**Proyecto Curso: Programación Orientada a Objetos**

**Administración Hotelera**

**Integrantes:**

Ángeles Trujillo, Dennys

Amaya Suarez, Juan Manuel

Herrera Goñi, Danny

Hurtado Arguedas, Giovanni

Vera Villanueva, Carlos Alberto

**2012**

# Presentación

La Programación Orientada a Objetos es una forma de programación que toma practicas de paradigmas de programación previos como la programación estructurada y modular.

La buena POO permite la reutilización de código que finalmente se traduce en menor esfuerzo y cantidad de código por proyecto.

# Índice

Contenido

[Presentación 1](#_Toc326072073)

[Índice 2](#_Toc326072074)

[Introducción 3](#_Toc326072075)

[Caso Propuesto 3](#_Toc326072076)

[Historia de Usuario 3](#_Toc326072077)

[Pruebas Unitarias 5](#_Toc326072078)

[Administracion de Cuenta Test : AdmCuentaTest 5](#_Toc326072079)

[Administrador Cuenta de Usuario : Adm Usuario Test 9](#_Toc326072080)

[Cliente Test 9](#_Toc326072081)

[Cuenta Test 10](#_Toc326072082)

[Plan Test 16](#_Toc326072083)

[Codigo Fuente 17](#_Toc326072084)

[Clases 17](#_Toc326072085)

[Conclusiones 18](#_Toc326072086)

# Introducción

En el siguiente caso práctico debemos ser capaces de aplicar los conceptos de POO para la definición y diseño de un proyecto software, basándonos en el lenguaje de Java.

Para el avance parcial lo que se solicita es desarrollar una historia de usuario con sus respectivos @Test.

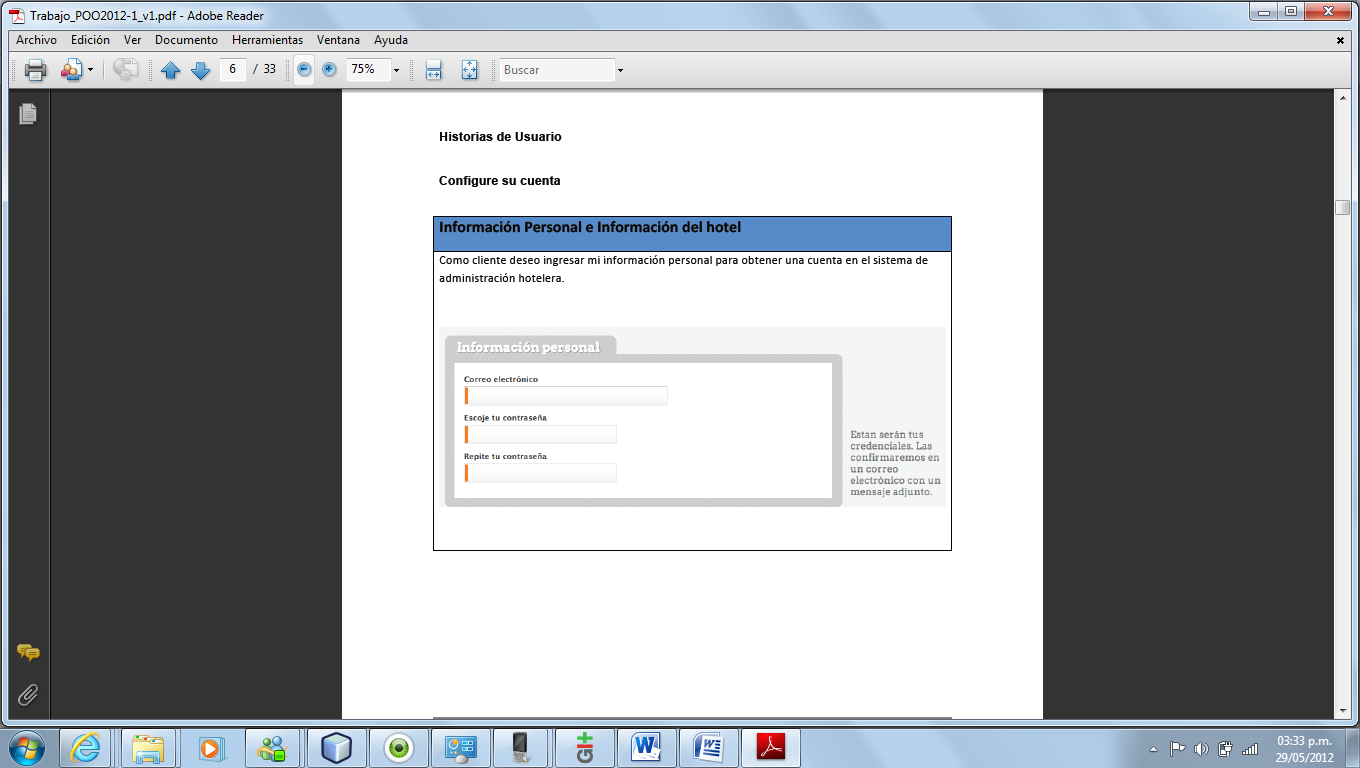
El avance debe estar en el repositorio GitHub, con su respectiva documentación.

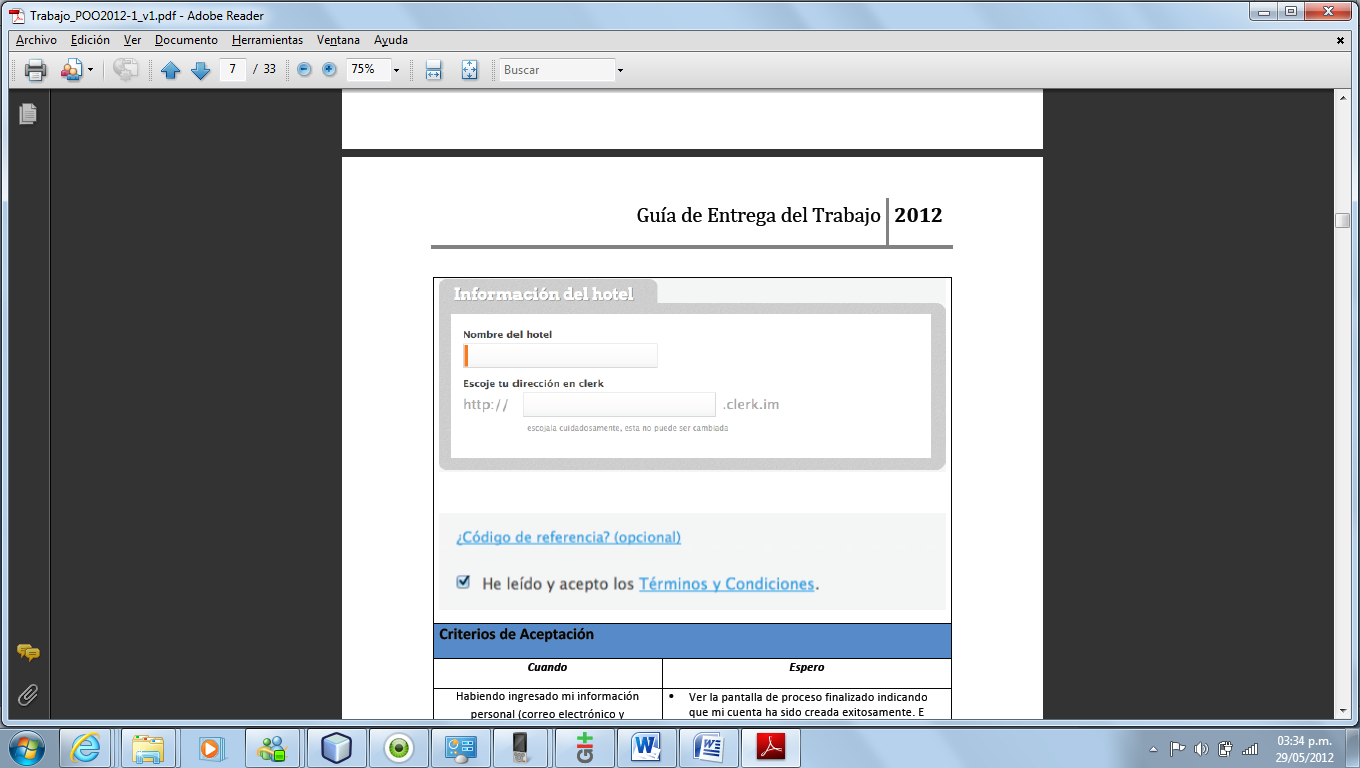
# Caso Propuesto

En el presente ciclo nos toca desarrollar el caso práctico de Administración Hotelera, el cual deberá ser desarrollado durante todo el ciclo académico, dividido en dos entregables.

# Historia de Usuario

La historia de usuario seleccionada es : **Configurar Cuenta en el sistema.**





|  |  |
| --- | --- |
| **Criterios de Aceptación** | |
| *Cuando* | *Espero* |
| * Habiendo ingresado i información personal (correo electrónico y contraseña) y la información del Hotel (nombre y dirección que tendré en el sistema hotelero), además de dar conformidad de haber leído y aceptado los Términos y Condiciones. | * Ver la pantalla de proceso finalizado indicando que mi cuenta ha sido creada exitosamente. E indicando que tengo 24 horas para confirmar el email recibido del sistema hotelero. Se indica además que se tiene el plan gratuito hasta que se cambie a un plan con costo. |
| * No ingreso los campos solicitado o aceptado los Términos y condiciones. | * Un mensaje de error por cada campo no ingresado |
| * Cuando ingreso el nombre del hotel | * Espero hasta 5 recomendaciones sobre el nombre. Una vez seleccionado o ingresado el nombre la dirección web se generara anteponiéndole http:// <<nombre>>.clerk.im |

# Pruebas Unitarias

### Administracion de Cuenta Test : AdmCuentaTest

@Test

public void clienteNoTieneCorreo(){

assertEquals(0, admcuenta.getCuentas().size());

}

@Test

public void clienteYaTieneUncorreoRegistrada(){

admcuenta.registrarCuenta("cvera@hotmail.com","car123","car123","Sheraton","Sheraton-hotel",true,101100);

assertEquals(1, admcuenta.getCuentas().size());

}

@Test

public void clienteYaTieneCorreosRegistradas(){

admcuenta.registrarCuenta("lsanchez@hotmail.com","lui123","lui123","Inkamar","Inkamar-hotel",true,101100);

admcuenta.registrarCuenta("mas2711@hotmail.com","jua123","jua23","Marriot","Marriot-hoterl",true,101100);

admcuenta.registrarCuenta("cvera@hotmail.com","car123","car123","Andes","Andes-hotel",true,101100);

assertEquals(3, admcuenta.getCuentas().size());

}

@Test

public void siNoIngresoCorreoElectronicoMeDebeDarError(){

assertFalse(admcuenta.verificarCorreo(null));

System.out.println("Debe ingresar correo electronico");

}

@Test

public void siIngresoCorreoDebeAceptarlo(){

assertTrue(admcuenta.verificarCorreo("cvera@hotmail.com"));

System.out.println("Correo Ingresado");

}

@Test

public void siNoIngresoPasswordMeDebeDarError(){

assertFalse(admcuenta.verificarPassword(null));

System.out.println("Debe ingresar Password");

}

@Test

public void siIngresoPasswordDebeAceptarlo(){

assertTrue(admcuenta.verificarPassword("lui123"));

System.out.println("Password Ingresado");

}

@Test

public void siNoIngresoRepetirPasswordMeDebeDarError(){

assertFalse(admcuenta.verificarRepetirPassword(null));

System.out.println("Debe Ingresar repetir Password");

}

@Test

public void siIngresoRepetirPasswordDebeAceptarlo(){

assertTrue(admcuenta.verificarRepetirPassword("lui123"));

System.out.println("Repetir Password Ingresado");

}

@Test

public void siNoIngresoNombreDeHotelMeDebeDarError(){

assertFalse(admcuenta.verificarNombreHotel(null));

System.out.println("Debe Ingresar nombre de hotel");

}

@Test

public void siIngresoNombreDeHotelDebeAceptarlo(){

assertTrue(admcuenta.verificarNombreHotel("Andes"));

System.out.println("Nombre de Hotel Ingresado");

}

@Test

public void siNoIngresoConformidadMeDebeDarError(){

assertFalse(admcuenta.verificarConformidad(false));

System.out.println("Debe Activar Conformidad");

}

@Test

public void siIngresoConformidadDebeAceptarlo(){

assertTrue(admcuenta.verificarConformidad(true));

System.out.println("Conformidad Activado");

}

@Test

public void registrarCuentas(){

assertEquals(0, admcuenta.getCuentas().size());

admcuenta.registrarCuenta("cvera@hotmail.com","car123","car123","Sheraton","Sheraton-hotel",true,023100);

assertEquals(1, admcuenta.getCuentas().size());

admcuenta.registrarCuenta("lsanchez@hotmail.com","lui123","lui123","Inkamar","Inkamar-hotel",true,023100);

admcuenta.registrarCuenta("jrodriguez@hotmail.com","jua123","jua23","Marriot","Marrot-hotel",true,023100);

admcuenta.registrarCuenta("pmendoza@hotmail.com","pau123","pau123","Andes","Andes-hotel",true,023100);

assertEquals(4, admcuenta.getCuentas().size());

try{

admcuenta.registrarCuenta("jrodriguez@hotmail.com","jua123","jua23","Marriot","Marriot-hotel",true,023100);

fail();

}catch(Exception ex){

assertEquals("CORREO ELECTRONICO YA REGISTRADO", ex.getMessage());

}

}

}

### Administrador Cuenta de Usuario : Adm Usuario Test

@Test

public void clienteSinCuenta(){

assertEquals(0, cliente.getCuentas().size());

}

### Cliente Test

@Test

public void clienteSinCuenta(){

assertEquals(0, cliente.getCuentas().size());

}

@Test

public void ValidarDatosCliente(){

Cliente cliente = new Cliente("Java Confort", "http://Java.clerk.im");

assertNotNull(cliente.getNombre(), cliente.getEnlace());

}

### Cuenta Test

@Test

public void clienteNoTieneCorreo(){

assertEquals(0, admcuenta.getCuentas().size());

}

@Test

public void clienteYaTieneUncorreoRegistrada(){

admcuenta.registrarCuenta("cvera@hotmail.com","car123","car123","Sheraton","http://Sheraton.com",true,111500);

assertEquals(1, admcuenta.getCuentas().size());

}

@Test

public void clienteYaTieneCorreosRegistradas(){

admcuenta.registrarCuenta("lsanchez@hotmail.com","lui123","lui123","Inkamar","",true,111500);

admcuenta.registrarCuenta("jrodriguez@hotmail.com","jua123","jua23","Marriot","",true,111500);

admcuenta.registrarCuenta("cvera@hotmail.com","car123","car123","Andes","",true,121500);

assertEquals(3, admcuenta.getCuentas().size());

}

@Test

public void siNoIngresoCorreoElectronicoMeDebeDarError(){

assertFalse(admcuenta.verificarCorreo(null));

System.out.println("Debe ingresar correo electronico");

}

@Test

public void siIngresoCorreoDebeAceptarlo(){

assertTrue(admcuenta.verificarCorreo("cvera@hotmail.com"));

System.out.println("Correo Ingresado");

}

@Test

public void siNoIngresoPasswordMeDebeDarError(){

assertFalse(admcuenta.verificarPassword(null));

System.out.println("Debe ingresar Password");

}

@Test

public void siIngresoPasswordDebeAceptarlo(){

assertTrue(admcuenta.verificarPassword("lui123"));

System.out.println("Password Ingresado");

}

@Test

public void siNoIngresoRepetirPasswordMeDebeDarError(){

assertFalse(admcuenta.verificarRepetirPassword(null));

System.out.println("Debe Ingresar repetir Password");

}

@Test

public void siIngresoRepetirPasswordDebeAceptarlo(){

assertTrue(admcuenta.verificarRepetirPassword("lui123"));

System.out.println("Repetir Password Ingresado");

}

@Test

public void siNoIngresoNombreDeHotelMeDebeDarError(){

assertFalse(admcuenta.verificarNombreHotel(null));

System.out.println("Debe Ingresar nombre de hotel");

}

@Test

public void siIngresoNombreDeHotelDebeAceptarlo(){

assertTrue(admcuenta.verificarNombreHotel("Andes"));

System.out.println("Nombre de Hotel Ingresado");

}

@Test

public void siNoIngresoConformidadMeDebeDarError(){

assertFalse(admcuenta.verificarConformidad(false));

System.out.println("Debe Activar Conformidad");

}

@Test

public void siIngresoConformidadDebeAceptarlo(){

assertTrue(admcuenta.verificarConformidad(true));

System.out.println("Conformidad Activado");

}

@Test

public void registrarCuentas(){

assertEquals(0, admcuenta.getCuentas().size());

admcuenta.registrarCuenta("cvera@hotmail.com","car123","car123","Sheraton","http://Sheraton.com",true,123456);

assertEquals(1, admcuenta.getCuentas().size());

admcuenta.registrarCuenta("lsanchez@hotmail.com","lui123","lui123","Inkamar","",true,051324);

admcuenta.registrarCuenta("jrodriguez@hotmail.com","jua123","jua23","Marriot","",true,062500);

admcuenta.registrarCuenta("pmendoza@hotmail.com","pau123","pau123","Andes","",true,021314);

assertEquals(4, admcuenta.getCuentas().size());

try{

admcuenta.registrarCuenta("jrodriguez@hotmail.com","jua123","jua23","Marriot","",true,101213);

fail();

}catch(Exception ex){

assertEquals("CORREO ELECTRONICO YA REGISTRADO", ex.getMessage());

}

}

//Codigo de Dennys

String mail="dennys@ata.com";;

String pass="123";

String rpass="123";

String hotel="Kenshin";

String web="Kenshin.com";

boolean Conformidad=false;

long HoraCreacion=111213;

Cuenta cuenta1=new Cuenta(mail,pass,rpass,hotel,web,Conformidad,HoraCreacion);

@Test

public void DebeIngresarCorreoElectronico(){

// Cuenta cuenta1=new Cuenta("","","","","",false);

assertNotNull("Debe Ingresar Correo", cuenta1.getCorreo());

}

@Test

public void DebeIngresarContraseña(){

// Cuenta cuenta1=new Cuenta("","","","","",false);

assertNotNull("Debe Ingresar Contraseña", cuenta1.getPassWord());

}

@Test

public void DebeIngresarRepetirContraseña(){

// Cuenta cuenta1=new Cuenta("","","","","",false);

assertNotNull("Debe Ingresar Repetir Contraseña", cuenta1.getRepetirPassword());

}

@Test

public void DebeIngresarNombreDelHotel(){

// Cuenta cuenta1=new Cuenta("","","","","",false);

assertNotNull("Debe Ingresar Nombre del Hotel", cuenta1.getNombreHotel());

}

@Test

public void DebeIngresarWebDelHotel(){

// Cuenta cuenta1=new Cuenta("","","","","",false);

assertNotNull("Debe Ingresar Web del Hotel", cuenta1.getWebHotel());

}

@Test

public void DebeIngresarConformidadTerminosYCondiciones(){

// Cuenta cuenta1=new Cuenta("","","","","",false);

assertNotNull("Debe Aceptar Terminos y Condiciones", cuenta1.isConformidad());

}

### Plan Test

@Test

public void ValidarTipoPlan(){

Plan plan = new Plan(true);

assertSame(plan.isEstado(),true);

}

@Test

public void TipoPlan(){

Plan plan = new Plan(true);

String mensaje = "Plan Gratuito";

assertSame(mensaje,plan.TipoPlan());

}

# Codigo Fuente

Se encuentra en el repositorio GitHUb.

# Clases

#### Cliente

Nombre String

Enlace String

#### Cuenta

Correo String

Password String

NombreHotel String

WebHotel String

Conformidad Boolena

HoraCreacion Long

#### Plan

Nombre Cliente

Estado Boolean

Cliente Cliente

# Conclusiones

El trabajo nos demuestra que existen diferentes maneras de ver el mundo real y los requerimientos del día a día, también nos valida que el paradigma del POO es más efectivo al paradigma de la programación estructurada, nos ahorra código y procesos para el desarrollo de ciertos proyectos, pero es cuestión de practicarlo.

A su vez el trabajo nos demuestra que lo más difícil no es programarlo, ni diseñarlo, sino encontrar el tiempo para poder trabajar en equipo debido a los diferentes labores diarias y el tiempo corto que se nos presento para el trabajo parcial; pero consideramos que es un buen inicio para el desarrollo del trabajo final.