Pregunta1:

NO LSP

Se implemento como lo establecido, con un constuctor y dos método y una lista de miembros.

```java
package Pregunta1.noLSP;
import java.util.List;

3 inheritors
public abstract class Member {
    private final String nombre;

    public Member(String nombre) {
        this.nombre = nombre;
    }
    3 implementations
    public abstract void joinTournament();
    3 implementations
    public abstract void organizeTournament();

    List<Member> miembros = List.of(
            new PremiumMember( nombre: "Abejita Azul"),
            new VipMember( nombre: "Kaperucita Feliz"),
            new FreeMember( nombre: "Inspectora Motita")
    );

}
```

La clase FreeMember hereda de la clase Member , pero esta no puede tener el método sobrescrito organizeTournament (aquí es donde refactorizaremos)

```java
package Pregunta1.noLSP;

public class FreeMember extends Member {
    public FreeMember(String nombre){
        super(nombre);
    }
    @Override
    public void joinTournament() {
        System.out.println("...");
    }
    /**
     * Este metodo no deberia existir aqui, porque el es MiembroLibre,
     * no puede organizar
     */
    @Override
    public void organizeTournament() {
        System.out.println();
    }


}
```

La clase PremiumMember y VipMember , ambos heredan del la clase Member y en este caso no hay problema con que use ambos métodos heredados.

```java
package Pregunta1.noLSP;

public class PremiumMember extends Member {
    public PremiumMember(String nombre){
        super(nombre);
    }
    @Override
    public void joinTournament() {
        System.out.println("");
    }

    @Override
    public void organizeTournament() {

    }

}
```

```java
package Pregunta1.noLSP;

public class VipMember extends Member {
    public VipMember(String nombre){
        super(nombre);
    }

    @Override
    public void joinTournament() {

    }
    @Override
    public void organizeTournament() {

    }

}
```

Aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa

CON LSP:

Cree dos Interfaces, asi cuando una clase lo necesite solo lo implemente.

```java
package Pregunta1.LSP;

4 implementations
public interface joinTournament {
    3 implementations
    void joinTournament();
}
```

```java
package Pregunta1.LSP;

3 implementations
public interface organizeTournament {
    2 implementations
    void organizeTournament();
}
```

Ahora mi clase abstract Member lo mantenemos solo agregamos un getNombre.

```java
package Pregunta1.LSP;

3 inheritors
public abstract class Member {
    private final String nombre;

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

    PremiumMember premiun = new PremiumMember( nombre: "Abejita Azul");
    VipMember vip = new VipMember( nombre: "Kaperucita Feliz");
    FreeMember free = new FreeMember( nombre: "Inspectora Motita");
    }
```

Ahora en mi Clase FREEMENBER, ahí si cambiamos, heredara de Member pero implementara solo la interfaz joinTournament, ahora si el no podrá organizar.

```java
package Pregunta1.LSP;

public class FreeMember extends Member implements joinTournament{
    String nombre;
    public FreeMember(String nombre){
        super(nombre);
    }
    @Override
    public void joinTournament() {
        System.out.println(".."+nombre);
    }

    /**
     * Este metodo no deberia existir aqui, porque el es MiembroLibre,
     * para eso creamos una interfaz donde estaran los metodos cuando pueden
     * organizar o unirse.
     */
    /*
    @Override
    public void organizeTournament() {
        System.out.println();
    }
     */

}
```

La clase VipMember y PremiunMenber heredan de Member y ambos implementaran joinTournament, organizeTournamet.

```java
package Pregunta1.LSP;

public class PremiumMember extends  Member implements joinTournament,organizeTournament{
    String nombre;

    public PremiumMember(String nombre) {
        super(nombre);
    }
    @Override
    public void joinTournament(){
        System.out.println("Uniendose"+nombre);
    }
    @Override
    public void organizeTournament() {
        System.out.println("Organizando "+nombre);
    }
}
```

```java
package Pregunta1.LSP;

public class VipMember extends Member implements joinTournament,organizeTournament {
    String nombre;
    public VipMember(String nombre){
        super(nombre);
    }

    @Override
    public void joinTournament() {
        System.out.println("Uniendose"+nombre);
    }
    @Override
    public void organizeTournament() {
        System.out.println("Organizando "+nombre);
    }

}
```