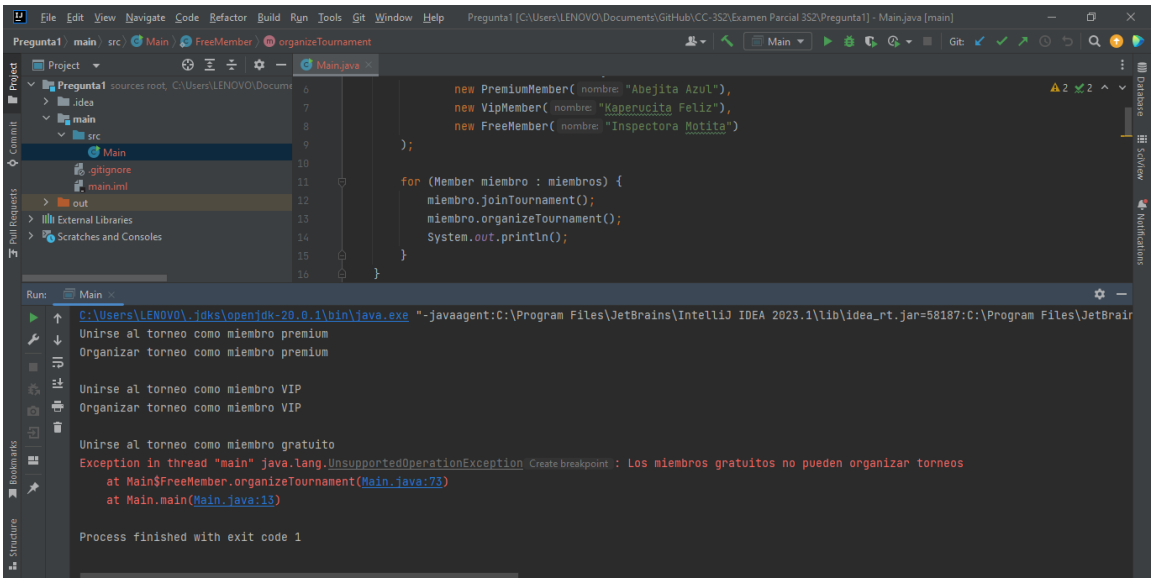
Alumna: Zuñiga Chicaña, Alejandra Aztirma **Código: 20202142E**

Pregunta 1

Al consultar la siguiente lista de miembros:

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

Nos muestra una excepción debido a que como se dijo en el texto, es decir no se efectua la compatibilidad con LSP puesto que un free member no puede organizar torneos, la siguiente captura nos muestra esto una vez ejecutado nuestro código



Ahora, lo que el problema nos pide es:

Rediseña la solución para obtener un código compatible con LSP a través de un proceso de refactorización.

En este caso para que la refactorización “sea visible” se procedió a crear otra clase llamada ChessClub, en la cual se encontrará la refactorización de los métodos para hacer posible que nuestro código sea compatible con LSP.

En nuestro código original se creó la clase main.

```java
public class Main {
    public static void main(String[] args) {
        List<Member> miembros = List.of(
                new PremiumMember( nombre: "Abejita Azul"),
                new VipMember( nombre: "Kaperucita Feliz"),
                new FreeMember( nombre: "Inspectora Motita")
        );

        for (Member miembro : miembros) {
            miembro.joinTournament();
            miembro.organizeTournament();
            System.out.println();
        }
    }
```

Aquí se observa la clase ChessClub, la cual itera sobre una lista de miembros y muestra si pueden unirse a un torneo y, si son TournamentParticipant, también intenta organizar un torneo. Si un miembro no implementa TournamentParticipant, se muestra un mensaje indicando que no pueden organizar torneos.

```java
public class ChessClub {
    public static void main(String[] args) {
        List<Member> miembros = List.of(
                new PremiumMember( nombre: "Abejita Azul"),
                new VipMember( nombre: "Kaperucita Feliz"),
                new FreeMember( nombre: "Inspectora Motita")
        );

        for (Member miembro : miembros) {
            miembro.joinTournament();
            if (miembro instanceof TournamentParticipant) {
                TournamentParticipant participant = (TournamentParticipant) miembro;
                participant.organizeTournament();
            } else {
                System.out.println("Este miembro no puede organizar torneos");
            }
            System.out.println();
        }
    }
```

Una vez modificado el código, este nos da una salida como se muestra a continuación:

```
Debugger  Console
Unirse al torneo como miembro VIP
Organizar torneo como miembro VIP

Unirse al torneo como miembro gratuito
Este miembro no puede organizar torneos

Disconnected from the target VM, address: '127.0.0.1:58424', transport: 'socket'

Process finished with exit code 0
```