Nom: Robert Masip Quintana

Niu:1459442

Nom pràctica: Joc Mastermind

Data: 15/11/2022

GitHub pràctica: https://github.com/catwrc/MasterMind

Main

Funcionalitat: Constructor del Main, inicialitza el tauler, el codi secret, les variables booleanes per si has guanyat, perdut i l'estat del joc, i el número d'intents per introduir el codi que és 8.

Localització: Mastermind/src/Controlador/Main.java

Test El test comprova que les variables s'hagin iniciat amb els valors correctes.

Tipus test: Caixa blanca i statement coverage

Localització test: Mastermind/TestMasterMind/TestMain.java

Funcionalitat: :Agafa el codi introduït per l'usuari EscollirColor() i si es correcte l'introdueix al tauler AfegirCodiTaulell(codi).

Localització: Mastermind/src/Controlador/Main.java i les funcions Mastermind/src/vista/Tauler.java i Mastermind/src/Model/Codi.java

Test: S'encarrega de comprovar que el codi que introdueixi per teclat sigui un codi tipus com l'exemple "RYGO", que és el predefinit en el joc i l'acceptat (amb els colors ("R", "Y", "G", "B", "O", "P") i de llargada màxima 4 colors).

Tipus test: Caixa blanca i statement coverage.

Localització test: Mastermind/TestMasterMind/TestCodi.java

Funcionalitat: Generar un codi aleatori que l'usuari haurà de trobar escrivint codis per teclat, a través de les pistes que se li aniran donant, al generar un codi aleatori i el codi introduït per l'usuari ja podem generar unes pistes que li donarem a l'usuari a través del taulell.

Localització: Mastermind/src/Controlador/Main.java i les funcions Mastermind/src/vista/Tauler.java i Mastermind/src/Model/CodiSecret.java

Test: S'encarrega de comprovar que el codi generat aleatòriament tingui una llargada de 4 colors, I també fem diferents test per veure que el codi que s'ha generat contingui només els colors correctes("R", "Y", "G", "B", "O", "P").

Tipus test: Caixa blanca i statement coverage.

Localització test: Mastermind/TestMasterMind/TestSecretCode.java

Funcionalitat: He creat també un mock object que juga al joc sense la intervenció de cap jugador, això ho hem implementat amb dues classes, la Joc_Interficie i la SecretCode_Interficie així podem simular el joc sense saber el codi secret ni introduir codi per trobar-lo.

Localització: Mastermind/src/Controlador/Main.java i Mastermind/src/Model.

Test: Hem creat un test que simula una partida a aquest test li passem les entrades d'un jugador simulat diem-li jugador 2 i el codi secret generat aleatòriament, aquest jugador anirà afegint els codis extrets d'un array que contindrà una llista de codis, comprovant els codis amb buna altra llista de pistes anirem creant el joc fins a arribar al final.

Tipus test: Caja Blanca , Decision Coverage, Condition Coverage.

Localització test: MasterMind/TestMasterMind/TestMain.java

Funció que recrea el mock.

```
public void mainMock(){
    int intents=0;
    boolean acabat=false;
    int i=1;
    while(!acabat){
        if(intents<Oportunitats){
            String codi = this.simulacioJoc.introduirCodi(); // llista codis simulat Jugador2
            this.llistaIntents.add(codi);
            while ( codi == null){
                System.out.println("El codi es null");
            }
            introdueixCodi(codi);
            intents++;
        }
        else{
            acabat = true;
        }
        i++;
    }
}
```

Test del mock li passem la llista de pistes, el mock del jugador2, creem el codi secret amb el mock de codiSecret i cridem a la funció de la imatge superior mainmock() perquè executi el joc.

```
@Test
public void Llista_codis(){
    List<String> pistesCorrectes  = new ArrayList<~>(List.of(
            "    ", "  F ", " FF ", "TF  ", "T  T", "TF T", "TFTF", "TTTT"));
    MockJugador2 jugador_2 = new MockJugador2();
    Main game = new Main();
    MockCodiSecret codi_secret = new MockCodiSecret();
    codi_secret.setCodiSecret("YYPO"); //Comprovem les pistes amb el codi
    game.setJugador(jugador_2);
    game.codisecretInterficie(codi_secret);
    game.mainMock();
    int i =0;
    for(Joc p: game.getTauler().getPistasTauler()){
        assertEquals(pistesCorrectes.get(i), p.getJoc());
        i++;
    }
}
```

Aquí tenim la llista de partides que introduirà el mock del jugador2.

```
public static final List<String> codiJugador = new ArrayList<>(List.of("FFFF","BGYR", "BOYR", "YOGR", "YRGO", "YOGO", "YOPY","YYPO"));
```

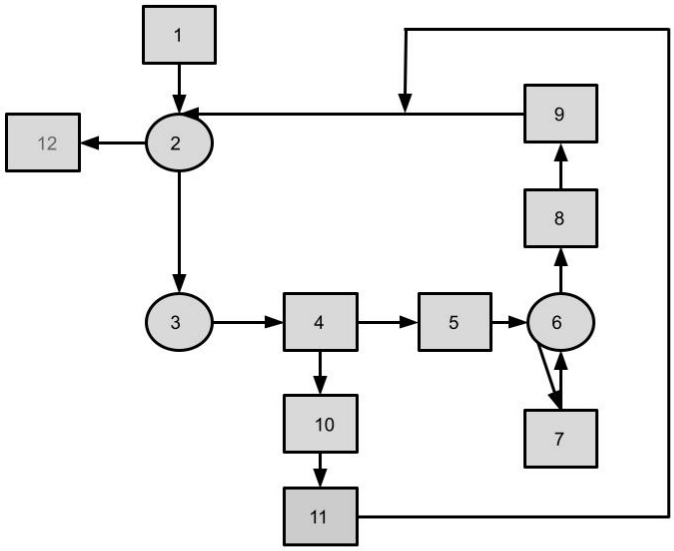
Funcionalitat: També he creat una altra funció per comprovar el test de path coverage, utilitzant la funció pathCoverage().

Localització: Mastermind/src/Controlador/Main.java-> pathCoverage().

Test: Utilitza uns inputs que li passem a la funció i la funció passarà per tots els camins possibles així executem totes les decisions i condicions.

Tipus test: Caja Blanca , Decision Coverage, Condition Coverage.

Localització test: MasterMind/TestMasterMind/TestMain.java



Funcionalitat: Aquesta funcionalitat recrea el codi que introduiria per teclat.

Localització: Mastermind/src/Controlador/Main.java

Test: Aquest test comprova que totes les parelles discretes de colors siguin provades en el joc. Per això hem hagut d'utilitzar la interfície SecredCode_Interficie ja que la classe general genera un codi secret aleatori i les pistes resultants serien sempre diferents.

Tipus test: Caixa Negra , Pairwise testing, MockCodisecret.

Localització test: Mastermind/TestMasterMind/TestMain.java, Mastermind/src/pairwise.csv i MasterMind/src/Model/Mocks/MockCodiSecret.java

Joc

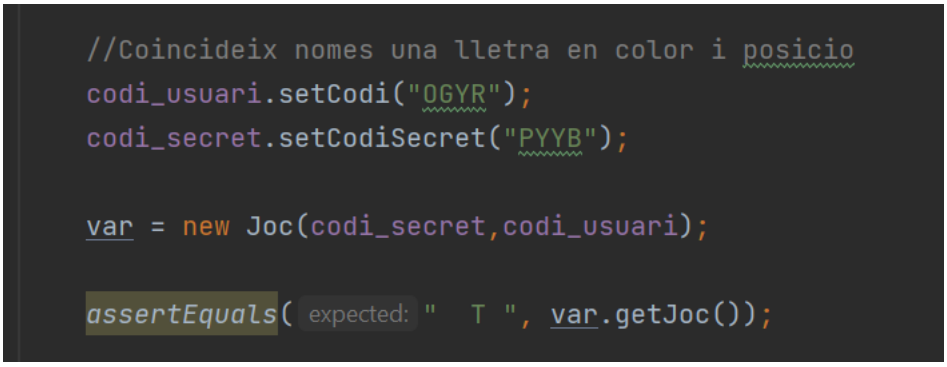
Funcionalitat: Crea un objecte Joc a partir del codi secret i el codi introduït per l'usuari, amb aquesta classe el que gestiona és com col·locar les pistes en el taulell en rebre el codi introduït per l'usuari i el codi secret podem anar comparant les posicions dels diferents codis i anar posant les pistes en el taulell, les pistes rebran forma de lletra amb la lletra T quan en la posició i el color són correctes, una F quan el color existeix en el codi, però no està a la posició correcta i posem conjunt buit quan el color no existeix el codi secret.

Localització: Mastermind/src/Model/Joc.java.

Test: En la classe test tenim dos tipus de test un que introduint el codi secret I el codi que introdueix l'usuari retorni les pistes correctament.
L'altre test és de condition i decision coverage, comprobem a la funció Joc() totes les

combinacions de true i false per fer les comprovacions de decision i contidiotn coverage.**Tipus test:** Caixa blanca i statement coverage coondition coverage, statement coverage.

Localització test: Mastermind/TestMasterMind/Joc.java



Codi

Funcionalitat: Aquesta classe el que fa és que a partir d'un codi generat per l'usuari que serà un String, pasem aquest String a un objecte del tipus codi.

Localització: Mastermind/src/Model/Codi.java

Test: Fem test per comprovar que s'introdueixi un codi correcte.

Tipus test: Caixa blanca i statement coverage.

Localització test: Mastermind/TestMasterMind/TestCodi.java

Funcionalitat: Comprova que converteixi el codi introduït per l'usuari a majúscules

Localització: Mastermind/src/Model/Codi.java

Test: Fem un test per comprovar que es retornin les lletres dels color a majúscules

Tipus test: Caixa blanca i statement coverage.

Localització test: Mastermind/TestMasterMind/TestCodi.java

Codi Secret

Funcionalitat: Creació d'un objecte CodiSecret a partir d'un string generat aleatòriament que contindrà el codi secret

Localització: Mastermind/src/Model/CodiSecret.java

Test: Hem realitzat proves de Caixa blanca per veure que el codi generat sigui el correcte.

Tipus test: Caixa blanca

Localització test: Mastermind/TestMasterMind/TestSecretCodejava

Funcionalitat: Generem aleatòriament un codi que contingui els caràcters corresponents ("R", "Y", "G", "B", "O", "P") i tingui una llargada igual a 4.

Localització: Mastermind/src/Model/CodiSecret.java

Test: Hem realitzat un test per comprovar aquesta funcionalitat.

Tipus test: Caixa blanca

Localització test: Mastermind/TestMasterMind/TestSecretCodejava

```java
    @Test
    public void Test4(){
        codi = new CodiSecret();
        String v_codi = codi.getSecretCode();
        boolean var = true;
        if(v_codi.length() == 4) {
            for (int x = 0; x < 4; x++) {
                char c = v_codi.charAt(x);
                if (c=='R'|| c=='Y'|| c=='G'|| c=='B'|| c=='O'|| c=='P') {
                } else {
                    var = false;
                    Assert.assertTrue(var);
                }
            }
            Assert.assertTrue(var);
        }
        else {
            var = false;
            Assert.assertTrue(var);
        }
    }
}
```

IntroduirColors

Funcionalitat: En aquesta classe és la que gestiona que l'usuari introdueixi el codi pel teclat.

Localització: Mastermind/src/Model/IntroduirColors.java

Test: Comprava que l'usuari hagi introduït un codi correcte, si no introdueix un codi correcte té 2 oportunitats més durant tot el joc per introduir caràcters correctes.

Tipus test: Caixa blanca

Localització test: Mastermind/TestMasterMind/TestIntroduirColors.java

Funcionalitat: Aquesta funció comprova que el codi introduït per l'usuari es trobi als límits del joc i sigui d'una correcta longitud, de mida quatre i usant els caràcters disponibles que representen a colors.

Localització: Mastermind/src/Model/IntroduirColors.java

Test: Es comprova que la funció retorni true o false correctament si els codis introduïts són realment correctes, o no provant amb diversos tipus de possibles codis.

Tipus test: Caixa blanca

Localització test: Mastermind/TestMasterMind/TestIntroduirColors.java

```java
public static boolean NomesConteLletres(String cadena) {
    for (int x = 0; x < cadena.length(); x++) {
        char c = cadena.charAt(x);
        if (!(c=='R'|| c=='Y'|| c=='G'|| c=='B'|| c=='O'|| c=='P')) {
            return false;
        }
    }
    return true;
}
```

Funcionalitat: Aquesta funció comprova que el codi introduït per l'usuari es trobi als límits del joc i sigui correcte (longitud de mida quatre i usant els caràcters disponibles que representen a colors).

Localització: Mastermind/src/Model/IntroduirColors.java

Test: S'ha creat un dataset de codis possibles i s'ha dividit en classes equivalents en funció de valors límit de longitud dels codis. Per a cada classe hi ha valors vàlids (els colors que accepta el joc) i invàlids (qualsevol altre tipus de caràcter). També comprova que totes les possibles decisions i condicions s'executen al mètode.

Tipus test: Caixa Negra , Partició equivalent, Valors límit, Decision Coverage, Condition Coverage.

Localització test: Mastermind/TestMasterMind/TestIntroduirColors.java i Mastermind/src/lletres.csv

null	Límit exterior frontera	Límit exterior frontera	frontera	frontera	Límit exterior frontera	Límit exterior frontera	Límit exterior frontera

```
null;A;YYY;RFDG;TRCV;OGRBY;OGRBY;DFRBYOGRBYOGRBYOGRBY
null;B;OGr;VGRT;BGVE;BRY#P;rdFGT;YPVOBYPVOBYPVOBYPVOB
null;R;OGP;SMNY;OHBTR;1G2B!;1G2BY;12345OGRBYOGRBYOGRBY
null;VM;UYG;BlPO;PONV;12HRT;FGT56;rdftgYPVOBYPVOBYPVOB
null;RY;12W;GBYV;MASI;98765;12345;YFCEZYPVOBYPVOBYP123
null;-1;-542;RPVO;ORB5;-90760;38293;1,23457E+19
```

```java
//test de particions equivalents parametritzat
catwrc
@ParameterizedTest
@CsvFileSource(resources = "lletres.csv",numLinesToSkip = 1,delimiterString = ";")
public void Parametritzacio(final String test1, final String test2,
                            final String test3, final String test4,
                            final String test5, final String test6,
                            final String test7, final String test8) {

    IntroduirColors i = new IntroduirColors();

    //valors null
    assertFalse(i.esCorrecte(test1));
    //valors fora de la frontera
    assertFalse(i.esCorrecte(test2));
    //valors limit frontera
    assertFalse(i.esCorrecte(test3));
    //valors frontera
    assertFalse(i.esCorrecte(test4));
    //valors frontera
    assertFalse(i.esCorrecte(test5));
```

Tauler

Funcionalitat: És la classe que tindrem per visualitzar el taulell del joc, per una part tindrem el codi que anem introduïts i per l'altra el de les pistes per generar el següent codi.

Localització: Mastermind/src/Vista/Tauler.java

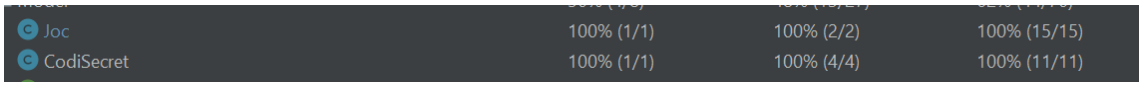
Test: Comprovem que el taulell es generi amb les mides i la forma correcta.

Tipus test: Caixa blanca

Localització test: Mastermind/TestMasterMind/TestTauler.java

Statement coverage

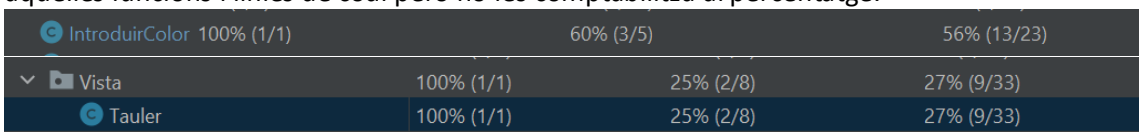
Aquí tenim una mostra del statement coverage del codi de la pràctica en què s'aprecia com totes les sentències del nostre codi (una vegada excloses les funcions que usàvem únicament per als tests) s'executen com a mínim una vegada. Cal destacar que el percentatge de línies executades és del 100% menys en el Main. En els resultats de les captures he simulat un joc real i també he tret els Mock Objects i tests de forma temporal només per fer el coverage. De totes maneres en haver fet TDD hem assegurat el Statement Coverage de tot el codi.



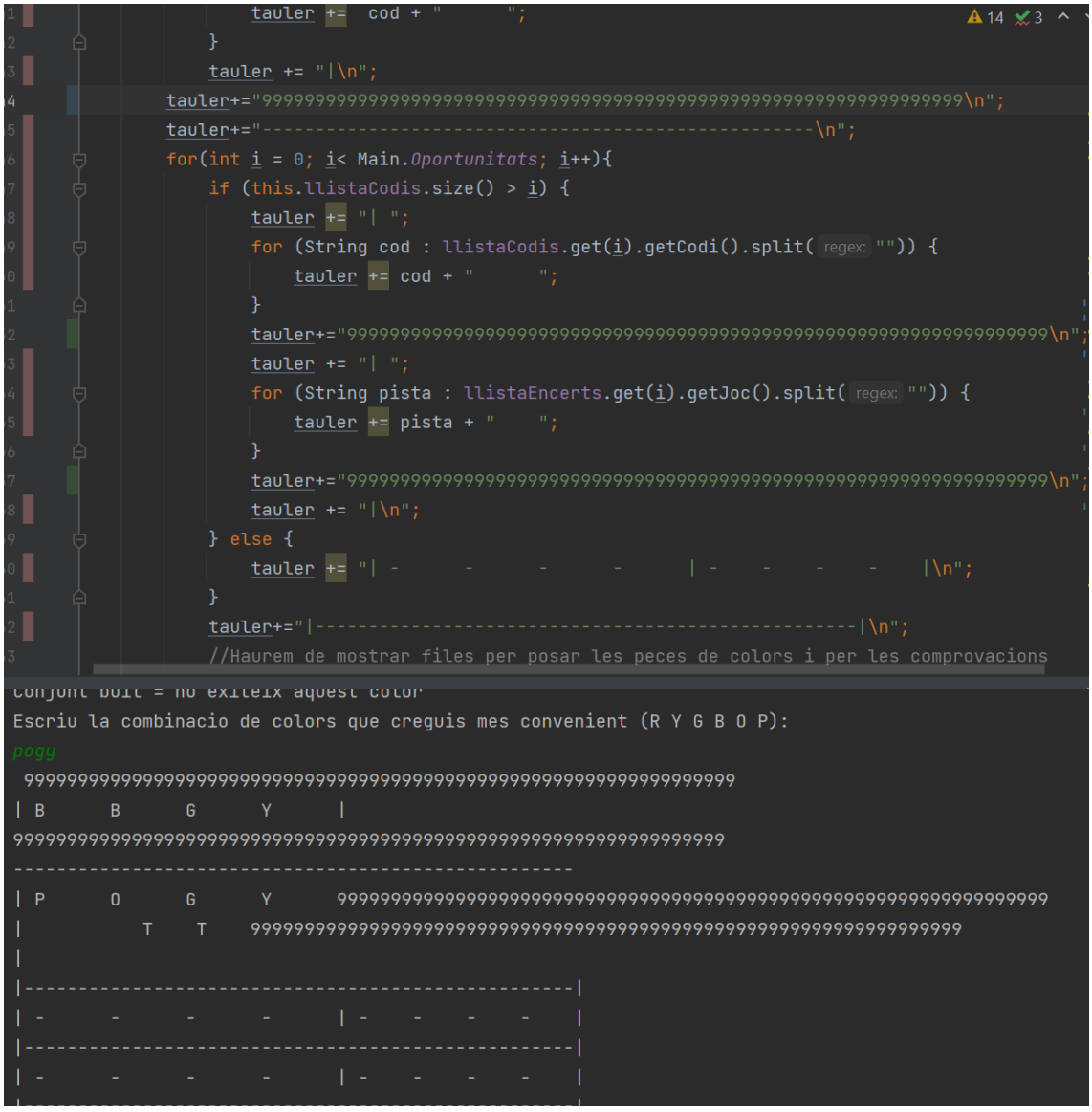
Aquí podem veure que només tenim un 84% de cobertura, això és a causa del fet que tenim una funció win i lose que només s'executen si guanyes o perds en aquest cas és per la versió en què perds a la partida.

Aquesta versió de la cobertura del main és per a les partides en què has guanyat

Amb les classes tauler i IntroduirColors m'he trobat la problemàtica que no m'indica el percentatge de coverage correcte tot i passar per aquelles funcions de la classe mencionada. Més avall del document he deixat imatges de la demostració que si passa per aquelles funcions i línies de codi però no les comptabilitza al percentatge.



Tauler



EscollirColor

```java
1 usage   catwrc *
public static String EscollirColor(){
    while(intents<3){ //2 intents permesos d'introduir codis no correctes
        System.out.println("99999999999999999999999999999999999999999999");

        String llista = escriure();
        if(esCorrecte(llista)){
            System.out.println("999999999999999999999999999999999999999999999");
            return llista;

        }
        intents++;
        System.out.println("99999999999999999999999999999999999999999999");
    }
  return null;
}
```

```
99999999999999999999999999999999999999999999

Escriu la combinacio de colors que creguis mes convenient (R Y G B O P):
rygo
99999999999999999999999999999999999999999999
```