PROGRAMACIÓN ORIENTADA A OBJETOS

TRABAJO PRÁCTICO N°3

1. Escriba un ejemplo del uso de herencia simple, detallando para cada clase cuál es su rol dentro de la herencia y los motivos por los cuales cree que es correcta la implementación. El ejemplo debe ser original, se ruega evitar el uso de casos comunes como Animal o Vehículo.

2. Explique el concepto de herencia múltiple y cuál es el problema que se presenta al implementarla.

3. Ejemplifique el uso de interfaces y explique si es mejor tener interfaces mas genericas o mas especificas.

4. Defina la clase abstracta Personaje con los atributos vida, nivelAtaque, nivelDefensa y los métodos atacar() que retorne un integer, y defender(ataque). Implemente el método atacar pero no el método defender(). Luego, cree dos clases hijas donde ambas implementan el método defenderse y al menos una de ellas extiende el método atacar. Cada clase debe establecer una cantidad de puntos de vida por defecto.

   En un ataque se deben realizar cierta cantidad de puntos de daño y en la defensa se debe reducir esa cantidad de puntos de daños. El resultado final de los puntos de ataque debe descontar la misma cantidad de puntos de vida del personaje que defiende.

5. Usando las clases del ejercicio anterior, realizar tres combates entre instancias de ellas. En un combate cada personaje realiza un ataque por turno y es perdedor aquel que queda primero sin vida. En caso de que la vida llegue a un valor negativo el sistema debe lanzar una excepción y en el control de la misma debe dejarse la vida en 0. La decisión de cual personaje ataca primero debe ser aleatoria.

6. Observe y analice el siguiente código:
   Sin ejecutar el código, ¿puede deducir qué se va a imprimir **exactamente** en el foreach de la clase Main? ¿Por qué?.

   Persona:

Python

```python
from abc import ABC, abstractmethod


class Persona(ABC):

    def __init__(self,nombre,apellido):
        self.__nombre = nombre
        self.__apellido = apellido

    def getNombre(self):
        return self.__nombre

    def setNombre(self,nombre):
        self.__nombre = nombre

    def getApellido(self):
        return self.__apellido

    def setApellido(self,apellido):
        self.__apellido = apellido

    @abstractmethod
    def materia():
        pass
```

Java

```java
public abstract class Persona {

    private String nombre;
    private String apellido;

    public Persona (String nombre, String apellido){
        this.nombre = nombre;
        this.apellido = apellido;
    }

    public String getNombre(){
        return this.nombre;
    }

    public void setNombre(String nombre){
        this.nombre = nombre;
    }

    public String getApellido(){
        return this.apellido;
    }

    public void setApellido(String apellido){
        this.apellido = apellido;
    }

    public abstract String materia();
}
```

Alumno:

Python

```python
from Persona import Persona

class Alumnno(Persona):

    def __init__(self, nombre, apellido):
        super().__init__(nombre, apellido)
        self.materias = []

    def get_materias(self):
        return self.materias

    def set_materias(self,materias):
        self.materias = materias

    def agregar_materia(self, materia):
        self.materias.append(materia)

    def materia(self):
        msg = "
El alumno se encuentra cursando las siguientes materias \n"
        for mat in self.materias:
            msg  = '{} {} {}'.format(msg,mat,"\n")

        return msg
```

Java

```java
import java.util.ArrayList;

public class Alumno extends Persona{

    private ArrayList<String> materias;

    public Alumno(String nombre, String apellido){
        super(nombre, apellido);
        this.materias = new ArrayList<>();
    }

    public ArrayList<String> getMaterias(){
        return this.materias;
    }

    public void setMaterias(ArrayList<String> materias){
        this.materias = materias;
    }

    public void agregarMateria(String materia){
        this.materias.add(materia);
    }

    @Override
    public String materia() {
        String msg =
"El alumno se encuentra cursando la siguientes materias \n";
        for (String mat : this.materias) {
            msg += mat+"\n";
        }
        return msg;
    }


}
```

Docente: Python

```python
from Persona import Persona

class Docente(Persona):

    def __init__(self, nombre, apellido, materia):
        super().__init__(nombre, apellido)
        self.__materia = materia

    def get_materia(self):
        return self.__materia

    def set_materia(self, materia):
        self.__materia = materia

    def materia(self):
        return "El docente dicta la siguente clase {} {}".format(self.
__materia,"\n")
```

Java

```java
public class Docente extends Persona{

    private String materia;

    public Docente(String nombre, String apellido, String materia
){
        super(nombre, apellido);
        this.materia = materia;
    }

    public String getMateria(){
        return this.materia;
    }

    public void setMateria(String materia){
        this.materia = materia;
    }

    @Override
    public String materia() {
        return "El docente dicta la siguiente clase "+this.materia
;
    }

}
```

Main:

Python

```python
import random
from Docente import Docente
from Alumno import Alumnno

personas = []
for i in range(10):
    if random.randrange(0, 2) == 1:
        personas.append(Docente("NomDoc {}".format(i), "ApeDoc {}".
format(i), "Mat {}".format(i)))
    else:
        alumno = Alumnno("AlumNom {}".format(i), "AlumApe {}".format(i
))
        for j in range(10):
            alumno.agregar_materia("Mat {}".format(j))
            if random.randrange(0, 2) == 1:
                break
        personas.append(alumno)

for per in personas:
    print(per.materia())
```

Java

```java
import java.util.ArrayList;
import java.util.Random;

public class App {
    public static void main(String[] args) throws Exception {
        Random rnd = new Random();
        ArrayList<Persona> personas = new ArrayList<>();

        for(int i = 0;i<10;i++){
            if(rnd.nextInt(0,2) == 1){
                personas.add(new Docente("NomDoc"+i, "ApeDoc"+i,
"Mat "+i));
            }else{
                Alumno alumno = new Alumno("AluNom"+i, "ApeNom"+i
);
                int limite = rnd.nextInt(10);
                for(int j = 0;j<limite;i++){
                    alumno.agregarMateria("Mat "+j);
                    if(rnd.nextInt(2) == 1){
                        break;
                    }
                }
                personas.add(alumno);
            }
        }

        for(Persona per:personas){
            System.out.println(per.materia());
        }
    }
}
```

7. ¿Qué es la sobrecarga de métodos? Y de parametros?

8. En pos de favorecer la reutilización de código, reescriba el siguiente utilizando herencia.
   ProfesorTitular:
   Python

```python
class ProfesorTitular:

    def __init__(self, nombre, apellido, edad, horasTrabajadas,
antiguedad):
        self.__nombre = nombre
        self.__apellido = apellido
        self.__edad = edad
        self.__horasTrabajadas = horasTrabajadas
        self.__antiguedad = antiguedad
        self.__valorAntiguedad = 1000.00
        self.__valorHora = 300.00


'''ATENCION: desarrolle los getter y setters que crea necesario para su
    funcionamiento'''

    def get_remuneracion_antiguedad(self):
        return self.__valorAntiguedad * self.get_antiguedad()

    def get_remuneracion_mensual(self):
        return (self.__valorHora * self.get_horas_trabajadas()) + self
.get_remuneracion_antiguedad()
```

   Java

```java
public class ProfesorTitular {

    private String nombre;
    private String apellido;
    private Integer edad;
    private Integer horasTrabajadas;
    private Integer antiguedad;
    private Double valorAntiguedad = 1000.00;
    private Double valorHora = 300.00;

    public ProfesorTitular(String nombre, String apellido, Integer
 edad,
    Integer horasTrabajadas, Integer antiguedad){
        this.nombre = nombre;
        this.apellido = apellido;
        this.edad = edad;
        this.horasTrabajadas = horasTrabajadas;
        this.antiguedad = antiguedad;
    }

    /* ATENCION: desarrolle los getter y
    setter que crea necesario para su funcionamiento*/

    public Double get_remuneracion_antiguedad(){
        return this.valorAntiguedad * this.get_antiguedad();
    }

    public Double get_remunercion_mensual(){
        return this.valorHora * get_horas_trabajadas()
        * get_remuneracion_antiguedad();
    }
}
```

ProfesorSuplente:
Python

```python
class ProfesorSuplente:

    def __init__(self, nombre, apellido, edad, horasTrabajadas):
        self.__nombre = nombre
        self.__apellido = apellido
        self.__edad = edad
        self.__horasTrabajadas = horasTrabajadas
        self.__valorHora = 200.00



'''ATENCION: desarrolle los getter y setters que crea necesario para su
    funcionamiento'''

    def get_remuneracion_mensual(self):
        return self.__valorHora * self.get_horas_trabajadas()
```

Java

```java
public class ProfesorSuplente {

    private String nombre;
    private String apellido;
    private Integer edad;
    private Integer horasTrabajadas;
    private Double valorHora = 200.00;

    public ProfesorSuplente(String nombre, String apellido,
    Integer edad, Integer horasTrabajadas){
        this.nombre = nombre;
        this.apellido = apellido;
        this.edad = edad;
        this.horasTrabajadas = horasTrabajadas;
    }

    /* ATENCION: desarrolle los getter y
    setter que crea necesario para su funcionamiento*/

    public Double get_remunercion_mensual(){
        return this.valorHora * this.get_horas_trabajadas();
    }


}
```

Python

```python
from ProfesorTitular import ProfesorTitular
from ProfesorSuplente import ProfesorSuplente

titulares = []
titulares.append(ProfesorTitular("Aldana", "Gutierrez", 40, 40, 15))
titulares.append(ProfesorTitular("Pedro", "Perez", 30, 30, 4))
titulares.append(ProfesorTitular("María", "Gomez", 29, 40, 1))

suplentes = []
suplentes.append(ProfesorSuplente("Tomas", "Sosa", 28, 40))
suplentes.append(ProfesorSuplente("Luciana", "Torres", 35, 10))

for titular in titulares:
    print("Nombre y Apellido {},{}".format(titular.get_nombre(),
titular.get_apellido()))
    print("Es titular? Si")
    print("Remuneracion: {}".format(titular.get_remuneracion_mensual
()))


for suplente in suplentes:
    print("Nombre y Apellido {},{}".format(suplente.get_nombre(),
suplente.get_apellido()))
    print("Es titular? No")
    print("Remuneracion: {}".format(suplente.get_remuneracion_mensual
()))
```

Java

```java
import java.util.ArrayList;

public class App {
    public static void main(String[] args) throws Exception {

        ArrayList<ProfesorTitular> titulares = new ArrayList<>();
        ArrayList<ProfesorSuplente> suplentes = new ArrayList<>();

        titulares.add(new ProfesorTitular("Aldana", "Gutierrez",
40, 40, 15));
        titulares.add(new ProfesorTitular("Pedro", "Perez", 30, 30
, 4));
        titulares.add(new ProfesorTitular("Maria", "Gomez", 29, 40
, 1));
        suplentes.add(new ProfesorSuplente("Tomas", "Sosa", 28, 40
));
        suplentes.add(new ProfesorSuplente("Luciana", "Torres", 35
, 10));

        for(ProfesorTitular prof : titulares){
            System.out.println("Nombre y Apellido "+prof.
get_nombre()+", "+prof.get_apellido());
            System.out.println("Es titular? Si");
            System.out.println("Remuneracion: "+prof.
get_remunercion_mensual());
        }

        for(ProfesorSuplente prof : suplentes){
            System.out.println("Nombre y Apellido "+prof.
get_nombre()+", "+prof.get_apellido());
            System.out.println("Es titular? No");
            System.out.println("Remuneracion: "+prof.
get_remunercion_mensual());
        }

    }
}
```

9. Defina la clase TarifaProveedor con un método calcular(totalSMS, totalMinutos, totalGigas) que, dado la cantidad de mensajes, minutos de llamada y GB de consumo de datos calcule el valor en pesos a pagar. El valor a retornar del método calcular debe ser la suma de los resultados obtenidos en los métodos calcularSMS(totalSMS), calcularMinutosDeLlamada(totalMinutos) y calcularConsumoGB(totalGigas)

   Los valores por defecto de cada servicio son
   - Mensajes de texto(SMS): $1
   - Minuto de llamada: $15
   - Gigas(GB) de internet: $20.

   Además de los métodos anteriores, debe poseer un método abstracto getNombre() que retorne el nombre del proveedor.

   Luego, defina una clase hija por cada uno de los siguientes proveedores:
   - Claro: que tiene un 20% extra sobre el básico total
   - Personal: que tiene un 20% extra sobre los minutos de llamada y 50% sobre los GB de datos.
   - Movistar: tiene un 10% extra sobre los mensajes de texto, 20% sobre las llamadas y 30% sobre los GB de datos.

   Desarrolle una interfaz gráfica que permita ingresar la cantidad de SMS, minutos de llamada, Gigas y muestre como resultado la tarifa que se obtendría con cada proveedor.

10. Desarrollé parte del sistema de control de horarios para la universidad.

    Para todo el personal que trabaja en la universidad se desea tener el nombre completo, antigüedad en años y sector. Además, es necesario que para cada uno sea posible almacenar y obtener la cantidad de horas trabajadas en el mes.

    El personal docente tiene asociada una categoría que define la cantidad de horas semanales a trabajar:
    - Simple: 10 hs semanales
    - Semiexclusiva: 20 hs semanales
    - Exclusiva: 40 hs semanales

El personal no docente puede tener dos tipos de jornadas asignadas:

- Completa: 30 hs semanales
- Reducida: 20 hs semanales

Defina la clase Reloj que sepa generar un informe. Este debe indicar para cada persona la cantidad de horas trabajadas en el mes indicando además si cumplió o no con el mínimo esperado.

Para poner a prueba el sistema debe realizarse un simulación con los siguientes criterios:

- Al menos diez personas entre docentes y no docentes
- Para los docentes debe definirse aleatoriamente la categoría
- Para los no docentes debe definirse aleatoriamente la jornada
- La cantidad de horas trabajadas en el mes debe ser generada aleatoriamente bajo las siguientes pautas:
    - Para docentes con categoría simple hay un 95% de posibilidad de que exceda la cantidad de horas requeridas.
    - Para docentes con categoría semiexclusiva hay un 75% de posibilidad de que exceda la cantidad de horas requeridas.
    - Para docentes con categoría exclusiva hay un 60% de posibilidad de que exceda la cantidad de horas requeridas.
    - Para no docentes hay un 80% de posibilidad de que exceda la cantidad de horas requeridas.
- Debe solicitarse a una instancia de la clase Reloj que imprima el informe.