

**UNIVERSIDAD DE LAS FUERZAS ARMADAS-ESPE SEDE SANTO DOMINGO**  
**DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN - DCCO-SS**  
**CARRERA DE INGENIERÍA EN TECNOLOGÍAS DE LA INFORMACIÓN**

**PERIODO** : Abril 2023 – Octubre 2023

**ASIGNATURA** : POO

**TEMA** : Guía de Ejercicios

**NOMBRES** : Kelvin Quezada

**NIVEL-PARALELO** : Segundo “A”

**DOCENTE** : Ing. Cevallos Farias Javier Moyota

**FECHA DE ENTREGA** : 14/08/2023



**SANTO DOMINGO - ECUADOR**  
**2023**

## Guía de Informe de Ejercicio

### Guía De Programación modular

1. **Guía de programación modular ejercicio 1:** Una compañía de seguros ofrece a sus clientes cuatro tipos de seguros de sepelio

2. Tipo	Máximo número de Personas	Pagos mensuales (s/.)
A	8	40
B	6	30
C	4	20
D	2	10

Si el cliente asegura a más personas de la indicadas en el cuadro anterior tendrá que pagar S/8.00 mensuales Por cada persona Adicional si es que el seguro es de tipo A o B Será de S/5.00 mensuales por cada persona adicional si es que seguro es de tipo COD calcular el monto anual que tiene que pagar un determinado cliente.

#### Paso 1: Main de Programa

Primeramente creamos una clase ProModular que será el main donde vamos a importar las librerías y definir la clase principal.

```
public class ProModular {  
    public static void main(String[] args) {  
        Scanner leer=new Scanner(System.in);  
    }  
}
```

#### Paso 2: Definir funciones para los modulos

Definimos funciones para calcular los modulares ya que la funcionMaximoPersonas calcula el numero máximo de personas permitidas en una categoría.

## Guía de Informe de Ejercicio

```
/*Esta funcion calcula y devuelve el numero maximo de personas permitidas en una categoria especificada.
Recibe como entrada el codigo de categoria 'cate' y devuelve el numero maximo de personas 'maxper'*/
static int MaximoPersonas(String cate){
    // Inicializa la variable que almacenara el numero maximo de personas permitidas.
    int maxper=0;
    // Utiliza un switch para asignar el valor correcto de 'maxper' segun la categoria especificada.
    switch(cate){
        // Nos dara a conocer las categoria
        case "A": maxper=8;break;
        case "B": maxper=6;break;
        case "C": maxper=4;break;
        case "D": maxper=2;break;
    }
    // Devuelve el numero maximo de personas permitidas segun la categoria especificada.
    return maxper;
}
```

La función PagoMensual calculara el pago mensual máximo en una categoría.

```
/*Esta funcion calcula y devuelve el numero maximo de PagoMensual permitidas en una categoria especificada.
Recibe como entrada el codigo de categoria 'cate' y devuelve el numero maximo de pagoMensual 'maxper'*/
static double PagoMensual(String cate){
    double pm=0;
    switch(cate){
        case "A": pm=40.00;break;
        case "B": pm=30.00;break;
        case "C": pm=20.00;break;
        case "D": pm=10.00;break;
    }
    // Devuelve el numero maximo de personas permitidas segun la categoria especificada.
    return pm;
}
```

La función CalcAumento calculara el aumento en el pago mensual según la categoría.

```
static double CalcAumento(String cate){
    double aum=0;
    switch(cate){
        case "A": aum=8.00;break;
        case "B": aum=8.00;break;
        case "C": aum=5.00;break;
        case "D": aum=5.00;break;
    }
    return aum;
}
```

La función imprimir los resultados del programa.

```
//creamos un modulo para imprimir
static void Imprimir(double pmi,double pmf,double pa){
    System.out.println("\nEl monto mensual iniciar a pagar es: "+pmi+
        "\nEl monto mensual final a pagar es : "+pmf+
        "\nEl monto mensual a pagar es : "+pa);
}
```

## Guía de Informe de Ejercicio

### Paso 3: Programa Principal

Declaramos variables necesarias para los cálculos

```
// Creamos un Scanner para leer la entrada del usuario
// llamaremos nuestras variables
double pmi, pmf, pa, aum;
// creamos nuestra variable
int cant_personas, maxper;
String cate;
```

Solicitamos la entrada del usuario que ingrese la categoría y la cantidad de personas.

```
System.out.println(x: "Ingrese la categoria A, B, C, D");
cate=leer.next();
System.out.println(x: "Ingrese la cantidad de personas: ");
cant_personas=leer.nextInt();
```

Realizamos los cálculos modulares utilizando funciones definidas anteriormente para realizar los cálculos.

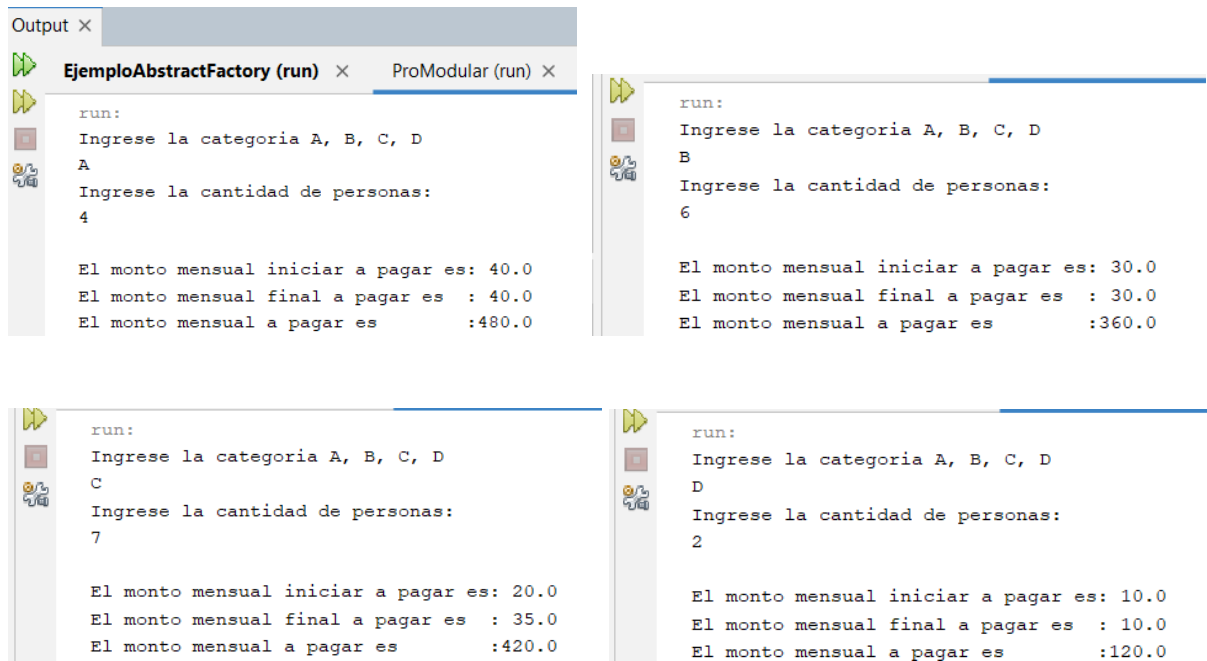
```
// validamos nuestros modulos
maxper=MaximoPersonas(cate);
pmi=PagoMensual(cate);
aum=CalcAumento(cate);
// creamos una condicional donde la cantidad de persona es mayor a maxper
// entonces el pmf=pmi mas a la cantidad de persona - maxper se multiplica por el aum
if (cant_personas > maxper) {
    pmf=pmi+((cant_personas-maxper)*aum);
    // oh sino el pmf es =pmi
} else {
    pmf=pmi;
}
// calculamos el pago anual
pa=pmf*12;
```

Llamamos a la función de imprimir para conocer los resultados Imprimir.

```
// Llamamos a la función para imprimir los resultados
Imprimir(pmi, pmf, pa);
```

**Imprimir los resultados del programa**

## Guía de Informe de Ejercicio



```
Output x
EjemploAbstractFactory (run) x ProModular (run) x

run:
Ingrese la categoria A, B, C, D
A
Ingrese la cantidad de personas:
4

El monto mensual iniciar a pagar es: 40.0
El monto mensual final a pagar es : 40.0
El monto mensual a pagar es :480.0

run:
Ingrese la categoria A, B, C, D
B
Ingrese la cantidad de personas:
6

El monto mensual iniciar a pagar es: 30.0
El monto mensual final a pagar es : 30.0
El monto mensual a pagar es :360.0

run:
Ingrese la categoria A, B, C, D
C
Ingrese la cantidad de personas:
7

El monto mensual iniciar a pagar es: 20.0
El monto mensual final a pagar es : 35.0
El monto mensual a pagar es :420.0

run:
Ingrese la categoria A, B, C, D
D
Ingrese la cantidad de personas:
2

El monto mensual iniciar a pagar es: 10.0
El monto mensual final a pagar es : 10.0
El monto mensual a pagar es :120.0
```

## Guía de Patrones de diseños

**Guía patrones de diseño Guía del ejercicio #2:** Patrón de diseño creacional abstract factory En este programa hacemos uso del patrón de diseño, tenemos relaciones entre sí, teniendo 2 paquetes. En el primer paquete hacemos uso de las clases y en el segundo es donde se desarrolla el patrón de diseño. El patrón de diseño abstract factory permite la creación, la instanciación de objetos de distintas clases que tienen métodos en común.

Mediante un software realizaremos un ejercicio de información donde la empresa nacional podrá elegir las opciones de información que requiere saber por medio de interfaz para crear conjuntos o familias de objetos relacionados que ayudan al sitio de ayuda.

Definición de la interfaz ServicioInformatico: Se define una interfaz que declara los métodos que deben ser implementados por diferentes tipos de servicios informáticos.

## Guía de Informe de Ejercicio

```
// Definición de la interfaz ServicioInformatico
public interface ServicioInformatico {
    //declaramos los metodos
    public void asignarTrabajo();
    public void indicarFechaEntrega();
    public void informarSobrePago();
}
```

Implementación de clases de servicios concretos (ServicioDesign, ServicioSoftwareEducativo, ServicioWebsites): Se implementan las clases concretas que implementan la interfaz ServicioInformatico y proporcionan la implementación específica para los métodos.

```
public class ServicioSoftwareEducativo implements ServicioInformatico {
    @Override
    public void asignarTrabajo() {
        System.out.println(x: "Nuestros programadores han sido informados del programa que deben realizar.");
    }
    @Override
    public void indicarFechaEntrega() {
        System.out.println(x: "Se ha fijado como fecha de entrega el día 01/10/2023.");
    }
    @Override
    public void informarSobrePago() {
        System.out.println(x: "El monto a pagar sera proporcional a la cantidad de estudiantes que haran uso del software.");
    }
}
```

```
public class ServicioWebsites implements ServicioInformatico {
    //creamos tres metodos que estan sobrescribiendo es nuestra clase interfaz de ServicioInformatico
    @Override
    public void asignarTrabajo() {
        System.out.println(x: "El personal encargado del desarrollo de sitios web ha aceptado el trabajo.");
    }
    @Override
    public void indicarFechaEntrega() {
        System.out.println(x: "El sitio web con Responsabilida Designara terminado el día 26/11/2023.");
    }
    @Override
    public void informarSobrePago() {
        System.out.println(x: "El monto a pagar no incluye el pago por dominio y hosting.");
    }
}
```

```
public class ServicioDesign implements ServicioInformatico {
    //creamos tres metodos que estan sobrescribiendo es nuestra clase interfaz de ServicioInformatico
    @Override
    public void asignarTrabajo() {
        System.out.println(x: "El trabajado ha sido asignado a disenadores graficos disponibles.");
    }
    @Override
    public void indicarFechaEntrega() {
        System.out.println(x: "Ellos han determinado terminar el trabajo como maximo para el día 09/08/2023.");
    }
    @Override
    public void informarSobrePago() {
        System.out.println(x: "Debe realizar el pago en efectivo al momento de recoger el logo completamente terminado.");
    }
}
```

## Guía de Informe de Ejercicio

Definición de la interfaz ServicioFactory: Se define una interfaz de fábrica abstracta que declara un método para crear instancias de ServicioInformatico.

```
public interface ServicioFactory {  
    // Declaración del método crearServicio()  
    // que devuelve un objeto del tipo ServicioInformatico  
    public ServicioInformatico crearServicio();  
}
```

Implementación de clases de fábrica concretas (DesignFactory, SoftwareFactory, WebsiteFactory): Se implementan las clases de fábrica concretas que implementan la interfaz ServicioFactory y proporcionan la implementación para el método crearServicio(), creando instancias de diferentes tipos de servicios.

```
public class WebsiteFactory implements ServicioFactory {  
    // Sobrescribir el método crearServicio() de la interfaz  
    @Override  
    public ServicioInformatico crearServicio() {  
        // Crear y devolver una instancia de ServicioWebsites  
        return new ServicioWebsites();  
    }  
}  
  
public class DesignFactory implements ServicioFactory {  
    // Sobrescribir el método crearServicio() de la interfaz  
    @Override  
    public ServicioInformatico crearServicio() {  
        // Crear y devolver una instancia de ServicioDesign  
        return new ServicioDesign();  
    }  
}  
  
public interface ServicioFactory {  
    // Declaración del método crearServicio()  
    // que devuelve un objeto del tipo ServicioInformatico  
    public ServicioInformatico crearServicio();  
}
```

Implementación de la clase Principal: Se crea la clase principal que contiene el método main. En este método, se muestra un menú de opciones al usuario y se utiliza un ciclo do-while para repetir el proceso hasta que el usuario elija salir. Dependiendo de la opción elegida por el

## Guía de Informe de Ejercicio

usuario, se llama al método `usarServicio()` pasando la fábrica correspondiente este método toma una instancia de `ServicioFactory` como parámetro y utiliza dicha fábrica para crear una instancia de `ServicioInformatico`. Luego, llama a los métodos de la interfaz `ServicioInformatico` en la instancia creada para ejecutar las operaciones correspondientes al servicio.

```
public static void main(String[] args) {
    Scanner leer = new Scanner(System.in);
    int opcion;
    do {
        System.out.print(
            "MENU DE OPCIONES\n"
            + "1. Solicitar servicio de diseno grafico.\n"
            + "2. Solicitar desarrollo de software educacional.\n"
            + "3. Solicitar creacion de sitios web.\n"
            + "4. Cerrar programa.\n"
            + "Seleccione opcion: "
        );
        opcion=leer.nextInt();
        switch (opcion) {
            case 1:
                // Llamamos al método usarServicio pasando una nueva instancia de DesignFactory
                usarServicio(new DesignFactory());
                break;
            case 2:
                // Llamamos al método usarServicio pasando una nueva instancia de SoftwareFactory
                usarServicio(new SoftwareFactory());
                break;
            case 3:
                // Llamamos al método usarServicio pasando una nueva instancia de WebsiteFactory
                usarServicio(new WebsiteFactory());
                break;
            case 4:
                System.out.println(x: "A finalizado el programa ");
                break;
            default:
                break;
        }
    } while (opcion!=4);
}

//creamos un metodo para usar un servicio
public static void usarServicio(ServicioFactory factory) {
    // Creamos una instancia de ServicioInformatico
    //utilizando la fábrica dada
    ServicioInformatico servicio = factory.crearServicio();
    // Llamamos a los métodos de la interfaz
    //ServicioInformatico en la instancia creada
    servicio.asignarTrabajo();
    servicio.indicarFechaEntrega();
    servicio.informarSobrePago();
}
}
```



## Guía de Informe de Ejercicio

Imprimimos los resultados del programa

```
Output - EjemploAbstractFactory (run) x
run:
MENU DE OPCIONES
---- -- ----->
1. Solicitar servicio de diseno grafico.
2. Solicitar desarrollo de software educacional.
3. Solicitar creacion de sitios web.
4. Cerrar programa.
Seleccione opcion: 1
El trabajado ha sido asignado a disenadores graficos disponibles.
Ellos han determinado terminar el trabajo como maximo para el dia 09/08/2023.
Debe realizar el pago en efectivo al momento de recoger el logo completamente terminado.
MENU DE OPCIONES
---- -- ----->
1. Solicitar servicio de diseno grafico.
2. Solicitar desarrollo de software educacional.
3. Solicitar creacion de sitios web.
4. Cerrar programa.
Seleccione opcion: 2
Nuestros programadores han sido informados del programa que deben realizar.
Se ha fijado como fecha de entrega el dia 01/10/2023.
El monto a pagar sera proporcional a la cantidad de estudiantes que haran uso del software.
MENU DE OPCIONES
---- -- ----->
1. Solicitar servicio de diseno grafico.
2. Solicitar desarrollo de software educacional.
3. Solicitar creacion de sitios web.
4. Cerrar programa.
Seleccione opcion: 3
El personal encargado del desarrollo de sitios web ha aceptado el trabajo.
El sitio web con Responsabilida Designara terminado el día 26/11/2023.
El monto a pagar no incluye el pago por dominio y hosting.
```