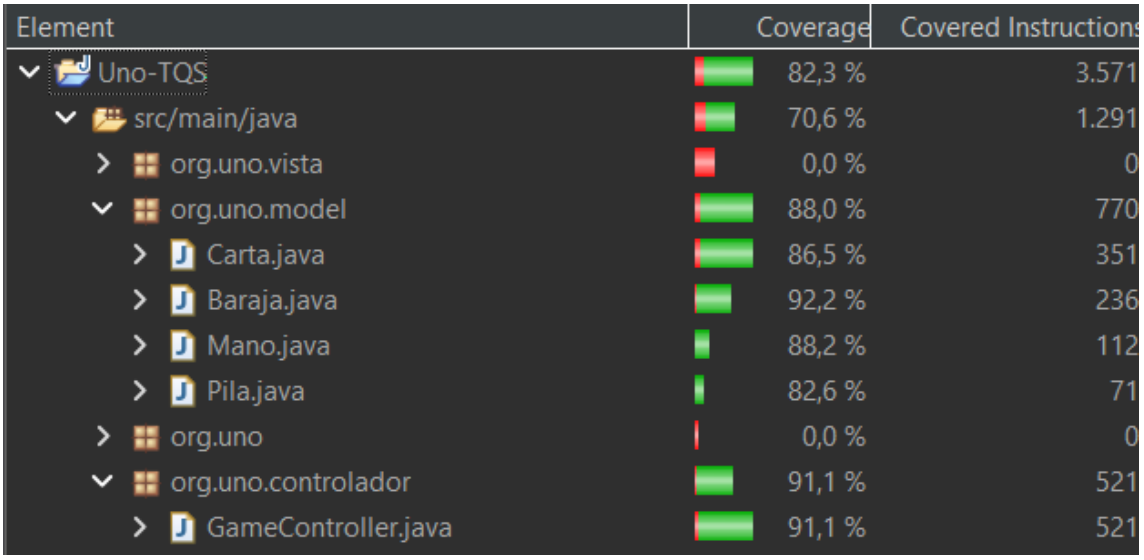
Marc Garcia Martín - 1559591
Marta Gutiérrez Vila - 1673889
Dimecres 8:30h
Uno-TQS: https://github.com/MarcGarciaUAB/Uno-TQS

Funcionalitat: Statement Coverage
Localització: Tots els arxius, Eclipse IDE
Test: s'ha realitzat TDD a tot el projecte, i s'ha fet un coverage de més del 90% de les línies de codi testejades, sent un 100% de les importants.



Element	Coverage	Covered Instructions
Uno-TQS	82,3 %	3.571
src/main/java	70,6 %	1.291
org.uno.vista	0,0 %	0
org.uno.model	88,0 %	770
Carta.java	86,5 %	351
Baraja.java	92,2 %	236
Mano.java	88,2 %	112
Pila.java	82,6 %	71
org.uno	0,0 %	0
org.uno.controlador	91,1 %	521
GameController.java	91,1 %	521

No és un coverage del 100% perquè hem considerat una bona pràctica introduïr invariants, precondicions i post-condicions, i algunes de les condicions i invariants no han sigut testejades en ambdós estats (true/false), baixant així el percentatge. Pel que fa al codi íntegre com a tal, s'ha fet un coverage complet.

Funcionalitat: Loop Testing
Localització: Baraja.java, Baraja, robar
Test: BarajaTest.java, testPathCoverageRobar, Caixa blanca, Loop Testing

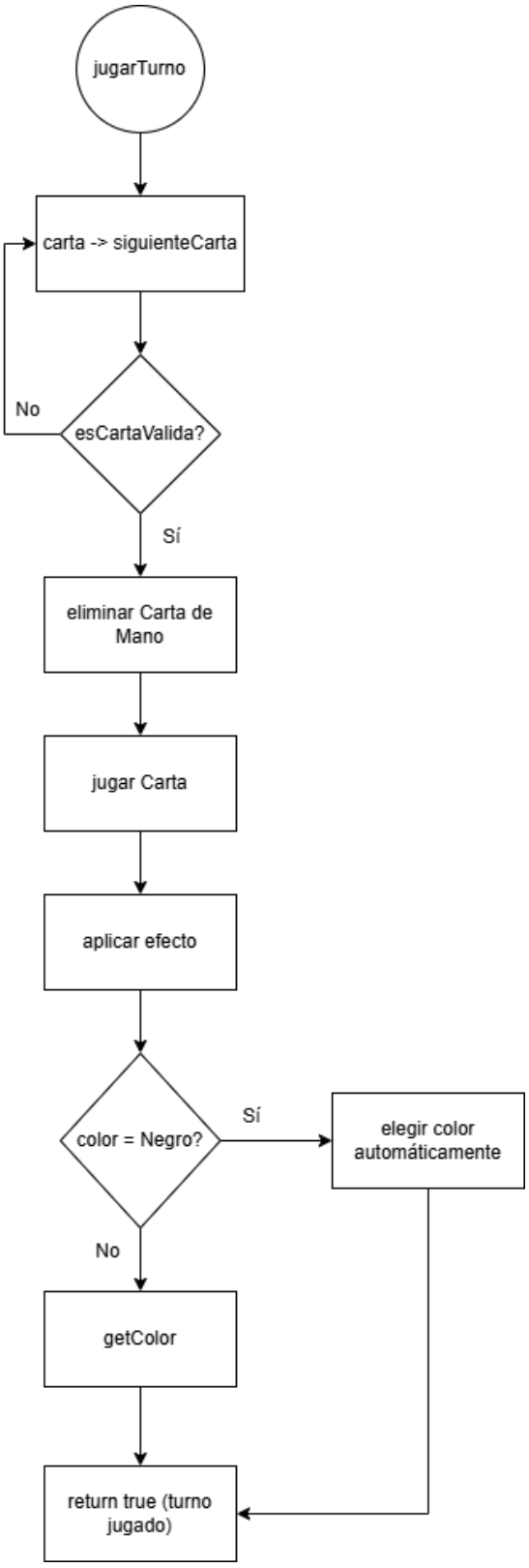
```
  public List<Carta> robar(int n) {
    List<Carta> robadas = new ArrayList<>();
    for (int i = 0; i < n && !baraja.isEmpty(); i++) { // loop simple
      robadas.add(robar());
    }
    return robadas;
  }
}
```

Els casos de 1, 2 o n cartes s'han comprovat des del +2, +4 i robar normals a la resta de tests, els quals es poden veure a l'arxiu GameControllerTest.java

Funcionalitat: Path Coverage

Localització: GameController.java, GameController, jugarTurno

Test: GameControllerTest.java, seqüència de tests que comencen amb “testJugarTurno”, caixa blanca, Path Coverage



Funcionalitat: Condition i Decision Coverage

Localització: GameController.java, GameController, iniciarPila

Test: GameControllerTest.java, testIniciarPila, caixa blanca, condition/decision coverage

A tot el GameControllerTest es pot veure moltes instàncies de decision i condition coverage, però aquí tenim un exemple, es passa per tots els casos del switch-case.

```
@Test  1673889
void testIniciarPilaColorNegroCubreTodasLasRamasDelSwitch() {
  // vamos a llamar iniciarPila hasta recoger los 4 colores distintos que el switch produce
  Set<String> vistos = new HashSet<>();
  int max = 200;
  for (int i = 0; i < max && vistos.size() < 4; i++) {
    Carta negra = new Carta( e: "Change",  c: "Negro");
    controlador.iniciarPila(negra);
    String color = controlador.getColorActual();
    if (color != null) vistos.add(color);
  }
```

Funcionalitat: Mock Object nº1

Localització: GameControllerTest.java, Baraja, tot l'arxiu

Test: GameControllerTest.java, mock objects.

Mock de baraja per a poder robar cartes específiques

```
@Test  1673889
void testRobarCartaCuandoBarajaDevuelveCarta() {
  when(mockBaraja.robar()).thenReturn(new Carta( v: 9,  c: "Verde"));
  Carta rob = controlador.robarCarta(jugador1);
  assertNotNull(rob);
  assertEquals( expected: 1, jugador1.getNumeroCartas());
}
```

Funcionalitat: Mock Object nº2

Localització: BarajaTest.java, Krupier, tot l'arxiu

Test: BarajaTest.java, mock objects

Mock de la interfície Krupier per a poder barallar

```
// Barajar usando mock
@Test  1673889
void testBarajar() {
  List<Carta> original = baraja.getBaraja();
  when(mockKrupier.barajar(original)).thenReturn(new ArrayList<>(original));
  baraja.barajar();
  // Statement coverage y mock object probado
  assertEquals(original.size(), baraja.tamaño());
}
```

Funcionalitat: Pairwise Testing

Localització: Carta.java, Carta, mismoValor/mismoColor

Test: CartaTest.java/ManoTest.java, testMismoValor/testMismoColor/testTieneCartaJugable, caixa negra, pairwise testing

Els tests permeten comprovar si els valors són iguals o no, si els colors són iguals o no o si al ser carta especial sempre ho permet mitjançant pairwise testing (juntant valors i colors)

Funcionalitat: Valors Límit

Localització: tot el codi, per exemple Carta.java, carta, invariant

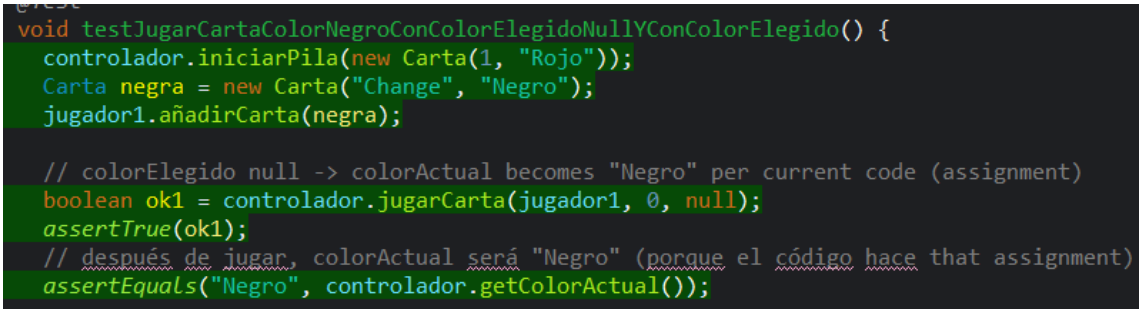
Test: CartaTest.java, tots els testInvariante, caixa negra, valor límit.

Funcionalitat: Particions equivalents

Localització: GameController.java, GameController, jugarCarta

Test: GameControllerTest.java, testJugarCarta, caixa negra, particions equivalents.

Com no provarem totes les combinacions de color i valor, es comprova si juntant el mateix color o el mateix valor funciona i viceversa.



```
void testJugarCartaColorNegroConColorElegidoNullYConColorElegido() {
    controlador.iniciarPila(new Carta(1, "Rojo"));
    Carta negra = new Carta("Change", "Negro");
    jugador1.añadirCarta(negra);

    // colorElegido null -> colorActual becomes "Negro" per current code (assignment)
    boolean ok1 = controlador.jugarCarta(jugador1, 0, null);
    assertTrue(ok1);
    // después de jugar, colorActual será "Negro" (porque el código hace that assignment)
    assertEquals("Negro", controlador.getColorActual());
```