# Problema I (25 puntos)

A continuación se muestra una manera desordenada y poco eficiente de crear usuarios.

```
//Tratemos de crear usuarios de la siguiente manera

//usuario0
let nombre = 'Paola';
let apellido = 'Ortiz';
let email = 'paola@company.ru'
let direccion = {
    municipio: 'Jocotenango',
    calle: 'Calle ancha',
    numero: 25,
};

//usuario1
let nombre1 = 'Paolo';
let apellido1 = 'Ortega';
let email1 = 'paolo@company.ru'
let direccion1 = {
    municipio: 'Jocotenango',
    calle: 'Calle ancha',
    numero: 25,
};

//usuario2...
```

Además de estos usuarios, crea al usuario2, al usuario3, al usuario4, al usuario5, con las mismas características que se muestran en la figura inmediatamente anterior. ¿Puedes mostrar estas características de los usuarios de manera ordenada y coherente en consola? ¿Tienen los usuarios alguna funcionalidad, pueden hacer algo?

## Solucion:

¿Puedes mostrar estas características de los usuarios de manera ordenada y coherente en consola?
**Si es posible, hay diferentes maneras de hacerlo, por ejemplo 1 es convertir todo a un array y luego mostrar ese array, también puede ser con una iteración, sin embargo esto es poco eficiente. Aquí hay un ejemplo de cómo hacerlo:**

```
// Mostrar datos en consola
for (let i = 1; i <= 5; i++) {
    console.log(`Persona ${i}:`);
    console.log(`Nombre: ${eval('nombre' + i)}`);
    console.log(`Apellido: ${eval('apellido' + i)}`);
    console.log(`Email: ${eval('email' + i)}`);
    console.log('Dirección:', eval('direccion' + i));
    console.log('--------------------');
}
```

¿Tienen los usuarios alguna funcionalidad, pueden hacer algo?

**Si, pero no es eficiente, la idea es poder guardar la información de los usuarios de tal manera que podamos manipular esa información como un objeto en lugar de hacerlo individualmente, básicamente agrupándolos en un objeto.**

---

## Problema II (25 puntos)

Crea otro código utilizando las características de los usuarios del Problema I, pero en esta ocasión agrega una funcionalidad, método recuperarClave, de la manera que se observa en la siguiente figura.

```
let user = {                    //objeto user
    nombre: 'Paola',
    apellido: 'Ortiz',
    email: 'paola@company.ru',
    direccion: {                 // objeto direccion dentro del ojeto
        municipio: 'Jocotenango',
        calle: 'Calle ancha',
        numero: 25,
    },
    estado: true,               //podemos asignar valores boolean
    recuperarClave: function () { //podemos asignar funciones, en este
        console.log('Recuperar clave...')
    }
};
```

Despliega en consola a los 6 usuarios creados, observa que ahora si hay un orden y existe coherencia entre las propiedades y métodos. El resultado será la creación de 6 usuarios, pero en esta ocasión diremos que son 6 objetos. ¿Qué diferencias conceptuales observas entre el

Problema I y el Problema II? ¿Para crear usuarios es más fácil y coherente la manera del Problema I o la manera en que se crean en el Problema II?

### Solucion:

**Las diferencias conceptuales  son que el problema 1 utiliza variables individuales para cada propiedad del usuario, en el problema 2 se encapsulan dentro de un solo objeto lo que facilita la organización y manipulación de los datos. Para la creación de usuarios es más fácil y coherente la manera del problema 2 ya que hay una mejora en la organización y manipulación de datos.**

**Para mostrar los datos de forma más ordenada los coloque en un array "usuarios" y luego realice una iteración para mostrar cada uno de ellos.**

## Problema III (25 puntos)

Crea otro código utilizando el Problema II, pero ahora agrega a cada usuario (objeto) la propiedad dpi y el método cambiarDireccion. Utiliza el ejemplo 05-Objetos/02-dinamico.js como guía para solucionar este problema. Muestra a cada usuario en la consola.

### Solucion:

**Para este problema, he creado 2 funciones, la primera(agregarDpi) al llamarla agrega todos los DPI's a cada uno de los usuarios. La segunda(cambiarDireccion) cambia los valores de los objetos dentro del objeto los cuales son dirección (municipio, calle y número)**

**Después de esto, muestro los valores por medio de la función mostrarLog.**

---

## Problema IV (25 puntos)

Utilizando las propiedades y métodos agrupados en cada usuario creado en el Problema III, vuelve a crear esos usuarios (objetos) pero en esta ocasión crea los usuarios (objetos) utilizando una factory function. Puedes guiarte en el ejemplo 05-Objetos/03-factoryFunction.js. Muestra a los usuarios en la consola.

Solucion:

**Para este problema utilice una factory function que en JavaScript es una función que devuelve un nuevo objeto cada vez que se llama, sin usar la palabra clave new. Es una forma más práctica y ordenada de crear objetos y encapsular la lógica de creación, permitiendo personalizar y configurar objetos de manera flexible. La función "crearPersona" la factory function es la encargada de crear los objetos, luego agregamos datos asignando una variable a cada uno de los objetos, y luego desplegamos en consola.**