

UNIVERSITAT DE LLEIDA



GRAU EN ENGINYERIA INFORMÀTICA  
ENGINYERIA DEL PROGRAMARI

---

## Tercera pràctica d'Enginyeria del Programari

---

*Autors:*  
Jaume Giralt Barbé

*Professor:*  
Juan Manuel Gimeno Illa

23 de gener de 2017

# Índex

<b>1</b>	<b>Contingut</b>	<b>2</b>
<b>2</b>	<b>Explicació dels tests</b>	<b>3</b>
2.1	Paquet data . . . . .	3
2.1.1	Mètodes test del paquet data . . . . .	3
2.2	Paquet Kiosk . . . . .	4
2.2.1	Classe Activate Emission Test . . . . .	4
2.2.2	Classe Activation Card Test . . . . .	5
2.2.3	Can Vote Test . . . . .	6
2.2.4	Send Receipt Test . . . . .	6

## 1 Contingut

Per la realització d'aquesta pràctica he dividit el contingut amb dos directoris. El directori *src* i el directori *tests*. En el directori *src* s'hi poden trobar els fitxers escrits amb llenguatge java que simulen el funcionament d'un sistema de votació electrònica. No he realitzat la part opcional de introduir el sistema de autenticació del iris per fer una mala planificació del temps a la hora de realitzar la pràctica.

En el directori *tests* estan implementats els jocs de tests per provar la pràctica. En aquest document intentaré explicar el que he volgut provar per cada classe i/o mètodes del sistema.

També hi ha la carpeta *.git* on es poden veure els diferents **commits** usant un sistema de control de versions. També he pujat la pràctica en el meu servei de emmagatzemament de repositoris **GitHub**<sup>1</sup>.

Seguidament explicaré els tests per cada classe.



---

<sup>1</sup>Git : <http://www.github.com>

## 2 Explicació dels tests

### 2.1 Paquet data

#### 2.1.1 Mètodes test del paquet data

Els mètodes del paquet data serveixen per emmagatzemar dades només. Hi han quatre classes en aquest paquet que són *Vote*, *Party*, *MailAddress* i *Signature*. Aquestes classes emmagatzemen un String excepte Signature que emmagatzema un array de bytes. Per tant, els tests que he fet són similars.

```
public class VoteTest {
    @Test
    public void testToString() throws Exception {
        Vote vote = new Vote("");
        String expected = "Vote{option='' }";
        assertEquals(expected, vote.toString());
    }

    @Test
    public void testEquals() throws Exception {
        Vote one = new Vote("");
        Vote two = new Vote("");
        assertTrue(one.equals(two));
    }

    @Test
    public void testHashCode() throws Exception {
        Vote one = new Vote("");
        Vote two = new Vote("");
        assertTrue(one.hashCode() == two.hashCode());
    }
}
```

Com podem veure he fet tres tests.

- **testToString:** En aquest test creo un vot nou. Posteriorment creo un String del que espero que surti quan executi el mètode toString de la classe i posteriorment comprovo amb el mètode assertEquals si els dos Strings són iguals.
- **testEquals:** En aquest test creo dos instàncies de vot i després utilitzo el mètode equals de la classe.
- **testHashCode:** En aquest test també creo dues instàncies de vot i posteriorment comprovo que el hashCode siguin idèntics.

En el mètode Signature, la diferència és que al ser un array de bytes he de passar el string a un array de bytes cridant la funció *GetBytes()*.

## 2.2 Paquet Kiosk

### 2.2.1 Classe Activate Emission Test

```
public class ActivateEmissionTest {

    @Test
    public void TestGoodActivateEmission() throws Exception {
        TrueValidationServiceMock TrueValidation = new TrueValidationServiceMock();
        VotingMachine votingMachine = new VotingMachine();
        ActivationCard activationCard = new ActivationCard("Any");

        votingMachine.setValidationService(TrueValidation);
        votingMachine.activateEmission(activationCard);
        assertTrue(votingMachine.canVote());
    }

    @Test(expected = IllegalStateException.class)
    public void TestActivateEmission2() throws Exception {
        FalseValidationServiceMock falseValidation = new FalseValidationServiceMock();
        VotingMachine votingMachine = new VotingMachine();
        ActivationCard activationCard = new ActivationCard("Any");

        votingMachine.setValidationService(falseValidation);
        votingMachine.activateEmission(activationCard);
    }

    @Test(expected = IllegalStateException.class)
    public void TestActivateEmission3() throws Exception {
        TrueValidationServiceMock TrueValidation = new TrueValidationServiceMock();
        VotingMachine votingMachine = new VotingMachine();
        ActivationCard activationCard = new ActivationCard("Any");

        votingMachine.setValidationService(TrueValidation);
        votingMachine.activateEmission(activationCard);
        votingMachine.activateEmission(activationCard);
    }
}
```

En aquest test comprovaré que el mètode Activate Emission de la classe VotingMachine funcioni correctament. Mirant la implementació del mètode vaig veure convenient fer un test on funciones correctament, on el servei de validació de la targeta em donés false i on la màquina ja estigues activada. En el primer test comprovo que tot funciona correctament. Creo un doble per a que em retorni que la targeta d'activació és vàlida i activo la targeta. Després comprovo que pugui exercir el dret a vot. En els altres test primer faig que falli la validació retornant que qualsevol targeta retorni false en el doble i per tant llenci una excepció i que la màquina de votació ja estigui activa intentant activar-la dos cops.

## 2.2.2 Classe Activation Card Test

```
public class ActivationCardTest {

    private ActivationCard testingCard;

    @Before
    public void setUp() throws Exception {
        this.testingCard = new ActivationCard("00000");
    }

    @Test
    public void eraseTest() throws Exception {
        System.out.print("erase□Test");
        this.testingCard.erase();
        assertTrue(!this.testingCard.isActive());
    }

    @Test
    public void isActiveTest() throws Exception {
        System.out.print("isActive□Test");
        assertTrue(this.testingCard.isActive());
    }

    @Test
    public void getCodeTest() throws Exception {
        System.out.print("getCode□Test");
        assertEquals("00000",this.testingCard.getCode());
    }

    @Test
    public void wrongGetCodeTest() throws Exception {
        System.out.print("Wrong□getCode□Test");
        String wrong = "00001";
        assertEquals(1 , wrong.compareTo(this.testingCard.getCode()));
    }

    @Test
    public void wrongEqualsTest() throws Exception {
        System.out.print("Wrong□Equals□Test");
        assertTrue(!this.testingCard.equals(new ActivationCard("00001")));
    }

    @Test
    public void goodEqualsTest() throws Exception {
        System.out.print("Good□Equals□Test");
        assertTrue(this.testingCard.equals(new ActivationCard("00000")));
    }

}
```

En aquesta classe provaré les funcionalitats de la classe Activation Test. Comprovaré que funcionin els mètodes isActive() i erase(). Per fer-ho tinc una variable booleana en el codi que m'indica si la targeta està activa o no. Si elimino la targeta, el codi queda desactivat. Per tant, el isActive em retornarà fals i si creo la targeta i ho comprovo si està activa em retornarà true. Per fer el getCode i el equals creo una targeta i comprovo amb codis diferents i/o iguals que em doni el resultat desitjat.

### 2.2.3 Can Vote Test

```
public class CanVoteTest {
    private VotingMachine votingMachine;
    private TrueValidationServiceMock trueValidationServiceMock;
    private ActivationCard card;

    @Before
    public void setUp() throws Exception {
        this.votingMachine = new VotingMachine();
        this.trueValidationServiceMock = new TrueValidationServiceMock();
        this.card = new ActivationCard("AnyCode");
    }

    @Test
    public void TestGoodCanVote() throws Exception {
        this.votingMachine.setValidationService(trueValidationServiceMock);
        this.votingMachine.activateEmission(this.card);
        assertTrue(this.votingMachine.canVote());
    }

    @Test
    public void TestBadCanVote() throws Exception {
        assertTrue(!this.votingMachine.canVote());
    }
}
```

En aquesta classe de test comprovaré el correcte funcionament del mètode canVote de la classe VotingMachine. Per comprovar el mètode només tenim la possibilitat de comprovar si la maquina està o no activada. En cas de estar activa es podrà votar i en cas que no no es podrà votar. Per comprovar he necessitat implementar un doble que em retorni sempre true al validar la targeta. En el primer test, faig un activate emission i per tant activo la maquina i en el segon intentó votar sense haver fet un activate emission i oer tant em retorna una excepció.

### 2.2.4 Send Receipt Test

```
public class SendReceiptTest {
    private VotingMachine votingMachine;
    private MailAddress mailAddress;
    private ActivationCard card;
    private TrueValidationServiceMock trueValidation;
    private MailerServiceImpl mailerService;
    private SignatureServiceImpl signatureService;

    @Before
    public void setUp() throws Exception {
        this.votingMachine = new VotingMachine();
        this.card = new ActivationCard("AnyCode");
        this.mailAddress = new MailAddress("AnyAddress");
        this.trueValidation = new TrueValidationServiceMock();
        this.mailerService = new MailerServiceImpl();
        this.signatureService = new SignatureServiceImpl();
    }
}
```

```

    }

    @Test(expected = IllegalStateException.class)
    public void TestBadMachineSendReceipt() throws Exception {
        this.votingMachine.setValidationService(this.trueValidation);
        this.votingMachine.sendReceipt(this.mailAddress);
    }

    @Test(expected = IllegalStateException.class)
    public void TestBadHasVotedSendReceipt() throws Exception {
        this.votingMachine.setValidationService(this.trueValidation);
        this.votingMachine.activateEmission(this.card);
        this.votingMachine.sendReceipt(this.mailAddress);
    }

    @Test
    public void TestGoodSendReceipt() throws Exception {
        Vote vote = new Vote("AnyVote");
        this.votingMachine.setValidationService(this.trueValidation);
        this.votingMachine.setSignatureService(this.signatureService);
        this.votingMachine.setMailerService(this.mailerService);
        this.votingMachine.setVotesDB(new VotesDBImpl());
        this.votingMachine.setVotePrinter(new VotePrinterImpl());

        this.votingMachine.activateEmission(this.card);
        this.votingMachine.vote(vote);
        this.votingMachine.sendReceipt(this.mailAddress);
        assertEquals(this.mailAddress, this.mailerService.address);
        Signature expect = this.signatureService.sign(vote);
        assertEquals(expect, this.mailerService.signature);
        assertTrue(this.mailerService.status);
    }
}

```