Marc Garcia - 1559591
Marta Gutiérrez - 1673889

INFORME DE PRÀCTIQUES: UNO

Link Repositori: https://github.com/MarcGarciaUAB/Uno-TQS

1. Arquitectura del Sistema (MVC)

El projecte s'ha dissenyat seguint el patró Model-Vista-Controlador per garantir la separació de responsabilitats i facilitar el testeig unitari.

1.1 Descripció dels Paquets

- **Model:** Conté la lògica i l'estat del joc. Aquest paquet és independent de la interfície d'usuari.
  - *Classes principals:* Carta, Baraja, Mano, Pila.
  - *Responsabilitat:* Gestionar les regles del joc, l'estat de les cartes i la lògica base.
- **Vista:** Encarregada de la interacció amb l'usuari.
  - *Classes principals:* GameUI.
  - *Responsabilitat:* Mostrar l'estat de la partida i recollir les accions del jugador (tirar carta, robar).
- **Controlador:** Actua com a intermediari i gestiona el flux del joc.
  - *Classes principals:* GameController.
  - *Responsabilitat:* Rep els inputs de la vista, invoca la lògica del model, gestiona el fluxe del joc i actualitza la vista segons el resultat.

2. Estratègia de TDD i Coverage

S'ha seguit estrictament el format TDD: primer es creen els tests que fallen, s'implementa el codi mínim per passar-los i finalment es refactoritza.

2.1 Resum de Cobertura (Coverage)

A continuació es mostren les mètriques de cobertura obtingudes:

Element	Coverage	Covered Instructions	Missed Instructions	Total Instructions
Uno-TQS	82,0 %	3.571	783	4.354
src/main/java	70,6 %	1.291	537	1.828
org.uno.vista	0,0 %	0	301	301
org.uno.model	88,0 %	770	105	875
Carta.java	86,5 %	351	55	406
Baraja.java	92,2 %	236	20	256
Mano.java	88,2 %	112	15	127
Pila.java	82,6 %	71	15	86
org.uno	0,0 %	0	80	80
org.uno.controlador	91,1 %	521	51	572
GameController.java	91,1 %	521	51	572
src/test/java	90,3 %	2.280	246	2.526

Marc Garcia - 1559591
Marta Gutiérrez - 1673889

Es veu com totes les línies de codi al voltant del 90% tenen coverage, i les restants són pre/post condicions i invariants. Tant vista com org.uno (main) no tenen coverage ja que no és necessari testejar.

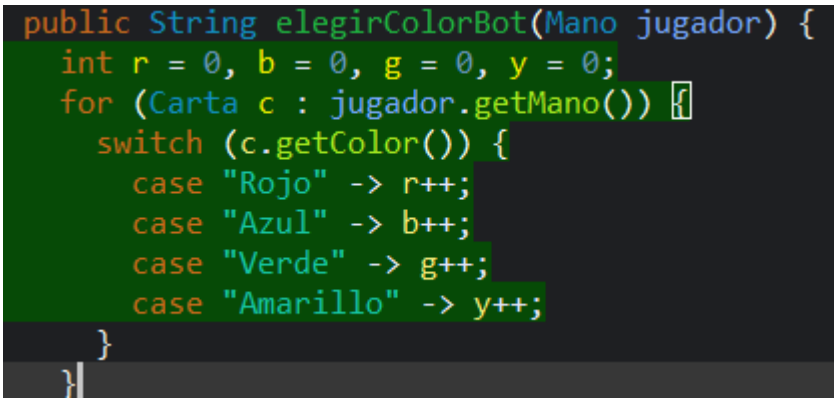
2.2 Proves de Caixa Blanca

S'han dissenyat proves específiques per validar l'estructura interna del codi.

A. Loop Testing (Prova de Bucles)

S'ha analitzat el bucle encarregat de decidir el color seleccionat per la màquina.

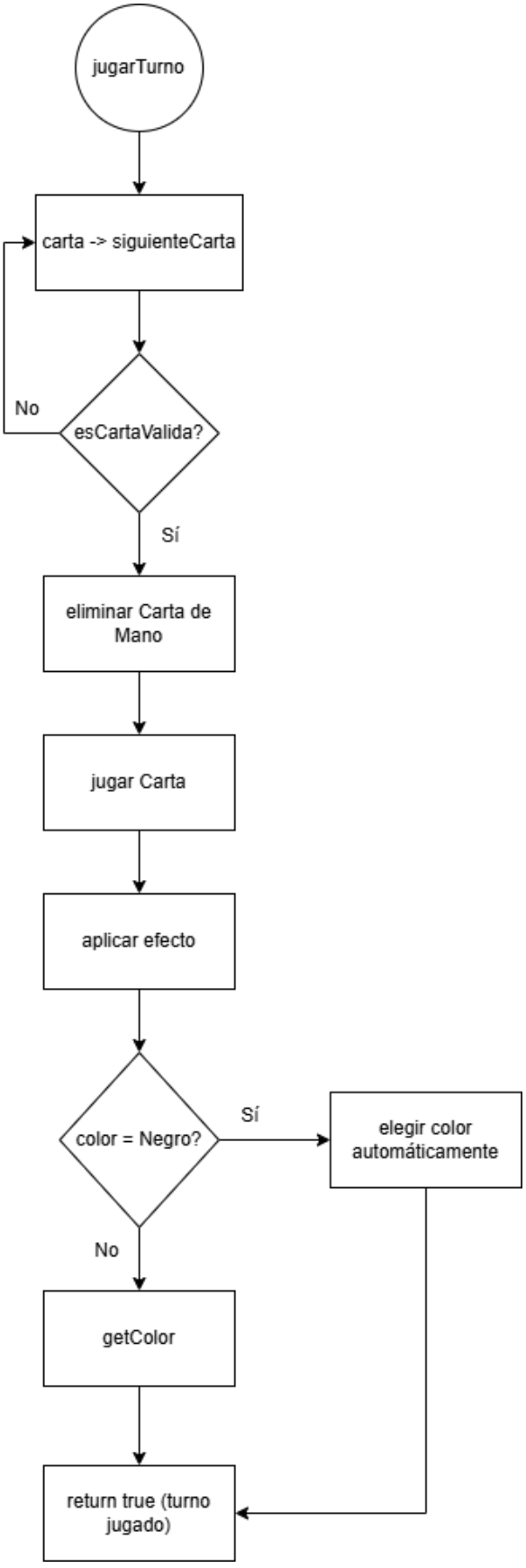
- **Codi analitzat:** elegirColorBot.
- **Cas 0 iteracions:** no té més cartes.
- **Cas n iteracions:** Té més d'una carta de diferents colors.



Marc Garcia - 1559591
Marta Gutiérrez - 1673889

B. Path Coverage i Diagrama de Flux

S'ha analitzat el mètode complex jugarTurno , que conté la lògica principal del torn.



Marc Garcia - 1559591
Marta Gutiérrez - 1673889

```java
public boolean jugarTurno(Mano jugador) {
  for (Carta c : jugador.getMano()) {  // loop simple
    if (esCartaValida(c, jugador)) {
      jugador.eliminarCarta(c);
      pila.jugarCarta(c);
      aplicarEfecto(c, jugador);
      colorActual = c.getColor().equals("Negro") ? elegirColorBot(jugador) : c.getColor();
      return true;
    }
  }
```

C. Decision & Condition Coverage

S'han verificat les condicions complexes, com ara la validació de si una carta es pot tirar sobre una altra.

- **Mètode:** esCompatible(Carta c1, Carta c2)
- **Condició:** (c1.color == c2.color || c1.valor == c2.valor || c1.esComodin)
- **Tests:** S'han creat casos per fer True i False cada part de la condició lògica.

3. Mock Objects

Per aïllar les classes del model i testejar sense dependències externes o aleatorietat, s'ha utilitzat **Mockito**.

Mock Object	**Tipus**	**Funció**	**Justificació**
Baraja	Mockito	Simular Baraja	Permet assegurar les cartes que es robaran quan es crida robar.
Krupier	Mockito	Controlar l'atzar	Permet assegurar l'ordre de les cartes quan es baralla.

4. Proves de Caixa Negra

S'han dissenyat tests basats en les especificacions del joc UNO (reglament oficial), utilitzant tècniques de:

- **Particions Equivalents:** Cartes normals (1-9), Cartes Acció (+2, Skip), Cartes Comodí.
- **Valors Límit:** Robar quan la baralla està buida (cal re-barrejar), jugar cartes que no es poden, etc.

5. Gestió de Versions i CI

5.1 Flux de Treball (Git Flow)

Marc Garcia - 1559591
Marta Gutiérrez - 1673889

S'ha treballat amb una branca develop on s’ha anat desenvolupant en equip tot el projecte, fent merges a main quan el projecte estava funcional i un cop s’ha fet tot el coverage necessari.

5.2 Integració Automàtica (GitHub Actions)

S'ha configurat un pipeline de CI que executa automàticament:

1. Compilació del projecte (Maven/Gradle).
2. Execució de tots els tests unitaris.