Manual usuari Projecte Aflorensa

PART JAVA

Per accedir al javadoc del programa entrar a la carpeta del programa i trobarem la carpeta doc on tenim el javadoc.

En la classe aposta podrem crear apostes per paràmetres, passant-li únicament la data i que crei la aposta de manera aleatòria o copiant la combinació d'una altra aposta.

```
public class Aposta {
    /*
     * @author Antonio Florensa Moyano
     *
     * @version 1.0
     */
    private int idAposta;
    private static int idSiguiente = 1;
    private Combinacio c1;
    private Date dataSorteig;
    private int numReint;
    private boolean cobrada;
    private boolean premi;
    private boolean reint;

    // Crear aposta per parametres
    public Aposta(Combinacio comb, int numRein, Date data, Boolean cobr) {
        this.c1 = comb;
        this.idAposta = idSiguiente;
        this.idSiguiente++;
        this.numReint = numRein;
        this.dataSorteig = data;
        this.cobrada = false;
        this.premi = false;
        this.reint = false;
    }

    // Crear aposta random amb la data del sorteig
    public Aposta(Date d) {
        this.c1 = new Combinacio();
        this.idAposta = idSiguiente;
        this.idSiguiente++;
        this.dataSorteig = d;
        this.numReint = 0;
        this.cobrada = false;
        this.premi = false;
        this.reint = false;
    }
```

En la classe combinació tindrem els mètodes per crear combinacions per les apostes de manera aleatòria, amb paràmetres o fent una copia d'un altra combinació.

```
public class Combinacio {
    /*
     * @author Antonio Florensa Moyano
     *
     * @version 1.0
     */

    private int[] numCombinacio;

    public Combinacio() {
        this.numCombinacio = crearCombinacionRandom();
    }

    // Copiador
    public Combinacio(Combinacio c) {
        this.numCombinacio = c.numCombinacio;
    }

    // Per introduir el usuari 6 numeros
    public Combinacio(int num1, int num2, int num3, int num4, int num5, int num6) {
        int[] num = { num1, num2, num3, num4, num5, num6 };
        this.numCombinacio = num;
    }

    public Combinacio(int[] comb) {
        this.numCombinacio = comb;
    }
```

En la classe estadística guardarem tota la informació sobre les apostes per a cada un dels sorteigs que creem a la aplicació.

```
public class Estadistica {

    /*
     * @author Antonio Florensa Moyano
     *
     * @version 1.0
     */

    private int nombreApostes;
    private int cantPremiades;
    private int cantPremiadesReint;
    private int premiadesTres;
    private int premiadesQuatre;
    private int premiadesCinc;
    private int premiadesSis;

    public int getNombreApostes() {
        return nombreApostes;
    }

    public void setNombreApostes(int nombreApostes) {
        this.nombreApostes += 1;
    }

    public int getCantPremiades() {
        return cantPremiades;
    }

    public void setCantPremiades() {
        this.cantPremiades += 1;
    }
```

Es la classe on tindrem mes mètodes per poder accedir des de el Main per crear sortejos, apostes, i controlar les estadístiques de cada un dels sortejos, per crear sortejos podem passar-li el any, mes, dia per paràmetres, una data en concret o la data, si ha estat realitzat, el vector de apostes i si es pot apostar.

```
public class Sorteig {

    /*
     * @author Antonio Florensa Moyano
     *
     * @version 1.0
     */

    private Date dataSorteig;
    private Boolean realitzat;
    private Vector<Aposta> apostes;
    private ResultatSorteig resultat;
    private Boolean esApostable;
    private Estadistica estadistica;

    public Sorteig(int any, int mes, int dia) {
        Calendar calendari = new GregorianCalendar(any, mes - 1, dia);
        this.dataSorteig = calendari.getTime();
        this.realitzat = false;
        this.apostes = new Vector<Aposta>();
        this.esApostable = false;
        this.estadistica = new Estadistica();
    }

    public Sorteig(Date d) {
        this.dataSorteig = d;
        this.realitzat = false;
        this.apostes = new Vector<Aposta>();
        this.esApostable = false;
        this.estadistica = new Estadistica();
    }

    public Sorteig(Date data, Boolean reali, Vector<Aposta> apost, Boolean apostable) {
        this.dataSorteig = data;
        this.realitzat = reali;
        this.apostes = apost;
        this.esApostable = apostable;
        this.estadistica = new Estadistica();
    }
```

Al crea un nou resultat ens tornara ja un resultat fet amb una combinació, un numero complementari i un numero de reintegrament.

```
public class ResultatSorteig {

    /*
     * @author Antonio Florensa Moyano
     *
     * @version 1.0
     */

    private Combinacio combina;
    private int numComplementari;
    private int numReintegrament;

    public ResultatSorteig() {
        this.combina = new Combinacio();
        this.numComplementari = crearNumCompl(this.combina);
        this.numReintegrament = (int) (Math.random() * 9) + 0;
    }

    public int crearNumCompl(Combinacio com) {
        int[] nums = com.getNumCombinacio();
        boolean band = false;
        int num = 0;
        while (band == false) {
            num = (int) (Math.random() * 49) + 1;
            if (com.conteNum(num, nums) == false) {
                band = true;
            }
        }
        return num;
    }
```

Al programa principal es on tindrem tots els mètodes per crear Sortejos, Apostes amb combinacions , realitzar sortejos, consultar si aquestes apostes han estat premiades amb algun premi i les estadístiques del sortejos realitzats. El programa ens crea un Sorteig per setmana, els dilluns de cada setmana(per defecte ja tenim 2 sortejos creats i sol podem apostar al primer).

```
 * @author Antonio Florensa Moyano
 *
 * @version 1.0
 */

public static void main(String[] args) {

    // Inici de programa on crearem 2 sorteigs.
    Vector<Sorteig> sorteigs = new Vector<Sorteig>();
    sorteigs = iniciarAmbSorteigs(sorteigs);
    Scanner sca = new Scanner(System.in);
    Boolean band = false;
    int leer = -1;

    do {

        System.out.println("1.- Realitzar sorteig d'aquesta setmana");
        System.out.println("2.- Consultar data sorteig en concret");
        System.out.println("3.- Afegir sorteig (Per data)");
        System.out.println("4.- Consultar data dels sorteigs als que es pot apostar");
        System.out.println("5.- Realitzar aposta random (Introduir Data)");
        System.out.println("6.- Realitzar aposta amb combinacio (Introduir Data i combinacio)");
        System.out.println("7.- Realitzar aposta amb combinacio anterior (Introduir id Aposta, Data i combinacio)");
        System.out.println("8.- Comprovar premi (Introduir id Aposta i Data)");
        System.out.println("9.- Comprovar reintegrament (Introduir id Aposta i Data)");
        System.out.println("10.- Cobrar aposta guanyadora. (Introduir id Aposta i Data)");
        System.out.println("11.- Comprovar si s'ha cobrat la aposta. (Introduir id Aposta i Data)");
        System.out.println(
                "12.- Consultar el nombre d'apostes, la quantitat d'apostes premiades amb cada un del diferent
                        + "premis i la quantitat d'apostes amb reintegrament de tots els sorteigs que s'han r
        System.out.println("0.- Exit");

        leer = sca.nextInt();

        switch (leer) {
        case 1:
            for (Sorteig s : sorteigs) {
                if (s.getRealitzat() == false && s.getEsApostable() && band == false) {
                    ResultatSorteig r1 = realitzarSorteig(s);
                    s.setRealitzat(true);
                    System.out.println(r1.toString());
```

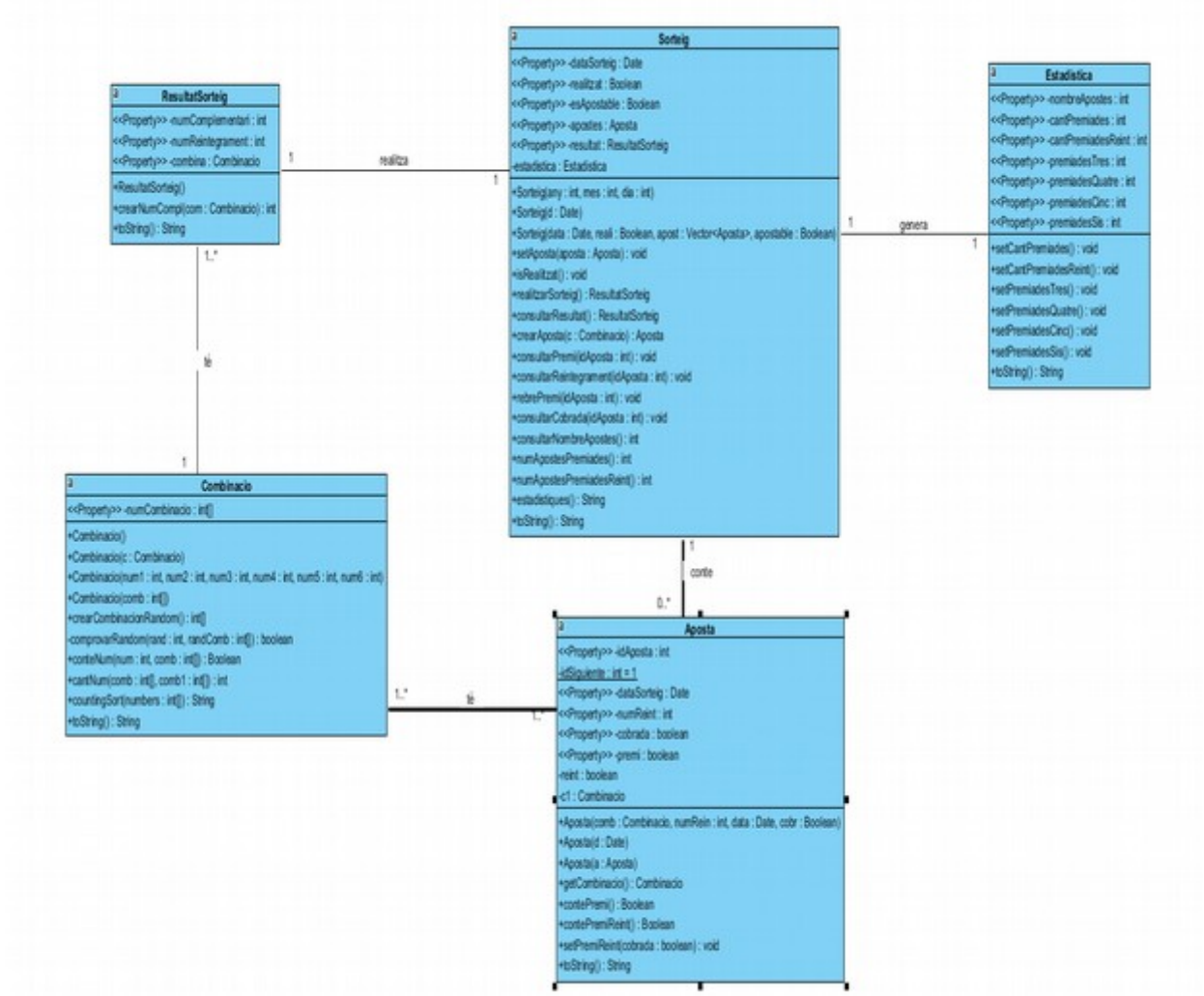
Al executar el programa seleccionarem amb el numero corresponent al mètode que vulguem accedir i ens entrara al mètode, un cop dintre ens dirà si hem d'introduir algun input com la Data(dia/mes/any) , la ID de la nostra aposta o una combinació numero per numero.

```
1.- Realitzar sorteig d'aquesta setmana
2.- Consultar data sorteig en concret
3.- Afegir sorteig (Per data)
4.- Consultar data dels sorteigs als que es pot apostar
5.- Realitzar aposta random (Introduir Data)
6.- Realitzar aposta amb combinacio (Introduir Data i combinacio)
7.- Realitzar aposta amb combinacio anterior (Introduir id Aposta, Data i combinacio)
8.- Comprovar premi (Introduir id Aposta i Data)
9.- Comprovar reintegrament (Introduir id Aposta i Data)
10.- Cobrar aposta guanyadora. (Introduir id Aposta i Data)
11.- Comprovar si s'ha cobrat la aposta. (Introduir id Aposta i Data)
12.- Consultar el nombre d'apostes, la quantitat d'apostes premiades amb cada un del diferents
premis i la quantitat d'apostes amb reintegrament de tots els sorteigs que s'han realitzat
0.- Exit
```

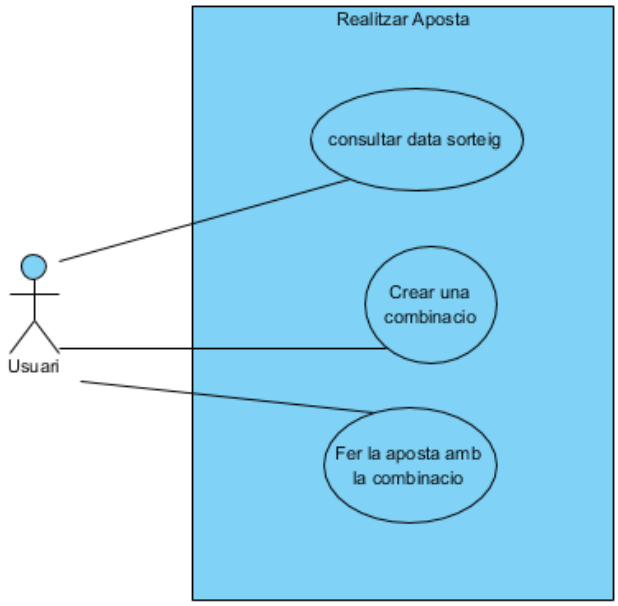
PART UML JAVA

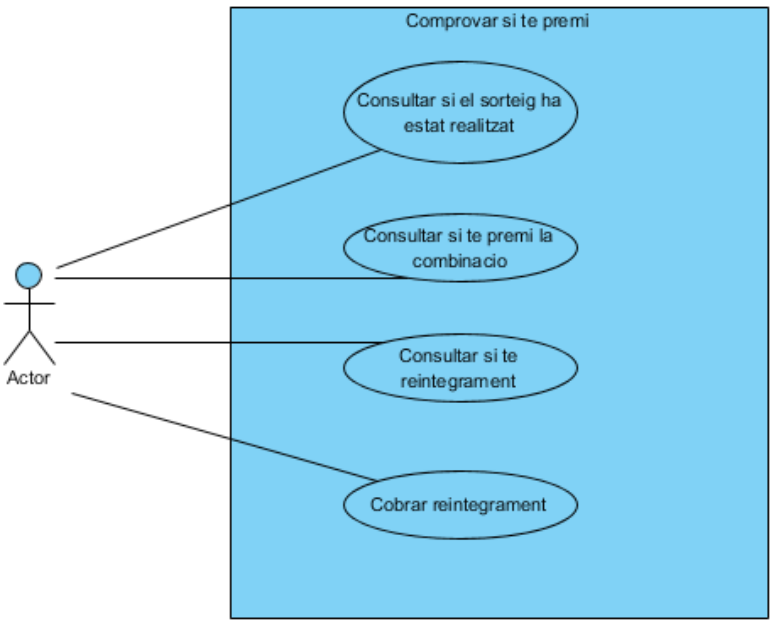
Model Conceptual

(Si no es veu be al projecte de java hi ha totes les imatges de la part del UML)



Model de casos d'ús.





Diagrames de seqüència.

Diagrama de cobrarPremi.

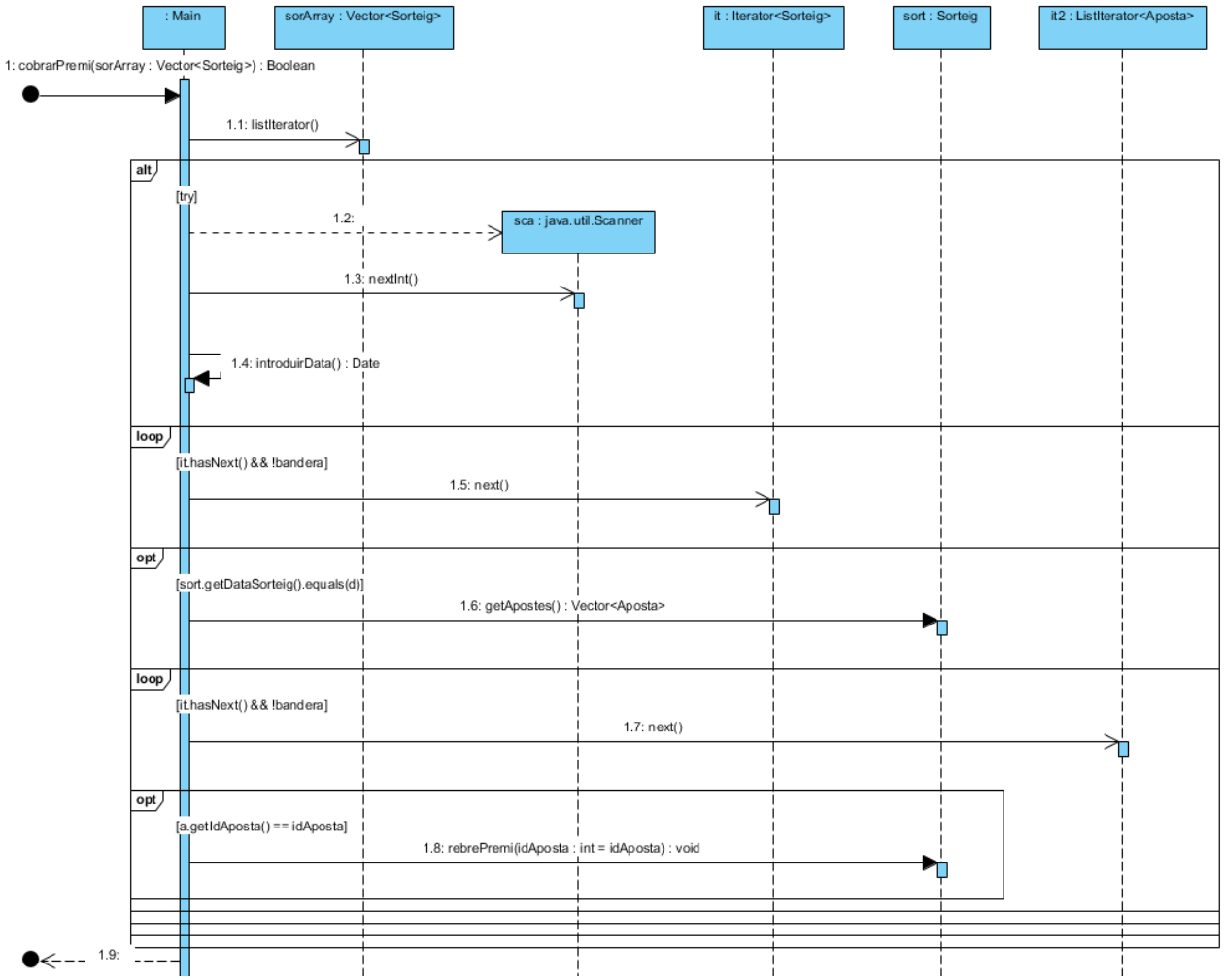


Diagrama de apostarCombinacio()

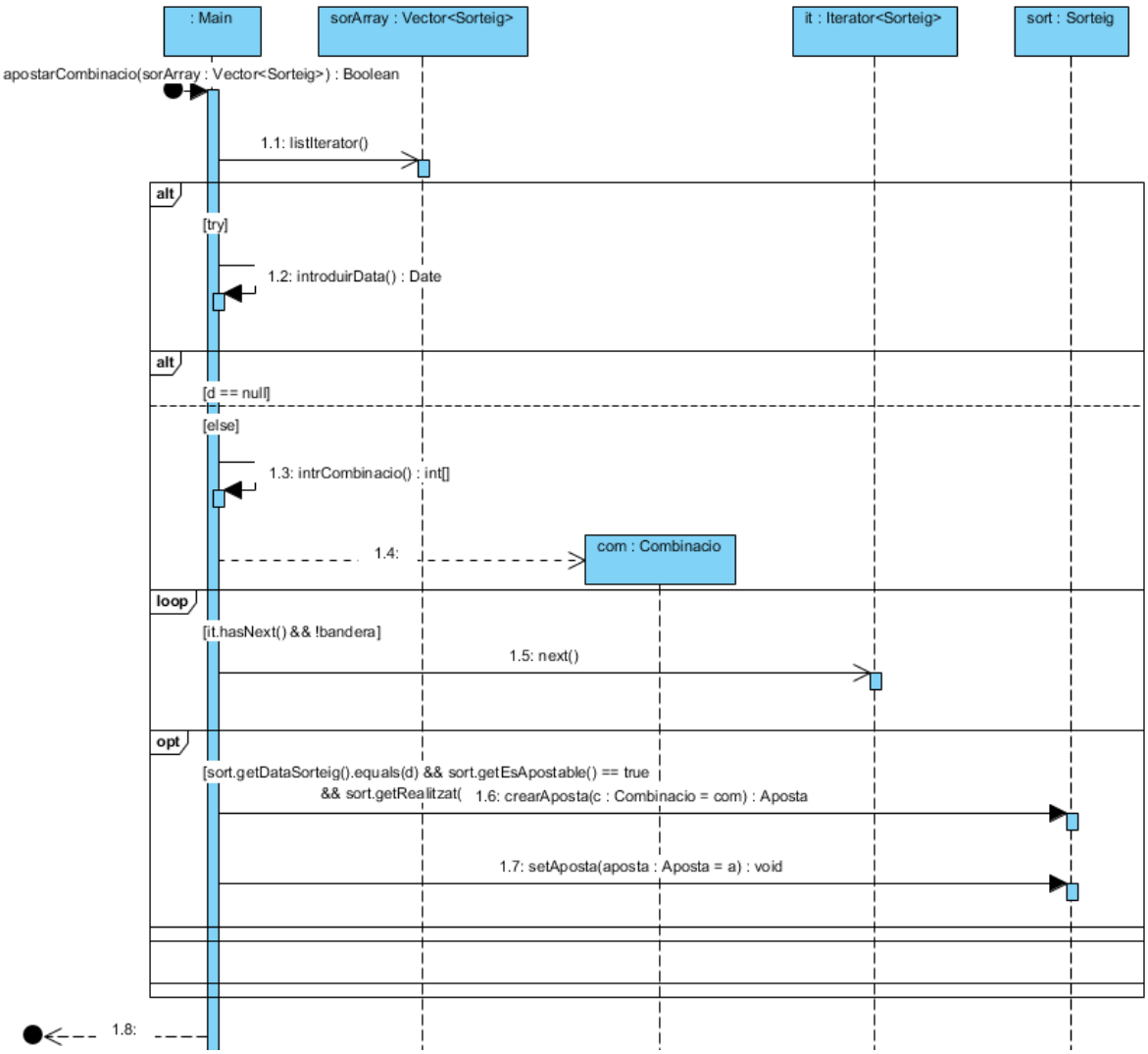
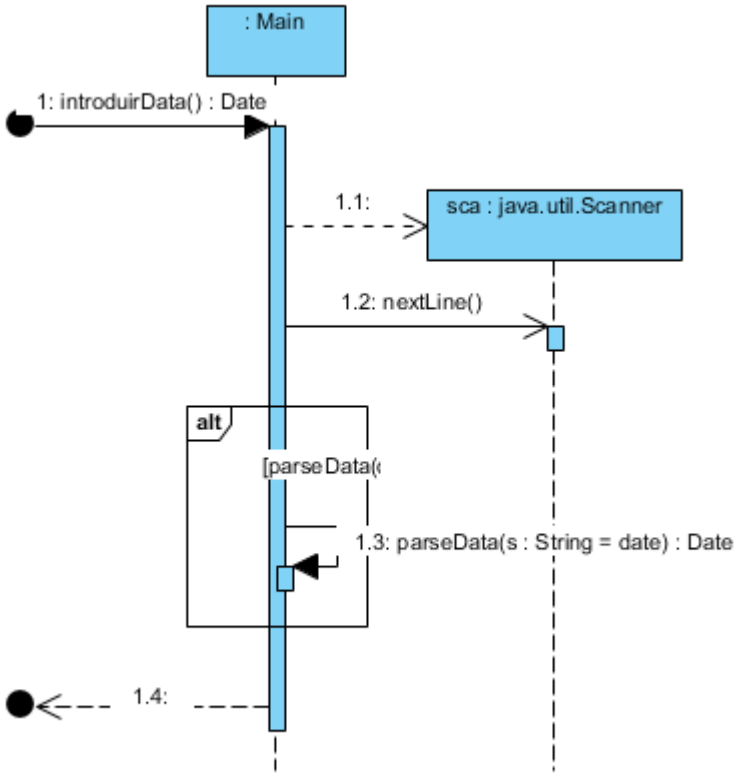


Diagrama d’introduir data

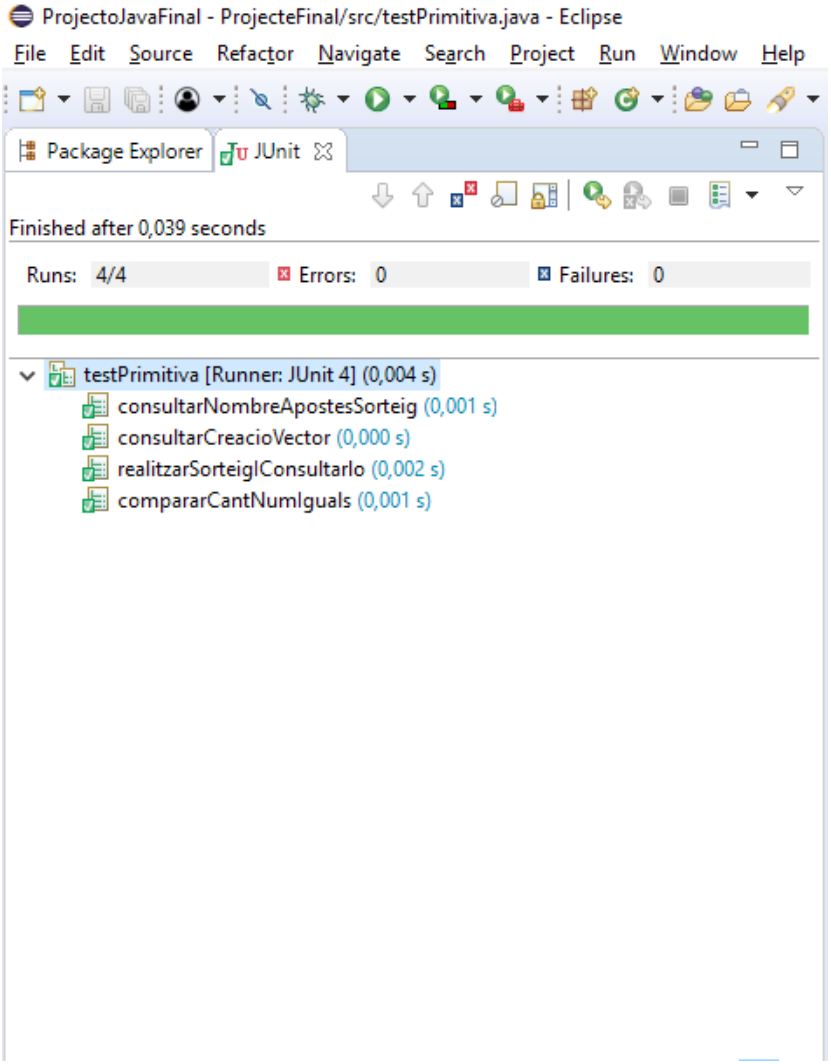


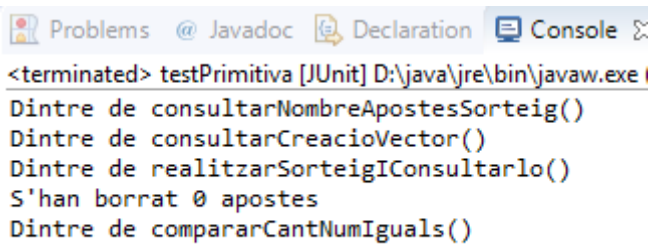
PART JUNIT JAVA

Classe testPrimitiva on tindrem els tests creats per executar-los amb junit, comprovarem que es puguin crear apostes, combinacions i Sorteigs.

```
public class testPrimitiva {

    Combinacio c = new Combinacio();
    Sorteig s = new Sorteig(Calendar.getInstance().getTime());
    Aposta a = new Aposta(c,3,s.getDataSorteig());


    //Comparacio de 2 combinacions per obtenir la cantitat de numeros iguals

    @Test
       public void compararCantNumIguals() {
          System.out.println("Dintre de compararCantNumIguals()");
          int numero = 6;
          Combinacio c2 = new Combinacio(c);
          assertEquals(numero, c.cantNum(c.getNumCombinacio(), c2.getNumCombinacio()));
       }

    //Consultar cuantes apostes te el Sorteig

    @Test
       public void consultarNombreApostesSorteig() {
          System.out.println("Dintre de consultarNombreApostesSorteig()");
          int numero = 1;
          s.setAposta(s.crearAposta(c));
          assertEquals(numero, s.consultarNombreApostes());
       }

     //Consultar que el Vector de apostes no es null cuan el creem sense cap aposta.

    @Test
       public void consultarCreacioVector() {
          System.out.println("Dintre de consultarCreacioVector()");
          assertNotNull("No es null", s.getApostes());
       }

    //Consultar si al realitzar el sorteig i consultarlo ens retorna el mateix.

    @Test
       public void realitzarSorteigIConsultarlo() {
          System.out.println("Dintre de realitzarSorteigIConsultarlo()");
         ResultatSorteig s3 = s.realitzarSorteig();
         ResultatSorteig s4 = s.getResultat();

          assertEquals(s4,s3);
       }
```

Execució de les proves amb jUnit





PART ANDROID STUDIO GPS

La aplicació constara de 3 botons i un spinner.

- En el primer boto (Cambiar coordenadas random) es on generarem les 2 coordenades aleatories i les marcarem con les que es s'han de comparar amb la posició actual al presionar el boto.

- En el segon boto (Comparar coordenadas) es on compararem les 2 coordenades aleatories actives amb la nostra posició actual.

- En el tercer actualitzarem i mostrarem el spinner que es on tenim la informació que es guarda a la base de dades.

- Al spinner al obrir la aplicació es carregara la informació que ens interessa mostrar de la base de dades.

PART GITHUB

https://github.com/aflorensa/projecte