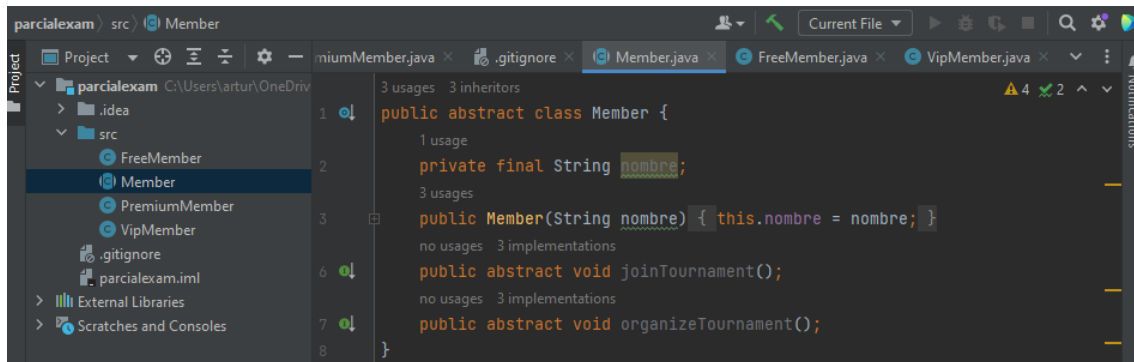
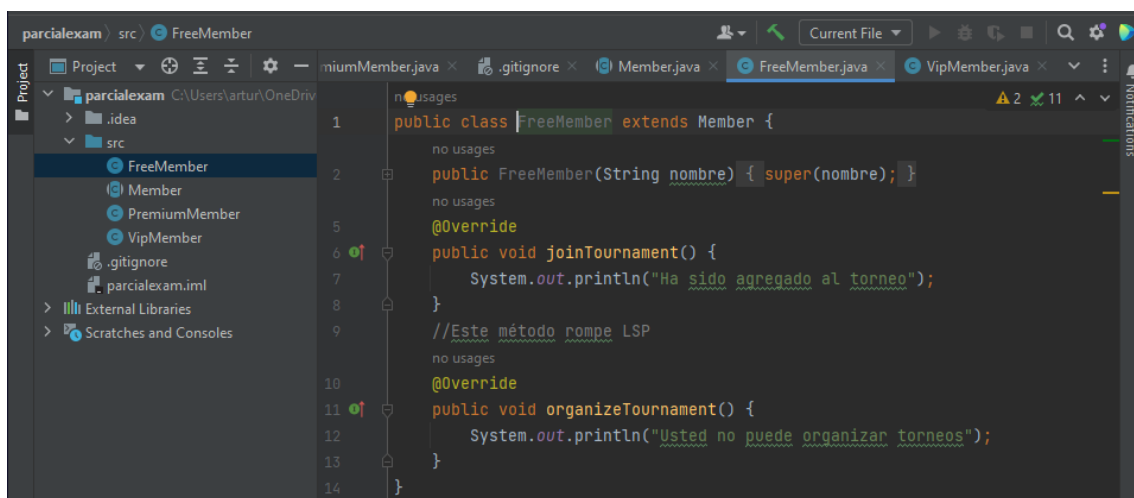


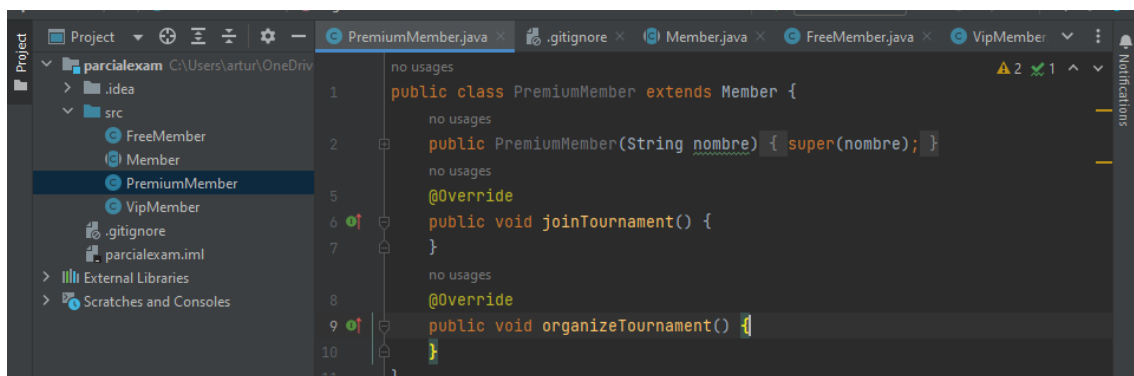
Inicialmente el programa arremetía contra el principio LSP (Liskov Substitution Principle) ya que la clase FreeMember no tenía la responsabilidad de organizar torneos y la forma de escribir solamente una impresión de pantalla no es suficiente. Solución: Refactorización.



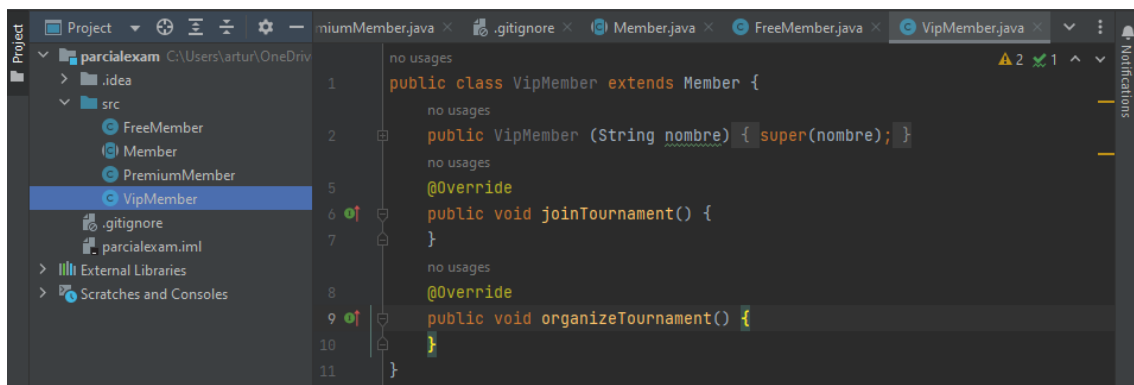
```
1 public abstract class Member {
2     private final String nombre;
3     public Member(String nombre) { this.nombre = nombre; }
4     public abstract void joinTournament();
5     public abstract void organizeTournament();
6 }
```



```
1 public class FreeMember extends Member {
2     public FreeMember(String nombre) { super(nombre); }
3     @Override
4     public void joinTournament() {
5         System.out.println("Ha sido agregado al torneo");
6     }
7     //Este método rompe LSP
8     @Override
9     public void organizeTournament() {
10        System.out.println("Usted no puede organizar torneos");
11    }
12 }
```

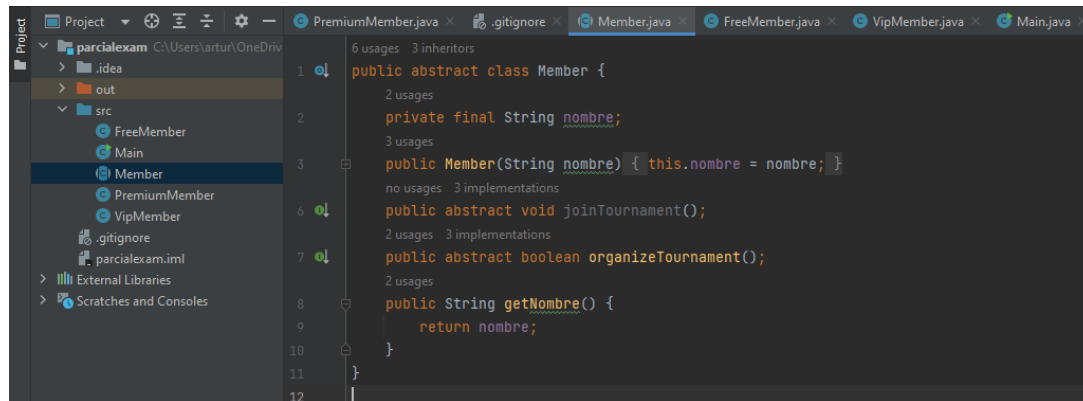


```
1 public class PremiumMember extends Member {
2     public PremiumMember(String nombre) { super(nombre); }
3     @Override
4     public void joinTournament() {
5     }
6     @Override
7     public void organizeTournament() {
8     }
9 }
```

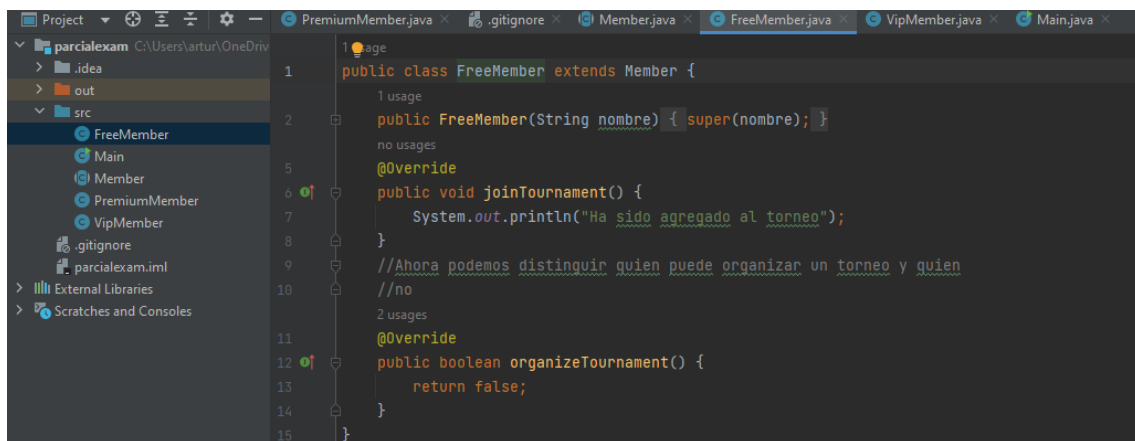


```
1 public class VipMember extends Member {
2     public VipMember (String nombre) { super(nombre); }
3     @Override
4     public void joinTournament() {
5     }
6     @Override
7     public void organizeTournament() {
8     }
9 }
```

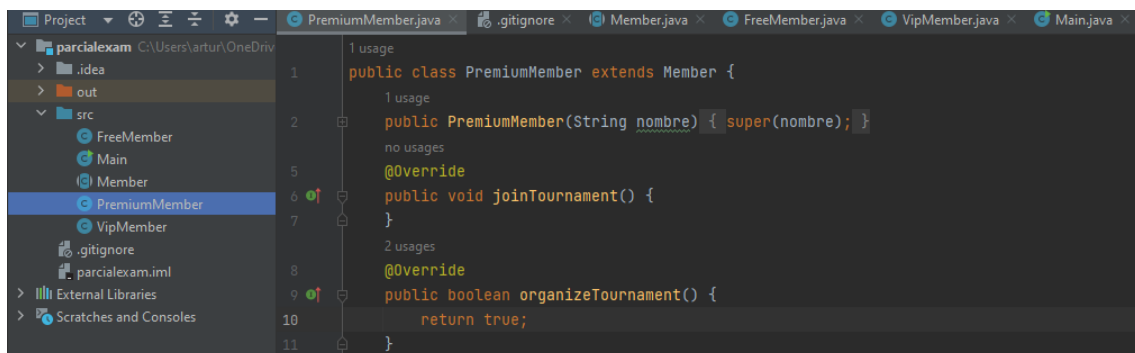
Posteriormente, ahora con la refactorización se puede distinguir quienes pueden organizar un torneo y quienes no, ya que al poner como método un boolean podemos identificar con true o false que objeto que recibe la herencia de Member, tiene la capacidad de realizar dicha acción. Se agrega getNombre() para poder utilizarlo en nuestra nueva clase main.



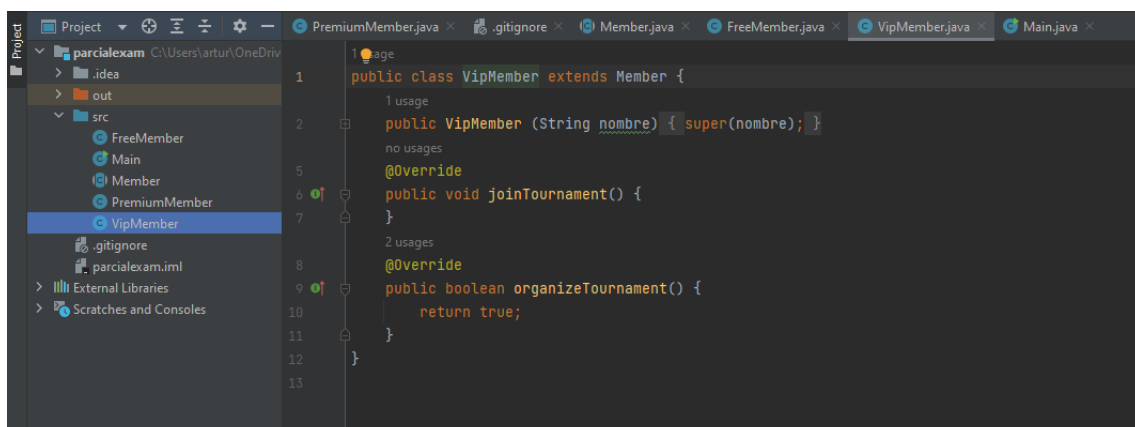
```
1 public abstract class Member {
2     private final String nombre;
3     public Member(String nombre) { this.nombre = nombre; }
4     public abstract void joinTournament();
5     public abstract boolean organizeTournament();
6     public String getNombre() {
7         return nombre;
8     }
9 }
```



```
1 public class FreeMember extends Member {
2     public FreeMember(String nombre) { super(nombre); }
3     @Override
4     public void joinTournament() {
5         System.out.println("Ha sido agregado al torneo");
6     }
7     //Ahora podemos distinguir quien puede organizar un torneo y quien
8     //no
9     @Override
10    public boolean organizeTournament() {
11        return false;
12    }
13 }
```

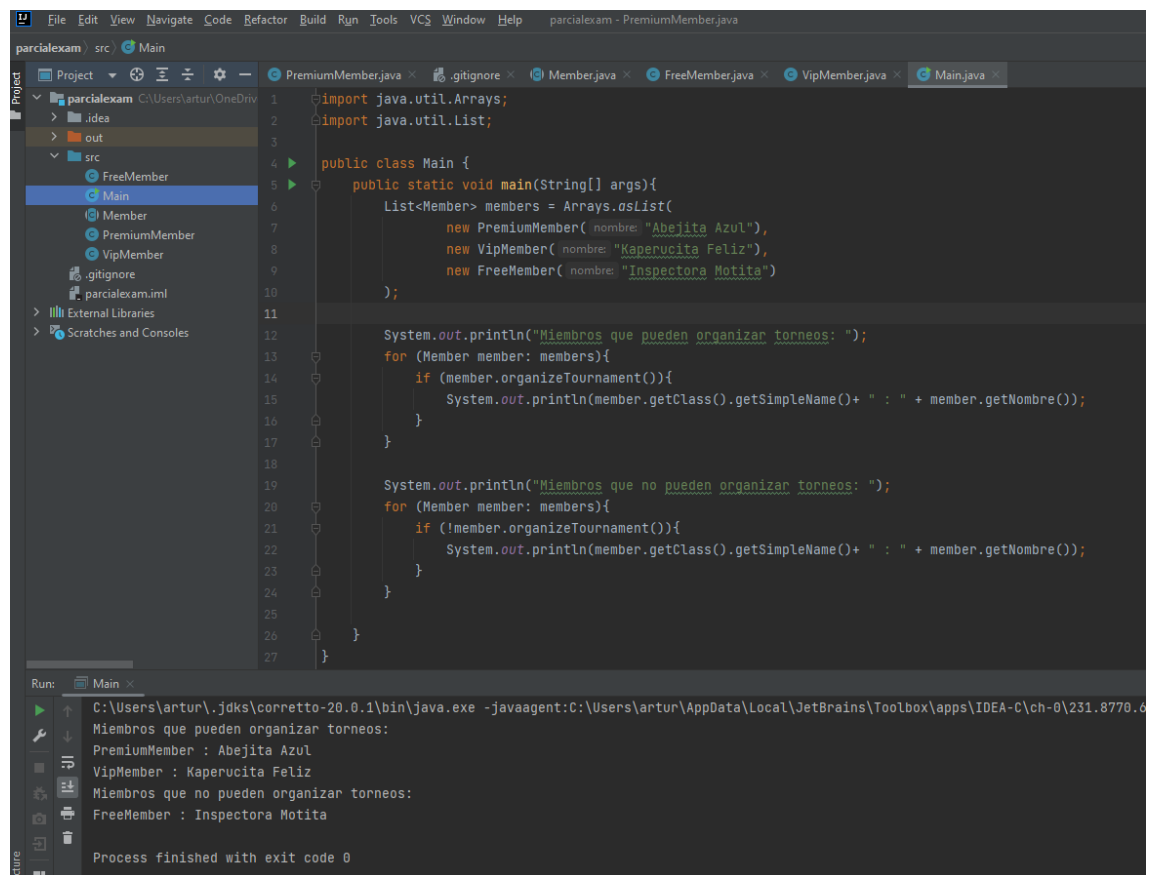


```
1 public class PremiumMember extends Member {
2     public PremiumMember(String nombre) { super(nombre); }
3     @Override
4     public void joinTournament() {
5     }
6     @Override
7     public boolean organizeTournament() {
8         return true;
9     }
10 }
```



```
1 public class VipMember extends Member {
2     public VipMember (String nombre) { super(nombre); }
3     @Override
4     public void joinTournament() {
5     }
6     @Override
7     public boolean organizeTournament() {
8         return true;
9     }
10 }
```

Para la función Main se tiene



The screenshot shows an IDE window titled "parcialeexam - PremiumMember.java". The project structure on the left includes "parcialeexam" with subfolders "out" and "src". The "src" folder contains "FreeMember", "Main", "Member", "PremiumMember", and "VipMember". The "Main.java" file is open, showing the following code:

```
1 import java.util.Arrays;
2 import java.util.List;
3
4 public class Main {
5     public static void main(String[] args){
6         List<Member> members = Arrays.asList(
7             new PremiumMember( nombre: "Abejita Azul"),
8             new VipMember( nombre: "Kaperucita Feliz"),
9             new FreeMember( nombre: "Inspectora Motita")
10        );
11
12        System.out.println("Miembros que pueden organizar torneos: ");
13        for (Member member: members){
14            if (member.organizeTournament()){
15                System.out.println(member.getClass().getSimpleName()+ " : " + member.getNombre());
16            }
17        }
18
19        System.out.println("Miembros que no pueden organizar torneos: ");
20        for (Member member: members){
21            if (!member.organizeTournament()){
22                System.out.println(member.getClass().getSimpleName()+ " : " + member.getNombre());
23            }
24        }
25    }
26 }
27 }
```

The Run console at the bottom shows the output of the program:

```
Run: Main x
C:\Users\artur\.jdk\corretto-20.0.1\bin\java.exe -javaagent:C:\Users\artur\AppData\Local\JetBrains\Toolbox\apps\IDEA-C\ch-0\231.8770.6
Miembros que pueden organizar torneos:
PremiumMember : Abejita Azul
VipMember : Kaperucita Feliz
Miembros que no pueden organizar torneos:
FreeMember : Inspectora Motita
Process finished with exit code 0
```