LOTERIA DE NAVIDAD

-Ficha técnica-

Xavier García
Biel Palomar
Óscar Zapatero
Kevin Burga

ÍNDICE

Resumen del proyecto

El documento redacta el informe técnico del proyecto del grupo FACHERITOS compuesto por: Xavier García, Biel Palomar, Óscar Zapatero y Kevin Burga. El proyecto ha sido desarrollado en las plataformas de NetBeans y GitHub utilizando el lenguaje java para programar el proyecto.

El proyecto consiste en simular el sorteo de la lotería de navidad, donde el cliente que haya comprado un boleto pueda preguntarle al programa si su boleto ha sido premiado o le corresponde otra cantidad por cercanía a premios, entre diferentes premios generados automáticamente.

Para poder trabajar en grupo de manera ordenada, y con el fin de poder preservar el código y otros elementos del proyecto, hemos utilizado la plataforma GitHub para poder guardar nuestro progreso, y así también tener una mayor facilidad para poder acceder al contenido para modificar o documentar el código.

A continuación se detallarán los diferentes contenidos, avances y retrasos que hemos tenido durante el proyecto.

Resumen de ampliación del proyecto

El documento redacta el informe técnico del proyecto del grupo FACHERITOS compuesto por: Biel Palomar, Óscar Zapatero y Kevin Burga. El proyecto ha sido desarrollado en las plataformas de NetBeans y GitHub utilizando el lenguaje java para programar el proyecto.

La ampliación consiste en modificar el proyecto con el fin de que el cliente pueda comprobar sorteos de años anteriores, establecer un idioma predefinido, y listar sus premios en peñas de diferentes usuarios basándose en el dinero puesto por cada uno de ellos.

Diario de desarrollo

Dia 12/12/2022 - 16/12/2022

- Entendimiento del proyecto.
- Creación de repositorio en GitHub grupal.
- Creación de grupo de Whatsapp para comunicarnos.

Dia 19/12/2022 - 13/01/2023

- Búsqueda de información para el desarrollo lógico del programa.
- Creación de fichero de los datos necesarios para el proyecto.
- Inicialización de un nuevo proyecto en Java gestionado en Github.
- Desarrollo del programa de manera no modular.
- Creación del sorteo
- Funciones básicas para facilitar la lectura.

Dia 16/01/2023 - 20/01/2023

- Creación de una ficha técnica para recoger la información del proyecto a nivel lógico y técnico.
- Finalización del apartado funcional del programa.
- Comprobación de los cupones

Dia 23/01/2023 - 27/01/2023

- Reestructuración en funciones de todo el programa.
- Creación de constantes útiles.
- Redacción de la ficha técnica.

Dia 30/01/2023 - 3/02/2023

- Documentación del código y comentado.
- Se descarta la idea de la compra de boletos.
- Finalización de la ficha técnica.
- Reunión para defensa y entendimiento del código.
- Casos a probar la funcionalidad del código.

Dia 13/02/2023 - 17/02/2023

- Mejora del código previo.
- Desarrollo inicial de la modificación de idiomas.
- Desarrollo inicial de la selección del sorteo.

Dia 20/02/2023 - 24/02/2023

- Desarrollo inicial del menú de peñas.
- Finalización de la estructura básica de los idiomas

Dia 27/02/2023 - 03/03/2023

- Desarrollo completo de todos los programas y expansiones planteadas.
- Refactorización del código.
- Inclusión de nuevos casos de prueba
- Documentación y comentado del código.
- Redacción de la ficha técnica.

Retrasos

- Uso de una ficha técnica para documentar el proceso de desarrollo del proyecto:
  Previsto: periodo 19/12/22 - 13/01/23
  Efectuado: periodo 16/01/23 - 20/01/23

- Comentado del código para un mejor entendimiento general:
  Previsto: semanalmente
  Efectuado: periodo 16/01/23 - 20/01/23

- Casos de pruebas para ver el funcionamiento del código.
  Previsto: 22/01/23 - 25/01/23
  Efectuado: Indeterminado

- Implementacion de test.
  Previsto: 30/01/23 - 02/02/23
  Efectuado: Indeterminado

- Entendimiento de los ficheros binarios y textos.
  Previsto: periodo 30/01/2023 -10/02/2023
  Efectuado: periodo 13/02/2023 - 17/02/2023

- Comienzo de la segunda fase del proyecto
  Previsto: 6/02/23 - 10/02/23
  Efectuado: 13/02/23 - 17/02/23

Funcionalidad

- Escoger un idioma para las interfaces del programa.

```
public static String EscogerIdioma() {
    String resultat = "";
    String[] opcions_menu = {"Català", "Castellano", "English", "Deustch", "Português", "Italiano", "Chino"};
    int seleccio = Utils.MenuBucle( array_opcions: opcions_menu,  mensaje: "Opció/Opción/Option/Opção/Opzione/選項: ");
    resultat = GestionMenuIdioma(seleccio);
    return resultat;
}
```

- Sorteo de premios
  - Escoger un premio nuevo o creado anteriormente
  -

```
public static NumPremiado[] tipoLoteria(String[] tipos) throws FileNotFoundException, IOException {
    boolean noArray = true;
    NumPremiado[] arrayPremios = new NumPremiado[QUANTITATPREMIS];

    while (noArray) {
        System.out.println( x: RetornarLinia( nomfitxer: IDIOMA,  numlinia: VOLEUCONSULTAR));
        int opcion = MenuBucle( array_opcions: tipos,  mensaje: RetornarLinia( nomfitxer: IDIOMA,  numlinia: OPCIONSELECCIONADA));
        noArray = ComprobarIndexVacio();
        if (opcion == 1) {

            MostrarAnyos(); //Muestra los años guardados
            arrayPremios = Sorteo();
            BubbleSortPremis( array: arrayPremios);
            year = IntroducirAnyo(); //Añades una nuevo sorteo al fichero, requerido un año no repetido
            while (!ComprobarValidezAnyo( anyo: year)) {
                year = IntroducirAnyo();
            }
            ComprobarValidezAnyo( anyo: year);
            GuardarDatos( premios: arrayPremios,  codigo: year);
            noArray = false;
        } else if (opcion == 2 && !noArray) {
            MostrarAnyos(); //Muestra los años guardados
            year = IntroducirAnyo();
            while (ComprobarValidezAnyo( anyo: year)) {
                year = IntroducirAnyo();
            }
            long posicion = BuscarPosicionIndice( anyo: year);
            if (posicion != -1) {
                arrayPremios = ExtraerDatos(posicion);
            }
        }
    }
```

  - Generación de un bombo de premios

```
public static int[] CrearBomboPremios() {
    final int GORDO = 4000000;
    final int SEGONPREMI = 1200000;
    final int TERCERPREMI = 500000;
    final int QUARTPREMI = 200000;
    final int QUINTOPREMI = 60000;
    final int PEDREADA = 1000;
    final int QUANTITATPREMIS = 1807;

    int[] bomboPremios = new int[QUANTITATPREMIS];
    for (int i = 0; i < bomboPremios.length; i++) {
        if (i < quantPremis) {
            for (int j = 0; j < quantGordo; j++) {
                bomboPremios[i] = GORDO;
                i++;
            }
            for (int j = 0; j < quantSegund; j++) {
                bomboPremios[i] = SEGONPREMI;
                i++;
            }
            for (int j = 0; j < quantTercer; j++) {
                bomboPremios[i] = TERCERPREMI;
                i++;
            }
            for (int j = 0; j < quantCuart; j++) {
```

- Generación de números (no repetidos) asignados a cada premio del bombo. Creación de un número premiado extrayendo una bola del bombo de premios.

```
public static NumPremiado[] Sorteo() { //Cada premio
    int[] bomboPremios = CrearBomboPremios();
    NumPremiado[] numeros_premiats = new NumPremiado[1807];
    int num_afegir = 0;
    int premi_afegir, posP;
    for (int i = 0; i < numeros_premiats.length; i++) { //Crear todos los premios
        boolean repeatnum = true;
        while (repeatnum) {
            num_afegir = rnd.nextInt( bound:100000); //Generar número aleatorio
            repeatnum = false;
            for (int j = 0; j < i && !repeatnum; j++) {//Comprobar que el número
                if (num_afegir == numeros_premiats[j].numero) {
                    repeatnum = true;
                }
            }
        }

        //Escoger número del bombo
        posP = rnd.nextInt(bomboPremios.length - i);
        premi_afegir = bomboPremios[posP]; //Escoger premio del bombo
        //Crear premio a partir de el número y premio sacados de los bombos
        numeros_premiats[i] = new NumPremiado();
        numeros_premiats[i].numero = num_afegir;
        numeros_premiats[i].premio = premi_afegir;
```

```
// </editor-fold>
/**
 * Clase de números premiados
 */
public static class NumPremiado {

    int numero;
    int premio;
}
```

- Muestra de los premios. Ordenación previa de los premios por el método de la burbuja.

```
/**
 * Función para mostrar el listado completo de números premiados
 * @param num_premi array de número premiados
 */
public static void MostrarPremios(NumPremiado[] num_premi) {
    String numeroS = "";
    System.out.println( x:"\nLoteria de Navidad");
    System.out.println( x:"*****************");
    for (int i = 0; i < num_premi.length; i++) {
        numeroS = "" + num_premi[i].numero;
        //Añadir ceros a los números de menos de 5 cifras
        while (numeroS.length() < 5) {
            numeroS = "0" + numeroS;
        }
        System.out.println((i + 1) + ". Número: " + numeroS + " Premio: " + num_premi[i].premio);

    }
}
```

```
Qué deseas consultar:
1) Revisar premio de cupón
2) Consultar todos los premios
3) Sortir
Opción seleccionada: 2

Loteria de Navidad
****************
1. Número: 73891 Premio: 4000000
2. Número: 62435 Premio: 1200000
3. Número: 44355 Premio: 500000
4. Número: 95769 Premio: 200000
5. Número: 80150 Premio: 200000
6. Número: 88179 Premio: 60000
7. Número: 26229 Premio: 60000
8. Número: 08347 Premio: 60000
9. Número: 04506 Premio: 60000
10. Número: 51934 Premio: 60000
11. Número: 53646 Premio: 60000
12. Número: 85255 Premio: 60000
```

- Comprobación de premios
    - Introducción de un número de cupón

```
Qué deseas consultar:
1) Revisar premio de cupón
2) Consultar todos los premios
3) Sortir
Opción seleccionada: 1
Introduce tu número: 23517
```

    - Comprobación de premios

```java
public static int ComprobarPremio(int cupon, NumPremiado[] premiados) {
    //Creamos la variable premio
    int premio = 0;

    //Miramos si ha tocado algun premio principal
    premio = ComprobarBombo(cupon, premiados);

    //Si no ha tocado miramos la aproximaión
    if (premio == 0) {
        premio = ComprobarAproximacion(cupon, premiados);
    }
    //Si sigue sin tocar nada miramos la centena
    if (premio == 0) {
        premio = ComprobarCentena(cupon, premiados);
    }
    //Si sigue sin tocar nada miramos las 2 últimas cifras del cupón
    if (premio == 0) {
        premio = ComprobarUltimas(cupon, premiados);
    }
    //Si sigue sin tocar nada miramos la pedrea
    if (premio == 0){
        premio = ComprobarBombo2(cupon, premiados);
    }
    //Por último miramos si se lleva el reintegro
    premio += ComprobarReintegro(cupon, premiados, premioactual:premio);

    //Devolvemos el valor premiado
    return premio;
}
```

1. Comprobación de premios principales. Se revisa si el número corresponde al gordo, segundo premio, tercer premio, cuarto premio o quinto premio.

```java
public static int ComprobarBombo(int cupon, NumPremiado[] premiados) {
    /*Creamos la constante con al cantidad de numeros mayores que hay - 1
    1 primer premio
    1 segundo premio
    1 terceer premio
    2 cuartos premios
    8 quintos premios
    */
    final int TOTALGUANYADORS = 12;
    //Creamos las variables del programa
    boolean ganador = false;
    int premio = 0;

    //Mirmos si el cupon corresponde con alguno de estos premios y le assignamos el premio si és asi
    for (int i = 0; i <= TOTALGUANYADORS && ganador == false; i++) {
        if (cupon == premiados[i].numero) {
            premio = premiados[i].premio;
            ganador = true;
            nompremi = nombrePremiado(premio, premiados);
        }
    }

    //Devolvemos el valor de premio
    return premio;
}
```

2. Comprobación de aproximación a un premio principal. Si el número de cupón sobrepasa por una unidad o queda una unidad por debajo de uno de los tres premios principales.

```java
public static int ComprobarAproximacion(int cupon, NumPremiado[] premiados) {
    //Creamos las constantes con los valores de los premios
    final int PREMIAPROX3 = 9600;
    final int PREMIAPROX2 = 12500;
    final int PREMIAPROX1 = 20000;

    //Creamos la avriable a devolver
    int premio = 0;
    //Miramos si el cupon se aproxima al primer premio
    if (cupon == premiados[0].premio - 1 || cupon == premiados[0].premio + 1) {
        premio = PREMIAPROX1;
        nompremi = "la aproximación al gordo";
    //Miramos si el cupon se aproxima al 2do premio
    } else if (cupon == premiados[1].premio - 1 || cupon == premiados[1].premio + 1) {
        premio = PREMIAPROX2;
        nompremi = "la aproximación al 2do premio";
    //Miramos si el cupon se aproxima al 3er premio
    } else if (cupon == premiados[2].premio - 1 || cupon == premiados[2].premio + 1) {
        premio = PREMIAPROX3;
        nompremi = "la aproximación al 3er premio";
    }
    //Devolvemos el valor de premio
    return premio;
}
```

3. Comprobación de la centena de un premio principal. Si el número de cupón forma parte de la misma centena que uno de los cuatro premios principales.

```java
public static int ComprobarCentena(int cupon, NumPremiado[] premiados) {
    //Creamos la constante del valor del premio
    final int PREMICENTENA = 1000;
    //Creamos la variable a devolver
    int premio = 0;
    //Comprovamos si la centena correspo0nde a alguna de ls 4 premios mayores
    for (int i = 0; i <= 4; i++) {
        if (cupon >= (premiados[i].numero / 100) * 100 && cupon <= (premiados[i].numero / 100) * 100) {
            premio = PREMICENTENA;
            nompremi = "el premio a la centena";
        }
    }

    //Devolvemos el valor de premio
    return premio;
}
```

4. Comprobación de las dos últimas cifras de un premio principal. Si el número de cupón comparte sus dos últimas cifras con uno de los tres premios principales.

```java
public static int ComprobarUltimas(int cupon, NumPremiado[] premiados) {
    //Creamos la constante del premio a las 2 útlimas cifras
    final int PREMI2ULTIMAS = 1000;

    //Cramos la variablre premio
    int premio = 0;
    //Creamos una variable con las 2 últimas cifras de cupón
    int ultimas2 = cupon % 100;
    //Miramos si coinciden las 2 últimas cifras con las 2 ultimas de un premio mayor
    for (int i = 0; i <= 2; i++) {
        if (ultimas2 == premiados[i].numero % 100) {
            premio = PREMI2ULTIMAS;
            nompremi = "las 2 últimas cifras de un premio mayor";
        }
    }

    //Devolvemos el premio
    return premio;
}
```

5. Comprobación de pedreada. Si el número **premiado** es cualquier otro número de los 1807 ganadores no perteneciente a los premios principales.

```java
public static int ComprobarBombo2(int cupon, NumPremiado[] premiados) {
    //Creamos la constante de todos los premios mayores que hay - 1
    final int TOTALGUANYADORS = 12;
    //Creamos las variable del programa
    boolean ganador = false;
    int premio = 0;

    //Miramos si el cupon corresponde a la pedrea
    for (int i = TOTALGUANYADORS; i < premiados.length && ganador == false; i++) {
        if (cupon == premiados[i].numero) {
            premio = premiados[i].premio;
            ganador = true;
            nompremi = "la pedrea";
        }
    }

    //Devolvemos el premio
    return premio;
}
```

6. Comprobación de última cifra del gordo (reintegro). Como extra, si el número de cupón, sea premiado o no, salvo el gordo, comparte su última cifra con el gordo, obtiene el reintegro.

```java
public static int ComprobarReintegro(int cupon, NumPremiado[] premiados, int premioactual) {
    //Creeamos la constante con el valor del premio
    final int PREMIREINTEGRO = 200;
    //Creamos la variable premio
    int premio = 0;
    //Miramos si el útlimo numero coincide con el último numero del gordo
    if (cupon % 10 == premiados[0].numero % 10) {
        //Miramos que el numero no sea el gordo
        if (cupon != premiados[0].numero) {
            //Assignamos el valor de la constante a premio
            premio = PREMIREINTEGRO;
            //Si todavía no habia ningún premio ponemos la vairable nompremi con reintegro
            if (premioactual == 0) {
                nompremi = "el reintegro";
            }
        }
    }
    //Devlvemos la variable premio
    return premio;
}
```

- Comprobación:

```
Opción seleccionada: 1
Introduce tu número: 23517
Has ganado la pedrea con una cantidad de 100 al décimo
```

- Uso de peñas y grupos de usuarios para compartir premios

```java
/** Procediment per a Crear una colla ...6 lines */
public static void CrearColla(NumPremiado[] premiados) throws FileNotFoundException, IOException {...11 lines }

/** Procediment per a determinar el numero de colla i la posicio al fitxer ...7 lines */
public static void EscribirColla(Colla colla) {...18 lines }

/** Procediment per escriure al fitxer index les dades de la colla que ...7 lines */
public static void EscriureIndex(indice2 id, RandomAccessFile raf) {...10 lines }

/** Procediment per a escriure les dades de la colla al fitxer de collas ...7 lines */
public static void EscriureDades(Colla colla, RandomAccessFile raf, long codi) {...13 lines }

/** Funcio que retorna un Usuari que s'ha llegit del fitxer usuaris ...6 lines */
public static Usuari LlegirUsuari(RandomAccessFile raf) {...17 lines }

/** Funció per a afegir un usuari a una colla ...7 lines */
public static int AfegirUsuari(Colla colla, NumPremiado[] premiados) throws IOException {...5 lines }

/** Procediment que serveix per a escriure un usuair al punt que se s'ha ...8 lines */
public static void EscriureUsuariP(Usuari usr, RandomAccessFile raf) {...16 lines }

/** Procediment per a escriure un usuari al final del fitxer ...5 lines */
public static void EscriureUsuari(Usuari usr) {...19 lines }

/** Funció per a tancar el RandomAccessFile ...5 lines */
public static void CerrarRAF(RandomAccessFile raf) {...7 lines }

/** Funció que demana al client les dades corresponents per a crear un usuari ...8 lines */
public static Usuari DemanarDadesUsuari(Colla colla, NumPremiado[] premiados) throws IOException {...14 lines }
```

```
public static void SubmenuColles(NumPremiado[] premiados) throws FileNotFoundException, IOException {...45 lines }

public static void ImprimirUsuariEspecific(long codi, String nom) throws FileNotFoundException, IOException {...11 lines }

/** Imprimir por pantalla los datos del parámetro u ...6 lines */
public static void EscribirDatosUsr(Usuari u) throws IOException {...11 lines }

/** Mostrar Usuarios borrados de una peña especifica ...5 lines */
public static void MostrarBorrados(long codi) {...34 lines }

/** Procediment per a esborrar un usuari d'una colla ...6 lines */
public static void EsborrarUsuari(long codi, String nom) {...32 lines }

/** Funció per a esborrar un usuari especific ...14 lines */
public static long EsborrarUsuariEspecific(RandomAccessFile raf3, String nom, Colla colla, long posicio, long posiciocolla, indice2 id, RandomAccessFile raf) throws IOException {...19 l
/** Recuperar usuarios borrados ...5 lines */
public static void RecuperarUsuaris(String nom, long codi) {...34 lines }
/** Buscar usuario para su recuperación ...13 lines */
public static long RecuperarUsuarioEspecifico(Usuari usr, String nom, Colla colla, RandomAccessFile raf3, long posicio, long posiciocolla, indice2 id, RandomAccessFile raf) throws IOExc

/** Procediment per a modificar un usuari especific ...7 lines */
public static void ModificarUsuaris(long codi, NumPremiado[] premiados, String nom) {...17 lines }

/** Procediment per a iterar per diferents Indexos de colles ...12 lines */
public static void IterarId(indice2 id, long codi, RandomAccessFile raf2, RandomAccessFile raf3, String nom, NumPremiado[] premiados, RandomAccessFile raf) throws IOException {...15 lin

/** Procediment per a iterar usuaris fins trobar el que s'ha de esborrar ...11 lines */
public static void IteracioUsuari(Usuari usr, String nom, Colla colla, NumPremiado[] premiados, RandomAccessFile raf3, long posicio) throws IOException {...15 lines }
```

Contenido del proyecto

- Proyecto java Lotería de Navidad
    - Código ejecutable
    - Tests de casos de prueba
    - JavaDoc
- Ficha técnica
- Sumario de funcionamiento de premios de la lotería
- Autoevaluaciones de cada miembro del equipo.
- Coevaluaciones de otros miembros.

Infografía

Premios de la loteria: https://www.loterias.com/loteria-de-navidad/como-funciona

En la versión del programa 1.0. El Bombo principal cuenta con un total de 1807 premios, subdivididos en un premio principal de 4.000.000€, un segundo premio de 1.500.000€, un tercer premio de 500.000€, dos cuartos premios de 200.000€ cada uno, ocho quintos premios de 60.000€ por premio, y todos los premios restantes son pedreadas de 1.000€ cada una.

En el caso de aproximarse una unidad por encima o por debajo de un premio primero, segundo o tercero, el cupón ganará 20.000€, 12.500€ y 9.600€ correspondientemente como premio por acercamiento.

Los cupones que entren en la centena de un premio primero, segundo, tercero o cuarto, o aquellos que compartan sus dos últimas cifras con el primer premio, el segundo y el tercero, obtendrán un premio de 1.000€.

Como compensación adicional, todo premio, salvo el gordo, que comparta su última cifra con la cifra final del Gordo, obtendrá un plus de 200€.