Trabajo de Programación

1 Introducción Teórica:

-Polimorfismo

Es la capacidad que tienen los objetos de una clase en ofrecer respuesta distinta e independiente en función de los parámetros (diferentes implementaciones) utilizados durante su invocación.

-Ejemplo

```
class Animal {
  public void makeSound() {
    System.out.println("Grr...");
  }
}
class Cat extends Animal {
  public void makeSound() {
    System.out.println("Meow");
  }
}
class Dog extends Animal {
  public void makeSound() {
    System.out.println("Woof");
  }
}
```

Como todos los objetos Gato y Perro son objetos Animales

```
public static void main(String[ ] args) {
  Animal a = new Dog();
  Animal b = new Cat();
}
```

Creamos dos variables de referencia de tipo Animal y las apuntamos a los objetos Gato y Perro. Ahora, podemos llamar a los métodos makeSound()

```
a.makeSound();
//Outputs "Woof"

b.makeSound();
//Outputs "Meow"
```

-Polimorfismo paramétrico

Existen funciones con el mismo nombre pero se usan diferentes parámetros (nombre o tipo). Selecciona el método dependiendo del tipo de datos que se envíe.

-Ejemplo

```
class Overload
{
    void demo (int a)
    {
       System.out.println ("a: " + a);
    }
    void demo (int a, int b)
    {
       System.out.println ("a and b: " + a + "," + b);
    }
    double demo(double a) {
       System.out.println("double a: " + a);
       return a*a;
    }
}
class MethodOverloading
{
    public static void main (String args [])
    {
        Overload Obj = new Overload();
        double result;
        Obj .demo(10);
        Obj .demo(10, 20);
        result = Obj .demo(5.5);
        System.out.println("O/P : " + result);
    }
}
```

Salida de datos:

```
a: 10
a and b: 10,20
double a: 5.5
O/P : 30.25
```

-Parámetros de inclusión

Es cuando se puede llamar a un método sin tener que conocer su tipo, así no se toma en cuanta los detalles de las clases especializadas, utilizando una interfaz común

-Ejemplo

```
abstract class Piece{
    public abstract void move(byte X, byte Y);
}

class Bishop extends Piece{
 @Override
  public void move(byte X, byte Y){

  }
}
```

-Herencia

Herencia es un concepto de la programación orientada a objetos. El cual es un mecanismo que permite derivar una clase a otra clase. En otras palabras, tendremos unas clases que serán hijos, y otras clases que serán padres

-Ejemplo

```
//Clase para objetos
de dos dimensiones
class DosDimensiones{
    double base;
    double altura;

    void mostrarDimension(){
        System.out.println("La
        base y altura es: "+base+"
        y "+altura);
    }
}
```

```
//Una subclase de DosDimensiones
para Triangulo
class Triangulo
extends DosDimensiones{
 String estilo;

    double area(){
     return base*altura/2;
    }

    void mostrarEstilo(){
     System.out.println
   ("Triangulo es: "+estilo);
  }
}

class Lados3{
  public static void
  main(String[] args) {
   Triangulo t1=new Triangulo();
   Triangulo t2=new Triangulo();

   t1.base=4.0;
   t1.altura=4.0;
   t1.estilo="Estilo 1";

   t2.base=8.0;
   t2.altura=12.0;
   t2.estilo="Estilo 2";

    System.out.println("Información
   para T1: ");
    t1.mostrarEstilo();
    t1.mostrarDimension();
    System.out.println("Su área es:
     "+t1.area());

    System.out.println();

    System.out.println("Información
   para T2: ");
    t2.mostrarEstilo();
    t2.mostrarDimension();
    System.out.println("Su área es:
     "+t2.area());

    }
}
```

-Sobre carga de métodos

La sobrecarga de métodos es la creación de varios métodos con el mismo nombre pero con diferente lista de tipos de parámetros. Java utiliza el número y tipo de parámetros para seleccionar cuál definición de método ejecutar

Ejemplo

```
/* Métodos sobrecargados */
int calculaSuma(int x, int y, int z){
    ...
}
int calculaSuma(double x, double y, double z){
    ...
}
```

2 Análisis Comparativo:

-Explicar las diferencias entre polimorfismo y sobrecarga de métodos

El polimorfismo a través de interfaces permite añadir nuevos tipos sin cambiar las clases existentes. La sobrecarga de métodos se refiere a tener varis implementaciones con el mismo nombre pero diferentes parámetros en una clase.

-Diferenciar entre sobrecarga (overloading) y redefinición (overriding) de métodos

La sobrecarga (overloading) implica tener varios métodos con el mismo nombre pero diferentes parámetros dentro de una misma clase, mientras que la redefinición (overriding) ocurre cuando una subclase proporciona una implementación específica de un método de su superclase con la misma firma

-Preguntas

-¿Qué es el termino firma?

La firma de un método en Java consiste en su nombre junto con la lista de parámetros que recibe. Cuando se sobrecargan métodos, es decir , se tienen varios métodos con el mismo nombre pero diferentes parámetros, cada uno tiene una firma única. Esto permite al compilador distinguir entre ellos y seleccionar el método adecuado según los argumentos utilizados en la llamada.

-¿Diferencias entre los términos Overloading y Overriding?

Overloading se refiere a tener varios métodos con el mismo nombre pero diferentes firmas dentro de una misma clase, mientras que Overriding implica redefinir un método en una subclase con la misma firma que el método en la superclase. Overloading se asocia con polimorfismo en tiempo de compilación, ya que se determina durante la compilación, mientras que Overriding se relaciona con

polimorfismo en tiempo de ejecución, ya que la decisión sobre qué método invocar se toma durante la ejecución del programa.

-¿Se pueden sobrecargar métodos estáticos?

Sí, es posible tener dos más métodos estáticos con el mismo nombre siempre que se diferencien en los parámetros de entrada

-¿Es posible sobrecargar la clase main() en Java?

Sí, es posible siempre que definamos correctamente los parámetros de entrada.