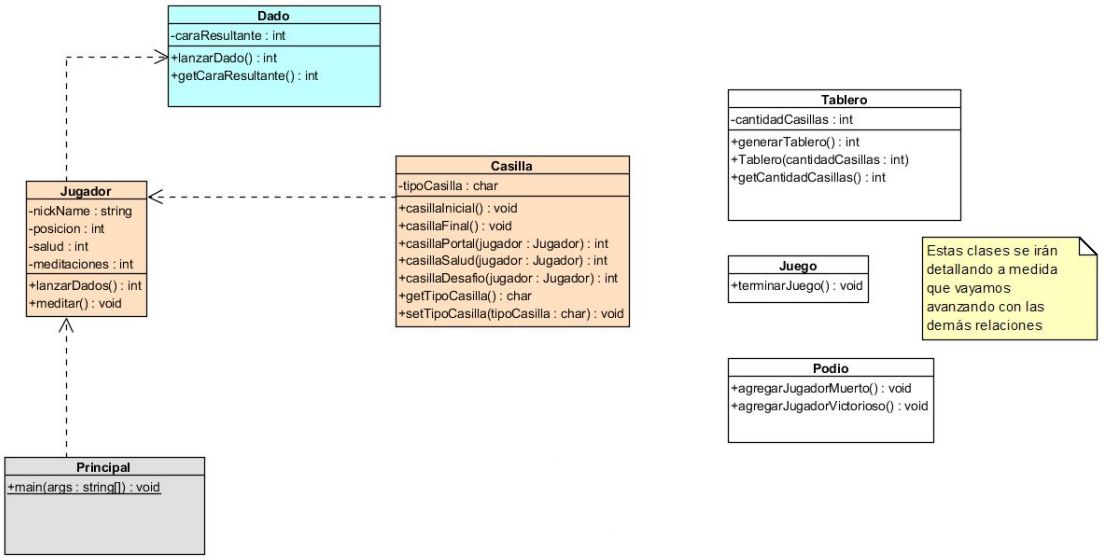
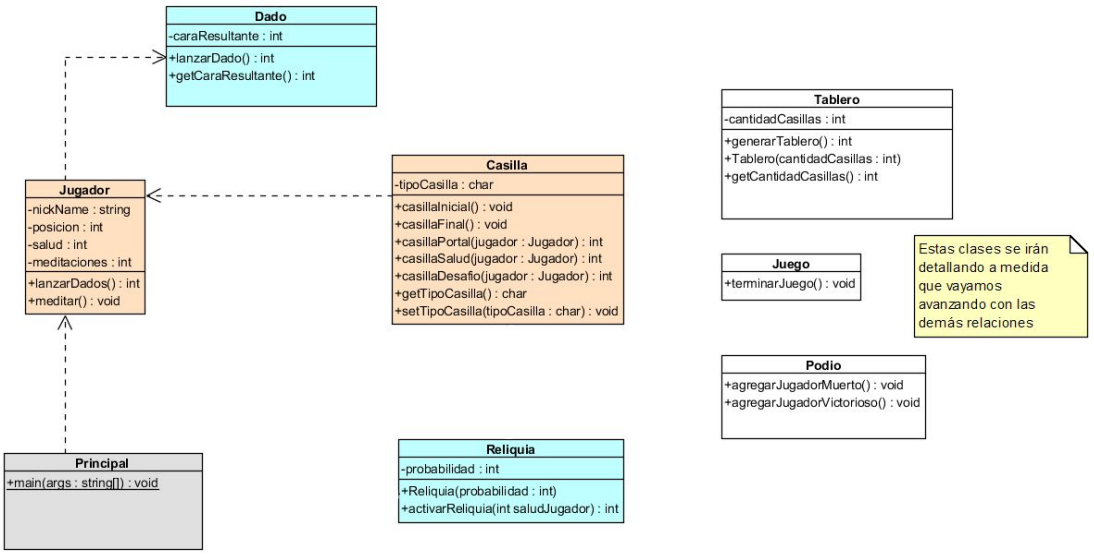
Pauta tarea 5.

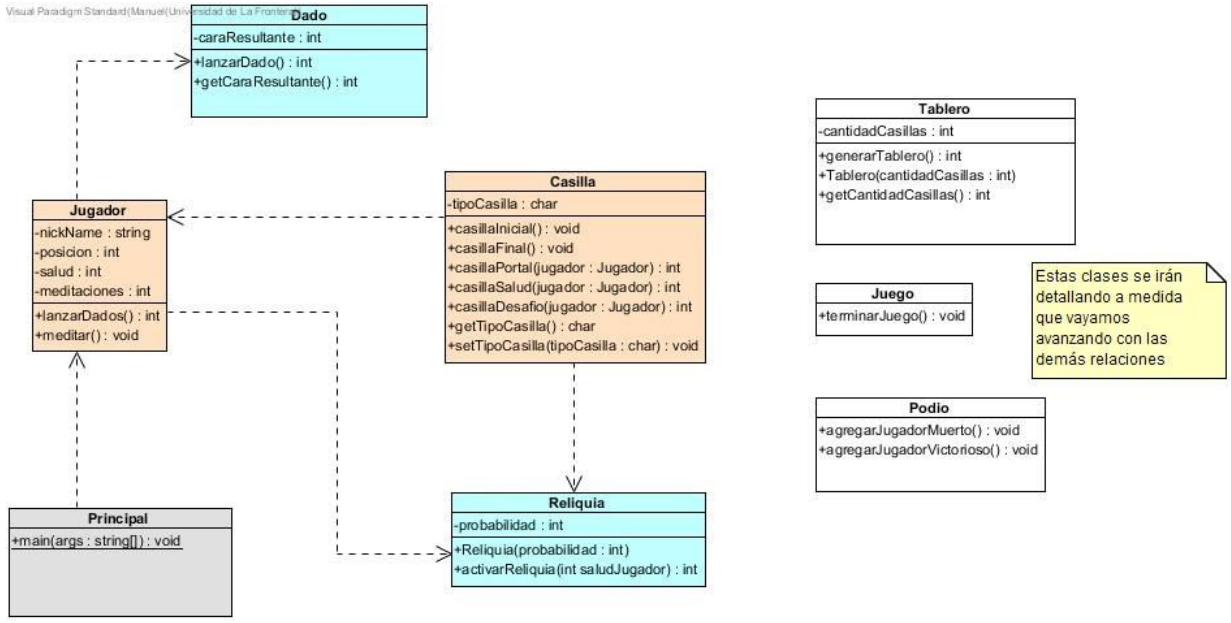
Teniendo en cuenta las clases que se identificaron en la tarea anterior, considerando sus respectivos atributos y métodos, al identificar las relaciones de dependencias, resultaría el siguiente diagrama UML:



Ahora incluímos la clase que se logra identificar en el enunciado de la tarea 5: Reliquia.



Por último, se agregan las relaciones de dependencia asociadas a la nueva clase. En este caso, “Jugador y Casilla usan la Reliquia” (la segunda en a través del método casillaSalud).



¿Cómo se visualiza en el código?

En primer lugar se genera el código desde Visual Paradigm hacia algún proyecto java para obtener la estructura de clases. Posterior a esto, agregamos el código que corresponde a las relaciones de dependencia.

“Jugador usa dados”:
Se habla de dependencia al momento de instanciar la clase Dado dentro de un método de la clase Jugador.

```java
public class Jugador {
    private String nickName;
    private int posicion;
    private int salud;
    private int meditaciones;

    public int lanzarDados() {
        Dado d1 = new Dado();
        Dado d2 = new Dado();

        return d1.lanzarDado()+d2.lanzarDado();
    }

    public void meditar() {
        /*
        .....
        */
    }
}
```

“Jugador usa Reliquia”:
Nos referimos a dependencia al momento de instanciar la clase Reliquia dentro de un método de la clase jugador.

```java
public class Jugador {
    private String nickName;
    private int posicion;
    private int salud;
    private int meditaciones;

    public int lanzarDados() {
        Dado d1 = new Dado();
        Dado d2 = new Dado();

        int resultado = d1.lanzarDado()+d2.lanzarDado();

        Reliquia rel;

        if (resultado == 12){
            rel = new Reliquia(50);
        }else{
            rel = new Reliquia(1)
        }

        rel.activarReliquia(salud);

        return resultado;
    }

    public void meditar() {
        /*
        .....
        */
    }
}
```

“Casilla usa a Jugador y a Reliquia”:
Se dice que Casilla usa a Jugador al momento de pasar como parámetro un objeto Jugador en uno de los métodos. Por otra parte, Casilla depende de Reliquia al momento de instanciar la clase reliquia dentro de uno de los métodos, particularmente en el método casillaSalud().

```java
public class Casilla {
    private char tipoCasilla;

    public void casillaInicial() {
        //...
    }

    public void casillaFinal() {
        //...
    }

    public int casillaPortal(Jugador jugador) {
        //...
    }

    public int casillaSalud(Jugador jugador) {
        //...
        Reliquia rel = new Reliquia(5);
        rel.activarReliquiaMeditaciones();
    }

    public int casillaDesafio(Jugador jugador) {
        //...
    }

    public char getTipoCasilla() {
        return this.tipoCasilla;
    }

    public void setTipoCasilla(char tipoCasilla) {
        this.tipoCasilla = tipoCasilla;
    }
}
```