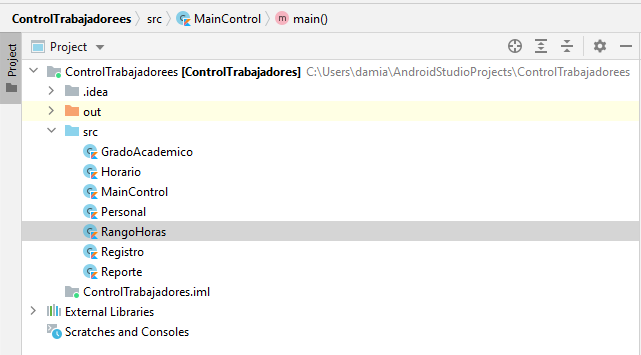


Figura 2. Clases

**Paso 2.** Se crea la clase Personal, con sus respectivas variables.



Figura 3. Clase Personal

**Paso 3.** Se crea la Clase, que contiene como atributos un idHorario, rangoHoras y dia.

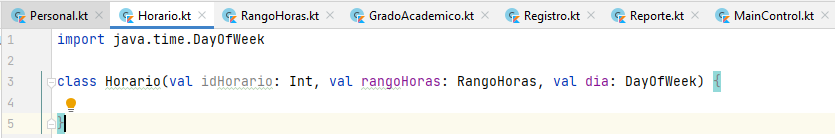


Figura 4. Clase Horario

**Paso 4.** Clase RangoHoras, con atributos, horaInicial y horaFinal.



Figura 5. Clase RangoHoras

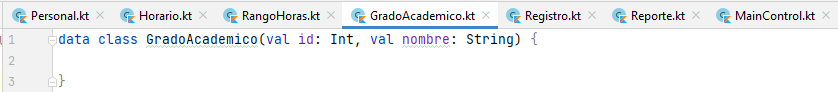
**Paso 5.** Clase GradoAcademico, con variables id y nombre. 

Figura 6. Clase GradoAcademico

**Paso 6.** Clase Registro, con variables personal, fecha y rangoHoras.

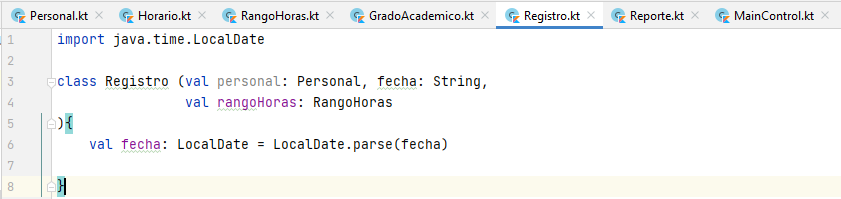


Figura 7. Clase Registro

**Paso 7.** Codificación de la clase Reporte, el cual se encargará de generar el reporte, verificar el horario, calcular retardos y faltas.

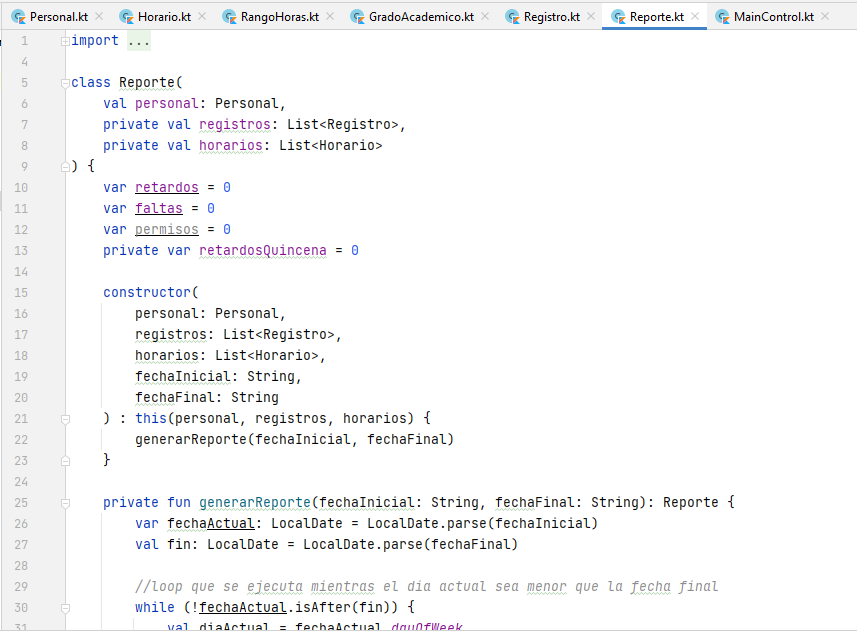


Figura 8. Clase Reporte

**Paso 8.** Codificación de la clase Main, con el cual se realizarán los tests.

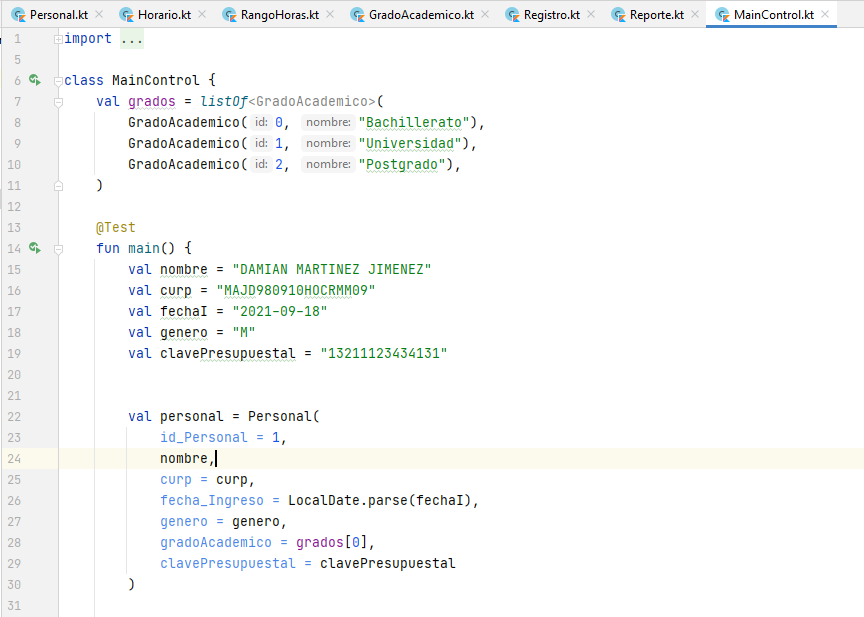


Figura 9. Clase MainControl

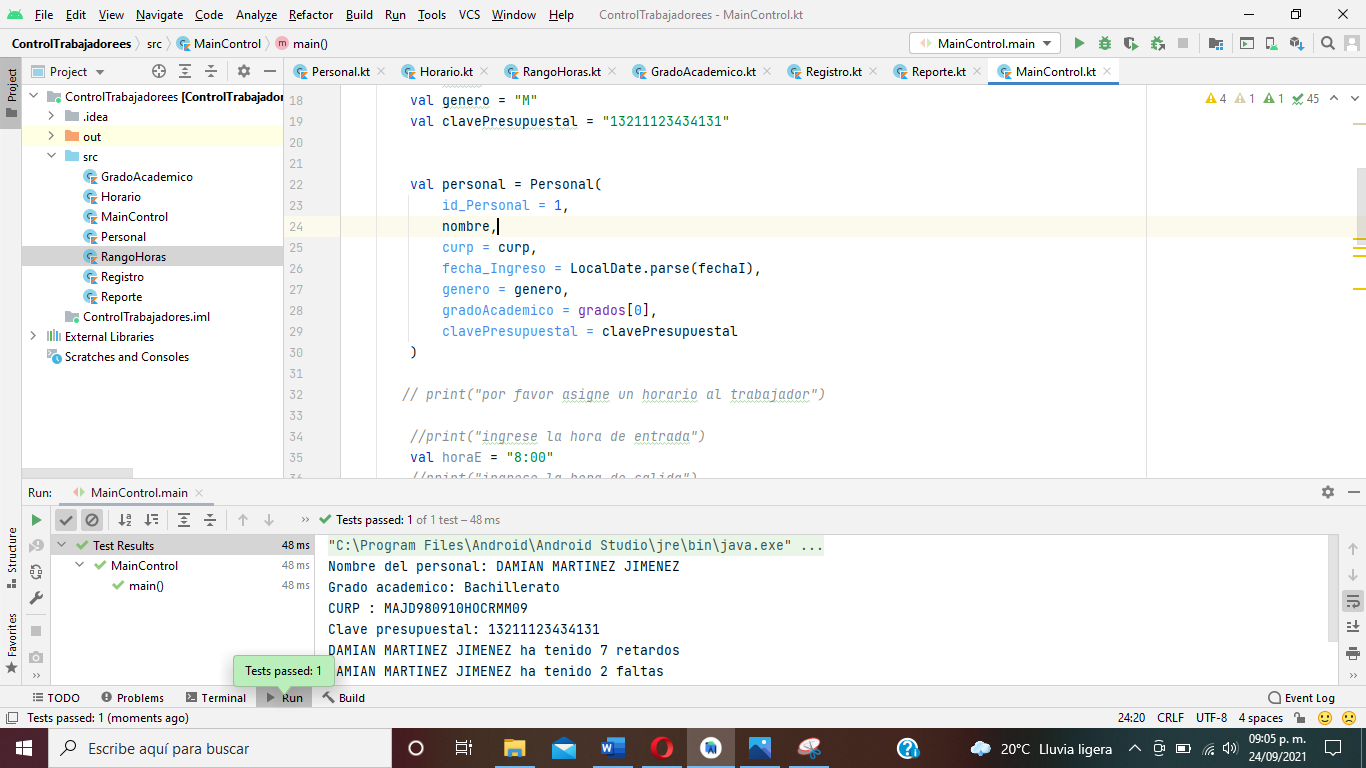
**Resultados**

Figura 10. Prueba

**Principios SOLID que se cumplen.**

* Principio de responsabilidad única (S):

Cada clase es responsable de realizar una única cosa.

* Principio Open/Closed

La clase Reporte es la única que se encarga de realizar cálculos, como el de generar el reporte, calcular faltas y retardos, se puede decir que la clase está abierta a nuevas funcionalidades de cálculo, sin afectar el código que ya existe.

* Principio de segregación de interfaces

Ninguna clase depende de métodos que no usa, lo cual ayuda reutilizar el código.

**Nota:** Para desarrollar estas aplicaciones me reuní con algunos de mis compañeros (Lizbeth Verónica, Emmanuel Josué, Gustavo y yo (Damian)), ya que era complicado en ciertos puntos y entre todos nos estuvimos apoyando a través de reuniones en meet.