Fecha : Agosto 23 de 2018
Grupo: 800
Nombre: Camilo Arias González

1. **¿Cómo definir y declarar los siguientes elementos?**

   a. **Interfaz**

   Es un contrato compuesto por métodos y propiedades declarado con la palabra clave interface, todas aquellas clases que implementan la interface deben obligatoriamente cumplimentar los métodos expuestos en ella.

   Ejemplo:

   Interface

```
public interface Cantante  {
    public void cantar();
 }
```

   Implementación

```
public class Canario implements Cantante  {
        public void cantar()          {
            System.out.println("Pio Pio Pio");
        }
  }
```

   b. Métodos

   Son aquellas funciones que permite efectuar el objeto y que nos rinden algún tipo de servicio durante el transcurso del programa.
   Determinan a su vez como va a responder el objeto cuando recibe un mensaje.

   Ejemplo

   Método

```
public int obtenerValor(int parametro){
        int x=parametro/2
        return x;
}
```

   Instancia

```
        obtenerValor(2);
```

   c. **Propiedades**

   Una propiedad es un miembro que proporciona un mecanismo flexible para leer, escribir o calcular el valor de un campo privado. Las propiedades se pueden usar como si fueran miembros de datos públicos, pero en realidad son métodos especiales denominados descriptores de acceso. Esto permite acceder fácilmente a los datos a la vez que proporciona la seguridad y la flexibilidad de los métodos.

   Ejemplo

```
Private double seconds;
Public double minutes;
Protected double minutes;
```

   d. **Eventos**

La palabra event clave se usa para declarar un evento en una clase de editor.

```
namespace ImplementInterfaceEvents {
    public interface IDrawingObject {
        event EventHandler ShapeChanged;
        }
        public class MyEventArgs : EventArgs {
        // class members
        }
        public class Shape : IDrawingObject {
        public event EventHandler ShapeChanged; void
    ChangeShape() {
        // Do something here before the event…
        OnShapeChanged(new MyEventArgs(/*arguments*/));
        // or do something here after the event.
        }
        protected virtual void OnShapeChanged(MyEventArgs e) {
        ShapeChanged?.Invoke(this, e);
        }
    }
}
```

2. ¿Como derivar a partir de una interfaz base?

Existe la posibilidad de derivar después de implementar de una interfaz, así mismo se puede heredar en la misma clase de implementación si afectar los métodos de la interfaz padre.

```
public interface Parlanchin {
   public abstract void habla();
}

public abstract class Animal implements Parlanchin{
        public abstract void habla();
}

class Perro extends Animal{
        public void habla(){
            System.out.println("¡Guau!");
          }
}
class Gato extends Animal{
          public void habla(){
            System.out.println("¡Miau!");
   }
}
```

3.¿Como utilizar la palabra New para reutilizar identificadores?

Se pueden reutilizar identificadores para instanciar objetos nuevos con la palabra reserv ada “New” del sistema.

```
public class PoliApp {
        public static void main(String[] args) {
            Gato gato=new Gato();
            hazleHablar(gato);
              Cucu cucu=new gato();
              hazleHablar(cucu);
      }
        static void hazleHablar(Parlanchin sujeto){
```

```
            sujeto.habla();
        }
}
```

4. ¿Como implementar interfaz en clases y estructura?

```
public interface IntegranteSeleccionFutbol {
        void concentrarse();
        void viajar();
        void entrenar()
        void jugarPartido();}

public abstract class SeleccionFutbol implements
IntegranteSeleccionFutbol {
        protected int id;
        protected String nombre;
        protected String apellidos;
        protected int edad;
public SeleccionFutbol() {}
public SeleccionFutbol(int id, String nombre, String apellidos,
int edad) {
        this.id = id;
        this.nombre = nombre;
        this.apellidos = apellidos;
        this.edad = edad;
        }
}
```

5. ¿Cómo implementar métodos de interfaz con el mismo nombre?

Se puede implementar métodos con el mismo nombre usando la palabra reservada del sistema "@override" dependiendo del lenguaje de implementación.

```
interface A{
          int f();
}
interface B{
        int f();
}

class Test implements A, B{
        public static void main(String... args) throws
Exception{
        }
          @Override  public int f() {
         // from which interface A or B    return 0;
          }
}
```