NOMBRE:
APELLIDOS:
NIA:
GRUPO:

Primer parcial

2ª Parte: Problemas (7 puntos sobre 10)

Duración: 70 minutos
Puntuación máxima: 7 puntos
Fecha: 25 de marzo de 2021

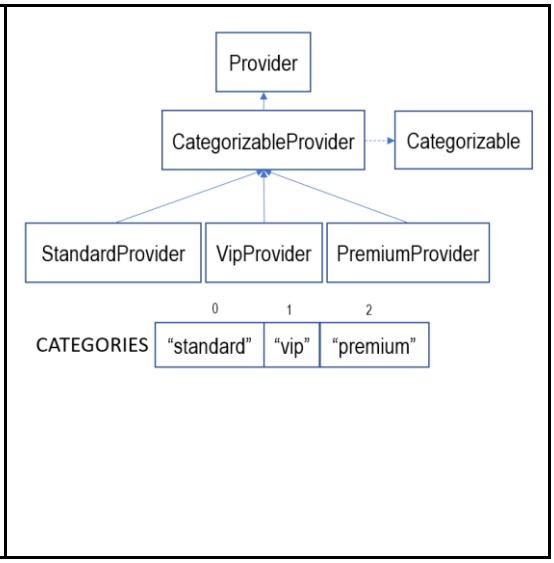
Instrucciones para el examen:

- No se permite el uso de libros o apuntes, ni tener teléfonos móviles u otros dispositivos electrónicos encendidos. Incumplir cualquiera de estas normas puede ser motivo de expulsión inmediata del examen.

Ejercicio 1 (5 / 7 puntos)

El programa de Gestión del almacén que has creado para tu proyecto ha decidido trabajar con tres tipos de proveedores (`Provider`): Los proveedores estándar (`StandardProvider`) que tienen un 10% de descuento al comprar productos del almacén, los proveedores VIP (`VipProvider`) que tienen un 20% y los proveedores premium (`PremiumProvider`) que tienen un 30. Para ello la clase `Provider` se ha especializado siguiendo la jerarquía que se muestra en la figura

```
import java.util.Scanner;
import java.io.*;
public class Provider implements Comparable{
      private int vat;
      private String name;
      private String taxAddress;
      private Person contactPerson;
      public Provider(int vat, String name,
                      String taxAddress,
                      Person contactPerson){
          setVat(vat);
          setName(name);
          setTaxAddress(taxAddress);
          setContactPerson(contactPerson);
      }
      //getters and setters
      //toString method
}
```



Se pide:

Interfaz Categorizable

- Implementa la interfaz `Categorizable` que permita modelar las categorías de servicio de esta y otras aplicaciones.
- Esta interfaz tiene una constante numérica para representar cada categoría: `0-STANDARD, 1-VIP, 2-PREMIUM`. Una constante `CATEGORIES` de tipo array de Strings con los nombres en formato String de cada una de las categorías (ver figura).
- Un método getCategory para devolver el String con el nombre de la categoría

- Un método `getType` para devolver el número correspondiente a cada categoría

Clase CategorizableProvider. Declaración y atributos

- Declara la clase `CategorizableProvider` que hereda de provider e implementa la interfaz `Categorizable`.
- Esta clase tiene dos atributos de tipo entero no accesibles desde ninguna otra clase: `discount` de tipo entero para representar el descuento aplicable a cada categoría y `type` que representa el número correspondiente a cada categoría. Este atributo sólo puede tomar los valores indicados en la interfaz `Categorizable`

Clase CategorizableProvider. Métodos

- Implementa el método `getDiscount` que devuelve el valor del atributo correspondiente. No existe método `setDiscount`.
- Implementa el método `getCategory`() que devuelve como resultado el String correspondiente al tipo numérico almacenado en el atributo `type` (NOTA: No puedes utilizar literales, utiliza en su lugar las constantes declaradas en la interfaz `Category`). No existe método `setCategory`
- Crea un constructor que reciba los mismos parámetros que la clase padre más dos parámetros adicionales de tipo entero: el tipo de categoría y el descuento aplicable a dicha categoría). (Recuerda que **`no existen los métodos set`**)
- Implementa un método `toString` que devuelva la información del proveedor seguida del tipo y el descuento aplicable. Puedes asumir que la clase `Provider` ya tiene un método `toString` que imprime toda la información del proveedor excepto la categoría y el descuento. No es necesario que implementes el `toString` de la clase Provider únicamente tienes que implementar el de la clase `CategorizedProvider`. El resultado de llamar a este método sería:

```
1000|Proveedor-S|Calle-S|1|Silvia|Sanchez|s.sanchez@gmail.com
Category: standard Discount: 10
```

siendo:

  - `1000` el VAT del proveedor, `Proveedor-S` su nombre, `Calle-S` la dirección de facturación o taxAddress,
  - `Silvia Sanchez` nombre y apellidos de la persona de contacto y `s.sanchez@gmail.com` su email
  - `standard` la categoría del proveedor y `10` el descuento aplicable a dicha categoría

Clase PremiumProvider. Declaración de clase y atributos

Declara la clase `PremiumProvider` respetando la jerarquía de la figura y teniendo en cuenta las siguientes especificaciones.

- Cada cliente premium tiene un carnet (`licence`) con un número que se asigna en el momento de su creación. El número de carnet coincide con el número de proveedores premium existentes incluyendo el recién creado (`numPremiumProviders`).
- El número de proveedores premium (`PremiumProviders`) es el mismo para todos los objetos de la clase `PremiumProvider` y se encarga de llevar la cuenta de todos los proveedores premium existentes.

Clase PremiumProvider. Métodos

- Implementa el constructor de la clase `PremiumProvider` que reciba los mismos atributos que la clase `Provider` (NOTA: Observa que no son los mismos que los de la clase `CategorizableProvider`) y asigne valor a todos los atributos (incluyendo el descuento).
- Implementa los métodos `getType` y `getCategory`. (NOTA: no utilices valores literales sino las constantes de la interfaz)
- Puedes asumir que existe el método `getLicence` no es necesario que lo implementes.

ClaseTestProvider

- Declara una clase `TestProviders` y su método main
- Crea un array (`myProvider`) con un proveedor de cada uno de los tres tipos (puedes usar los datos que te damos en el resultado esperado al final del apartado). (NOTA: Puedes asumir para hacerlo que ya existen tres objetos de tipo Person llamados:

```
Person silvia = new Person (1, "Silvia", "Sanchez", "s.sanchez@gmail.com");
Person vicente = new Person (2, "Vicente", "Vazquez", "v.vazquez@gmail.com");
Person pamela = new Person (3, "Pamela", "Perez", "p.perez@gmail.com");
```

)

- Imprime la información de cada proveedor. El resultado esperado sería:

```
1000|Proveedor-S|Calle-S|1|Silvia|Sanchez|s.sanchez@gmail.com
Category: standard Discount: 10
2000|Proveedor-V|Calle-V|2|Vicente|Vazquez|v.vazquez@gmail.com
Category: standard Discount: 20
3000|Proveedor-P|Calle-P|3|Pamela|Perez|p.perez@gmail.com
Category: premium Discount: 30
```

Ejercicio 2 (1,5 / 7 Puntos)

Añade a la clase `TestProvider` el método recursivo:
`sumDiscounts(CategorizedProvider[] providers, int pos)`
que dado un array de proveedores imprima la suma de todos los descuentos que acumulan entre todos. Si lo llamamos sobre el array de proveedores que se describe como resultado esperado en el apartado anterior el resultado esperado sería

```
Total discount: 60
```

Ejercicio 3 (0,5 / 7 Puntos)

Dado el siguiente Test para probar la clase `Person` indica el código que falta en los huecos indicados. Utilizando la nomenclatura vista en clase.

```
        (1)        {
    Person p = new Person(1, "Silvia", "Sanchez", "s.sanchez@gmail.com");
    @Test
    void testGetEmail() {
        assertEquals("s.sanchez@gmail.com",        (2)        );
    }
}
```

SOLUCIÓN DE REFERENCIA (Varias soluciones son posibles)

Ejercicio 1. Solución (5 / 7 puntos)

- **Interfaz Categorizable (1 punto)**
  - Declaración interfaz: 0,25
  - Declaración array de constantes: 0,25
  - Constantes: 0,25
  - Métodos: 0,25

```
public interface Categorizable {
    static final String[] CATEGORIES = { "standard", "vip", "premium" };
    static final int STANDARD = 0;
    static final int VIP = 1;
    static final int PREMIUM = 1;
    String getCategory();
    int getType();
}
```

- **Clase CategorizableProvider. Declaración clase, atributos (0,5 puntos)**
  - 0,3 Declaración de clase
  - 0,2 Declaración de atributos

```
public abstract class CategorizedProvider extends Provider implements Categorizable {
    private int discount;
    private int type;
    //...
```

- **Clase CategorizableProvider. Métodos (1 punto)**
  - 0,5: Constructor
  - 0,25: getters
  - 0,25; toString()

```
public CategorizedProvider(int vat, String name, String taxAddress,
                           Person contactPerson,
                           int type, int discount) {
    super(vat, name, taxAddress, contactPerson);
    this.type = type;
    this.discount = discount;
}
public int getDiscount() {
  return discount;
}
public String getCategory() {
    return CATEGORIES[type];
}

public String toString() {
    return super.toString() + "Category: " + getCategory() +
                              " Discount: " + getDiscount();
}
```

- **Clase PremiumProvider. Declaración de clase y atributos (0,5)**
  - 0,25 declaración de clase
  - 0,25 atributos

```
public class PremiumProvider extends CategorizedProvider {
    private int licence;
    private static int numPremiumProviders;
    //...
```

- **Clase PremiumProvider. Métodos (1 punto)**
  - 0,5 Constructor (0,1 declaración 0,2 super, 0,2 asignacion)
  - 0,5 Métodos (0,25 los que llevan constantes)

```
public PremiumProvider(int vat, String name, String taxAddress, Person contactPerson) {
    super(vat, name, taxAddress, contactPerson, Categorizable.PREMIUM, 30);
    licence = ++numPremiumProviders;
}

public int getType() {
    return Categorizable.PREMIUM;
}
public String getCategory() {
   return Categorizable.CATEGORIES[2];
}
```

- **Clase TestProvider (1 punto)**
  - 0,25: Declaración clase y main
  - 0,25: Creación array
  - 0,25: Creación objetos Provider y pasar adecuadamente los objetos Persona al constructor
  - 0,25: Imprimir correctamente

```
public class TestProvider  {
    public static void main(String[] args) {
        Person silvia = new Person (1, "Silvia", "Sanchez", "s.sanchez@gmail.com");
        Person vicente = new Person (2, "Vicente", "Vazquez", "v.vazquez@gmail.com");
        Person pamela = new Person (3, "Pamela", "Perez", "p.perez@gmail.com");

        CategorizedProvider[] myProviders = {new StandardProvider(1000, "Proveedor-S", "Calle-S", silvia),
                                             new VipProvider(2000, "Proveedor-V", "Calle-V", vicente),
                                             new PremiumProvider(3000, "Proveedor-P", "Calle-P", pamela)};
        for (int i = 0; i < myProviders.length; i++) {
            System.out.println(myProviders[i]);
        }
    }
}
```

Ejercicio 2. Solución (1,5 / 7 puntos)

Rúbrica

- 0,5: condición de parada y caso base
- 0,5: llamada recursiva
- 0,5: operación correcta

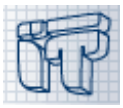
```
public static int sumDiscounts(CategorizedProvider[] providers, int pos) {
    int result = 0;
    if (pos == 0) {
        result = providers[0].getDiscount();
    } else {
        result = providers[pos].getDiscount() + sumDiscounts(providers, pos - 1);
    }
    return result;
}
```

Ejercicio 3. Solución (0,5 / 7 Puntos)

Rúbrica

- 0.5: class PersonTest
- 0,5: p.getEmail()

Solución



```java
import static org.junit.jupiter.api.Assertions.*;

class PersonTest {
    Person p = new Person(1, "Silvia", "Sanchez", "s.sanchez@gmail.com");
    @Test
    void testGetEmail() {
        assertEquals("s.sanchez@gmail.com", p.getEmail());
    }
}
```