

# PROYECTO FINAL

---

KAROL STEFANY ORDOÑEZ PEÑALOZA  
SARA XIMENA CONTRERAS QUIROGA

17/11/2023

PROGRAMACION ORIENTADA A OBJETOS

# INTRODUCCIÓN

Bienvenidos a nuestro proyecto de programación, donde la creatividad y la destreza se unen en un juego interactivo diseñado para desafiar y entretener. En esta aventura digital, nos sumergimos en el fascinante mundo de la programación de juegos utilizando Java y JFrame como nuestras herramientas principales.

## ENUNCIADO DEL PROBLEMA

Este proyecto tiene como objetivo principal la concepción y ejecución de un sistema de juego de cartas en el entorno de programación Java. La tarea ya ha sido abordada en la guía número 4, donde se llevó a cabo el diseño y la presentación del proyecto. Ahora, el siguiente paso es la implementación efectiva del sistema.

El sistema de juego de cartas debe contar con la capacidad de generar una baraja de cartas, realizar operaciones de mezcla y distribuir las cartas a los jugadores de manera adecuada. Además, se espera que el sistema sea capaz de simular un juego de cartas completo, brindando a los jugadores la posibilidad de jugar cartas, visualizar sus manos y acceder a otras funcionalidades pertinentes que hayan sido destacadas durante las exposiciones anteriores.

La implementación exitosa de este proyecto no solo implica la creación de un sistema funcional, sino también la incorporación de las características esenciales para garantizar una experiencia de juego completa y satisfactoria.

## PREGUNTAS ORIENTADORAS

- ¿Cómo implementarán la creación de una baraja de cartas con los diferentes tipos y valores?

Creamos diferentes interfaces en las cuales se puedan ver los 4 tipos de carta y cada uno de sus textos correspondientes, por ejemplo, la carta normal contaba con 30 retos, la carta bomba tenía 15 reglas, la carta comodín 10 ayudas y la carta dados 7 juegos.

- ¿Qué estructura de datos utilizarán para almacenar las cartas en la baraja? ¿Cómo implementarán la función de mezclar la baraja?

Usamos vectores para almacenar los datos y a su vez matrices en las cuales pusimos un random para que se mezclara la baraja y mostrara las cartas aleatorias.

- ¿Qué algoritmo de mezcla utilizarán?

Usamos las matrices, con este algoritmo fue más sencillo y práctico a la hora de programar la muestra de datos tan grandes, como contábamos con más o menos 75 cartas fue un algoritmo de mucha ayuda.

- ¿Cómo se llevará a cabo la repartición de cartas a los jugadores?

Fue de manera autónoma, cada jugador tiene un tiempo sin embargo no es una opción ver a que jugador le toca que carta, sino que va de manera personal.

- ¿Cuántas cartas recibirán los jugadores al principio?

Las cartas son individuales y se toma de a 1 carta por jugador.

- ¿Cómo representarán las jugadas de los jugadores y los turnos en el juego?

El tiempo y cuando la baraja se vuelva a revolver determinarán los turnos de los diferentes jugadores.

- ¿Qué reglas se seguirán para permitir o restringir ciertas jugadas?

En realidad, contamos con tarjetas reglas y comodín que te pueden o ayudar o hacer más difícil el juego dependiendo de tu suerte y de la baraja.

- ¿Considerarán implementar una interfaz gráfica de usuario (GUI) para mostrar las cartas y las jugadas?

La verdad para nuestro juego no serviría mucho, por eso solo realizamos la interfaz gráfica para mostrar la carta que sale y el tiempo.

- ¿Cómo se mostrarán las cartas en la interfaz?

Son imágenes del tipo de carta y al lado conllevan el texto que dice cuál es el reto que deben seguir.

- ¿Qué tipo de documentación proporcionarán en el código para que otros puedan entender su implementación?

La explicación la agregamos en la presentación, sin embargo, es un juego bastante sencillo de comprender.

- ¿Qué comentarios agregarán a las secciones clave del código?

Tenemos que hacer el juego un poco más interactivo sin embargo si somos conscientes del proceso de mejora que podríamos implementar.

- ¿Cómo verificarán que su sistema funciona correctamente?

Las diferentes pruebas y errores antes de presentar el resultado final definieron los errores del juego.

- ¿Qué casos de prueba consideran necesarios?

Tal vez el uso de nuestro juego de manera presencial y si medimos el suficiente tiempo para realizar los retos.

- ¿Tienen ideas para expandir su proyecto más allá de los requisitos básicos?

No lo planteamos, pero esta es una muy buena base para crear un juego de este tipo a gran masa.

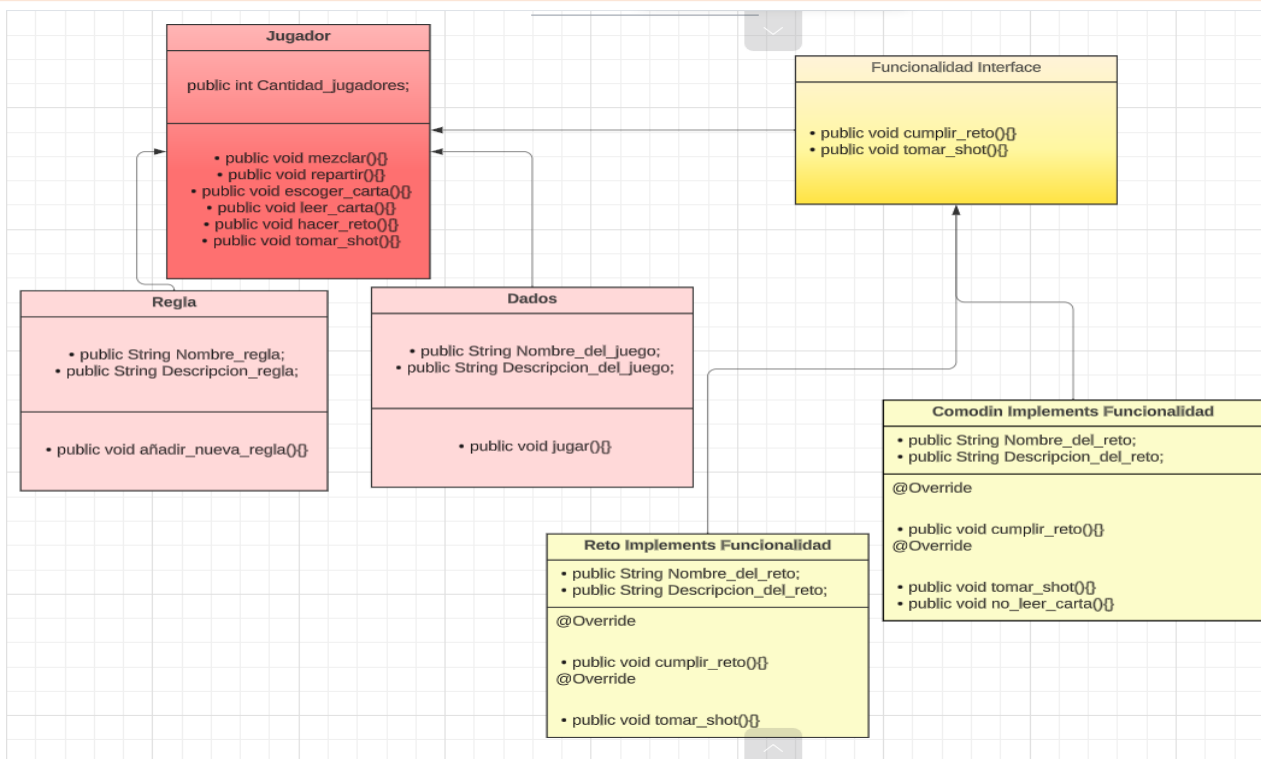
- ¿Qué características adicionales podrían agregar?

Tal vez un timer más sencillo o unos botones para saltar turnos

- ¿Cómo aplicarán conceptos de polimorfismo e interfaces en su proyecto?

Cada una de nuestras clases tenía su función, sin embargo, podemos ver el polimorfismo a la hora de ver el método tomar “shot” o realizar reto y estos están contenido en la interfaz funcionalidad.

## SOLUCION PROPUESTA



## CODIGO

```
1  /*
2  * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to
3  * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Main.java to edit this
4  */
5  package drink_game_proyecto;
6
7  import java.util.Random;
8  import javax.swing.JOptionPane;
9
10 /**
11  *
12  * @author Principal
13  */
14 public class Drink_Game_Proyecto {
15
16     /**
17      * @param args the command line arguments
18      */
19     public static void main(String[] args) {
20         new Empezar_1().setVisible(b: true);
21         Tiempo tiempo = new Tiempo();
22
23         // Esperar 5 segundos antes de iniciar el bucle
24         try {
25             Thread.sleep(millis: 5000); // 5000 milisegundos = 5 segundos
26         } catch (InterruptedException e) {
27             e.printStackTrace();
28         }
29
30         // Bucle para seleccionar al azar una carta
31         Random random = new Random();
32         while (true) {
33             // Lógica para elegir una carta al azar
34             int cartaSeleccionada = random.nextInt(bound: 4); // Reemplaza 3 con e
35
36             // Lógica para mostrar la carta en el JFrame
37             switch (cartaSeleccionada) {
38                 case 0:
39                     new Dados().setVisible(b: true);
40                     tiempo.ContarTiempo();
41                     break;
42                 case 1:
43                     new Comodin().setVisible(b: true);
44                     tiempo.ContarTiempo();
45                     break;
46                 case 2:
47                     new Reto().setVisible(b: true);
48                     tiempo.ContarTiempo();
49                     break;
50                 case 3:
51                     new Regla().setVisible(b: true);
52                     tiempo.ContarTiempo();
```

```

53         break;
54     }
55
56     try {
57         Thread.sleep(60000); // 60000 milisegundos = 60 segundos =
58     } catch (InterruptedException e) {
59         e.printStackTrace();
60     }
61 }
62
63 }
64
65 }
66

```

```

1  /*
2   * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to
3   * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this
4   */
5  package drink_game_proyecto;
6
7  import java.util.Random;
8
9  /**
10   *
11   * @author Principal
12   */
13  public class Carta_comodin extends Datos implements cumplir_reto, tomar_shot {
14      public Carta_comodin(String Nombre, String Descripcion){
15          super(Nombre, Descripcion);
16      }
17      public String NuevoComodin(){
18          String[] comodin = {"Comodin Saltar", "Comodin Bailar", "Comodin Cantar"};
19          Random random = new Random();
20          int indice = random.nextInt(bound: comodin.length);
21          return comodin[indice];
22      }
23
24      @Override
25      public void cumplir_reto(){
26
27      }
28      public void tomar_shot(){
29
30      }
31

```

```

1  /*
2   * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to
3   * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this
4   */
5  package drink_game_proyecto;
6
7  import java.util.Random;
8

```



```

9  /**
10 *
11 * @author Principal
12 */
13 public class Carta_dados extends Datos{
14     public Carta_dados(String Nombre, String Descripcion){
15         super(Nombre, Descripcion);
16     }
17
18     public String NuevoJuego(){
19         String[] juegos = {"1.Personalidades", "2. Sillas musicales", "3. Bebe y
20 Random random = new Random();
21 int indice = random.nextInt(bound: juegos.length);
22 return juegos[indice];
23     }
24
25 }

```

```

1  /**
2   * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to
3   * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit thi
4   */
5  package drink_game_proyecto;
6
7  import java.util.Random;
8
9  /**
10 *
11 * @author Principal
12 */
13 public class Carta_regla extends Datos{
14     public Carta_regla(String Nombre, String Descripcion){
15         super(Nombre, Descripcion);
16     }
17     public String NuevaRegla(){
18         String[] reglas = {"1.Por cada groseria deberás tomar 2 shots",
19 "2. Por cada shot que tome tu amigo de la izquierda deberás tomar 2 shots"
20 "3. Si te ries en las próximas 3 rondas deberás tomar 3 shots",
21 "4. Por cada vez que mienta uno del grupo deberás tomar 2 shots",
22 "5. Por cada vez que alguien diga el nombre de una persona del grupo deber
23 "6. Cada vez que alguien nombre a un animal deberás tomar 4 shots",
24 "7. Habla sobre ti mismo en tercera persona durante el próximo turno por c
25 "8. Solo vas a poder decir los nombres de tus ex en vez de los integrantes
26 "9. Cada vez que alguien se pare deberás tomar 2 shots",
27 "10. Por cada vez que alguien te nombre deberás tomar 1 shot."};
28         Random random = new Random();
29         int indice = random.nextInt(bound: reglas.length);
30         return reglas[indice];
31     }

```

```

1  /**
2   * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to
3   * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit thi
4   */
5  package drink_game_proyecto;

```

```

6
7 import java.util.Random;
8
9 /**
10  *
11  * @author Principal
12  */
13 public class Carta_reto extends Datos implements cumplir_reto, tomar_shot {
14     public Carta_reto(String Nombre, String Descripcion) {
15         super(Nombre, Descripcion);
16     }
17
18     public String obtenerRetoAlAzar() {
19         String[] retos = {
20             "1. Toma un shot de una mezcla de bebidas elegida por el grupo.",
21             "2. Llama a tu ex y cantan \nuna canción romántica",
22             "3. Comparte tu secreto más \nnoscuro o vergonzoso.",
23             "4. Comparte un secreto \nvergonzoso sobre \nalguien más en el grupo.",
24             "5. Realiza una búsqueda \nrápida en internet y lee en voz \nalta la última",
25             "6. Toma un shot por cada \naño que hayas estado \nen una relación.",
26             "7. Besa a la persona \na tu izquierda durante al \nmenos 30 segundos.",
27             "8. Realiza una llamada a \nun servicio de entrega de comida y \ncoquetea co",
28             "9. Comparte una fantasía \nque hayas tenido \nrecientemente",
29             "10. Crea una historia \nerótica corta y léela \nen voz alta,",
30             "11. Haz una llamada de broma \na un número aleatorio y coquetea \ncon la pe",
31             "12. Toma un shot por cada \nvez que hayas besado a \nalguien en el último a",
32             "13. Envía un mensaje de \ntexto a tu jefe o profesor \ndiciendo que no pued",
33             "14. Dibuja un tatuaje \ntemporal en tu cuerpo con un \nmarcador permanente",
34             "15. Envía una foto VERGONZOSA \na la tercera persona \nen tu lista de conta",
35             "16. Realiza una llamada \na un amigo y dile que te \nenamoraste de él/ella",
36             "17. Toma un shot por cada \nmes que hayas estado soltero/a en \nlos últimos",
37             "18. Besa a la persona \na tu derecha en el \n lugar que elijan",
38             "19. Envía un mensaje de \ntexto a tu ex diciendo que todavía \ntienes senti
39         };
40
41         Random random = new Random();
42         int indice = random.nextInt(bound: retos.length);
43         return retos[indice];
44     }
45     public String obtenerShotAlAzar() {
46         String[] shots = {"1 Shot", "2 Shots", "3 Shots"};
47         Random random = new Random();
48         int indice = random.nextInt(bound: shots.length);
49         return shots[indice];
50     }
51
52     @Override
53     public void cumplir_reto() {
54     }
55
56     public void tomar_shot() {
57     }
58 }

```



```

1  /*
2  * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to
3  * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit thi
4  */
5  package drink_game_proyecto;
6
7  /**
8   *
9   * @author Principal
10  */
11  public class Datos {
12      public String Nombre;
13      public String Descripcion;
14
15      public Datos(String Nombre, String Descripcion){
16          this.Nombre = Nombre;
17          this.Descripcion = Descripcion;
18      }
19  }
20
21  /*
22  * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to
23  * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit thi
24  */
25  package drink_game_proyecto;
26
27  import javax.swing.JOptionPane;
28  import javax.swing.SwingUtilities;
29
30  public class Tiempo {
31
32      public void ContarTiempo() {
33          Thread countdownThread = new Thread(() -> {
34              for (int i = 60; i >= 0; i--) {
35                  final int currentCount = i;
36                  SwingUtilities.invokeLater(() -> JOptionPane.showMessageDialog(p
37
38                      try {
39                          Thread.sleep(1000);
40                      } catch (InterruptedException e) {
41                          e.printStackTrace();
42                      }
43              }
44          });
45
46          countdownThread.start();
47      }
48
49      public static void main(String[] args) {
50          Tiempo tiempo = new Tiempo();
51          tiempo.ContarTiempo();
52      }
53  }

```

```

1  /**
2   * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to
3   * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Interface.java to edit
4   */
5   package drink_game_proyecto;
6
7   /**
8    *
9    * @author Principal
10   */
11   public interface cumplir_reto {
12       public void cumplir_reto();
13   }
14

```

```

1  /**
2   * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt t
3   * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Interface.java to edit
4   */
5   package drink_game_proyecto;
6
7   /**
8    *
9    * @author Principal
10   */
11   public interface tomar_shot {
12       public void tomar_shot();
13   }
14

```

```

1  /**
2   * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt t
3   * Click nbfs://nbhost/SystemFileSystem/Templates/GuiForms/JFrame.java to edit t
4   */
5   package drink_game_proyecto;
6
7   /**
8    *
9    * @author Principal
10   */
11   public class Empezar_1 extends javax.swing.JFrame {
12
13       /**
14        * Creates new form Empezar_1
15        */
16       public Empezar_1() {
17           initComponents();
18           this.setLocationRelativeTo(null);
19       }
20
21       /**
22        * This method is called from within the constructor to initialize the form.
23        * WARNING: Do NOT modify this code. The content of this method is always
24        * regenerated by the Form Editor.
25        */

```

26 @SuppressWarnings("unchecked")

27 + Generated Code

53 - private void EmpezarActionPerformed(java.awt.event.ActionEvent evt) {

55 this.setVisible(b: false);

56 new Carta().setVisible(b: true);

57 }

58

59 /\*\*

60 \* @param args the command line arguments

61 \*/

62 - public static void main(String args[]) {

63 /\* Set the Nimbus look and feel \*/

64 + Look and feel setting code (optional)

65

66 /\* Create and display the form \*/

67 java.awt.EventQueue.invokeLater(new Runnable() {

68 - public void run() {

69 new Empezar\_1().setVisible(b: true);

70 }

71 });

72 }

73

74 // Variables declaration - do not modify

75 private javax.swing.JButton Empezar;

76 private javax.swing.JLabel jLabel1;

77 // End of variables declaration

78 }

1 - /\*\*

2 \* Click <nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt> to

3 \* Click <nbfs://nbhost/SystemFileSystem/Templates/GuiForms/JFrame.java> to edit t

4 \*/

5 package drink\_game\_proyecto;

6

7 - /\*\*

8 \*

9 \* @author Principal

10 \*/

11 public class Carta extends javax.swing.JFrame {

12

13 - /\*\*

14 \* Creates new form Carta

15 \*/

16 - public Carta() {

17 initComponents();

18 this.setLocationRelativeTo(c: null);

19 }

20

21 - /\*\*

22 \* This method is called from within the constructor to initialize the form.

23 \* WARNING: Do NOT modify this code. The content of this method is always

24 \* regenerated by the Form Editor.

25 \*/

26 @SuppressWarnings("unchecked")

```

27  + | Generated Code
54  - |
55  | private void CartaActionPerformed(java.awt.event.ActionEvent evt) {
56  |     // TODO add your handling code here:
57  |     this.setVisible(b: false);
58  |     new Baraja().setVisible(b: true);
59  | }
60
61  - | /**
62  |     * @param args the command line arguments
63  |     */
64  - | public static void main(String args[]) {
65  |     /* Set the Nimbus look and feel */
66  |     Look and feel setting code (optional)
67
68  |     /* Create and display the form */
69  |     java.awt.EventQueue.invokeLater(new Runnable() {
70  |         public void run() {
71  |             new Carta().setVisible(b: true);
72  |         }
73  |     });
74  | }
75
76  - | // Variables declaration - do not modify
77  | private javax.swing.JButton Carta;
78  | private javax.swing.JLabel jLabel1;
79  | // End of variables declaration
80
81  - | }

```

```

1  - | /**
2  |     * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to
3  |     * Click nbfs://nbhost/SystemFileSystem/Templates/GUIForms/JFrame.java to edit t
4  |     */
5  | package drink_game_proyecto;
6
7  - | import java.awt.event.ActionEvent;
8  | import java.awt.event.ActionListener;
9  | import javax.swing.Timer;
10
11  - | /**
12  |     *
13  |     * @author Principal
14  |     */
15  | public class Baraja extends javax.swing.JFrame {
16
17  |     private Timer timer;
18  |     private int position = 400;
19
20  - | /**
21  |     * Creates new form Baraja
22  |     */
23  - | public Baraja() {
24  |     initComponents();
25  |     this.setLocationRelativeTo(c: null);
26  |     iniciarAnimacion();

```

```

27     }
28
29     /**
30      * This method is called from within the constructor to initialize the form.
31      * WARNING: Do NOT modify this code. The content of this method is always
32      * regenerated by the Form Editor.
33      */
34     @SuppressWarnings("unchecked")
35     Generated Code
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56     private void iniciarAnimacion() {
57         int duracionAnimacion = 5000; // Duración en milisegundos (5 segundos)
58         int paso = 50; // Paso de movimiento
59
60         int numeroPasos = duracionAnimacion / paso;
61         int periodo = duracionAnimacion / numeroPasos;
62
63         timer = new Timer(delay: periodo, e -> moverPuntosHorizontalmente());
64         timer.start();
65     }
66
67     private void moverPuntosHorizontalmente() {
68         position += 10;
69         puntos.setLocation(x: position, y: puntos.getY());
70
71         if (position >= 700) {
72             position = 400;
73         }
74     }
75
76     /**
77      * @param args the command line arguments
78      */
79     public static void main(String args[]) {
80         /* Set the Nimbus look and feel */
81         Look and feel setting code (optional)
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103         /* Create and display the form */
104         java.awt.EventQueue.invokeLater(new Runnable() {
105             public void run() {
106                 new Baraja().setVisible(true);
107             }
108         });
109     }
110
111     // Variables declaration - do not modify
112     private javax.swing.JLabel jLabel1;
113     private javax.swing.JLabel puntos;
114     // End of variables declaration
115 }

```



```

1  /*
2  * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to
3  * Click nbfs://nbhost/SystemFileSystem/Templates/GuiForms/JFrame.java to edit t
4  */
5  package drink_game_proyecto;
6
7  /**
8   *
9   * @author Principal
10  */
11  public class Comodin extends javax.swing.JFrame {
12
13      /**
14       * Creates new form Comodin
15       */
16      public Comodin() {
17          initComponents();
18          this.setLocationRelativeTo(c: null);
19          Carta_comodin comodinn = new Carta_comodin(Nombre: "Nombre del Reto", Descr
20          String ComodinAlAzar = comodinn.NuevoComodin();
21          Com.setText(text: ComodinAlAzar);
22      }
23      private void mostrarInformacionComodin(String[] comodin) {
24          StringBuilder mensaje = new StringBuilder(str: "Reglas:\n");
25          for (String carta_comodin : comodin) {
26              mensaje.append(str: carta_comodin).append(str: "\n");
27          }
28          Com.setText(text: mensaje.toString());
29      }
30
31
32      /**
33       * This method is called from within the constructor to initialize the form.
34       * WARNING: Do NOT modify this code. The content of this method is always
35       * regenerated by the Form Editor.
36       */
37      @SuppressWarnings("unchecked")
38      Generated Code
39
40
41      /**
42       * @param args the command line arguments
43       */
44      public static void main(String args[]) {
45          /* Set the Nimbus look and feel */
46          Look and feel setting code (optional)
47
48          /* Create and display the form */
49          java.awt.EventQueue.invokeLater(new Runnable() {
50              public void run() {
51                  new Comodin().setVisible(b: true);
52              }
53          });
54      }
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95

```

```

1  /**
2   * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to
3   * Click nbfs://nbhost/SystemFileSystem/Templates/GUIForms/JFrame.java to edit t
4   */
5  package drink_game_proyecto;
6
7  /**
8   *
9   * @author Principal
10  */
11  public class Dados extends javax.swing.JFrame {
12
13      /**
14       * Creates new form Dados
15       */
16      public Dados() {
17          initComponents();
18          this.setLocationRelativeTo(c: null);
19          Carta_datos juego = new Carta_datos(Nombre: "Nombre del Reto", Descripcion:"D
20          String JuegoAlAzar = juego.NuevoJuego();
21          Nuevo_J.setText(text: JuegoAlAzar);
22      }
23
24      private void mostrarInformacionJuegos(String[] juegos) {
25          StringBuilder mensaje = new StringBuilder(str:"Juegos:\n");
26          for (String carta_datos : juegos) {
27              mensaje.append(str:carta_datos).append(str:"\n");
28          }
29          Nuevo_J.setText(text: mensaje.toString());
30      }
31

```

```

1  /**
2   * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt t
3   * Click nbfs://nbhost/SystemFileSystem/Templates/GUIForms/JFrame.java to edit t
4   */
5  package drink_game_proyecto;
6
7  /**
8   *
9   * @author Principal
10  */
11  public class Regla extends javax.swing.JFrame {
12
13      /**
14       * Creates new form Regla
15       */
16      public Regla() {
17          initComponents();
18          this.setLocationRelativeTo(c: null);
19          Carta_regla regla = new Carta_regla(Nombre: "Nombre del Reto", Descripcion:"D
20          String ReglaAlAzar = regla.NuevaRegla();
21          Nueva_R.setText(text: ReglaAlAzar);
22      }
23
24      private void mostrarInformacionReglas(String[] reglas) {



```

```

24     StringBuilder mensaje = new StringBuilder(str: "Reglas:\n");
25     for (String carta_regla : reglas) {
26         mensaje.append(str: carta_regla).append(str: "\n");
27     }
28     Nueva_R.setText(text: mensaje.toString());
29 }
30
31

```

```

1  /*
2  * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to
3  * Click nbfs://nbhost/SystemFileSystem/Templates/GUIForms/JFrame.java to edit t
4  */
5  package drink_game_proyecto;
6
7  /**
8   *
9   * @author Principal
10  */
11  public class Reto extends javax.swing.JFrame {
12
13      /**
14       * Creates new form Reto
15       */
16      public Reto() {
17          initComponents();
18          this.setLocationRelativeTo(c: null);
19          Carta_reto reto = new Carta_reto(Nombre: "Nombre del Reto", Descripcion: "Desc
20          String retoAlAzar = reto.obtenerRetoAlAzar();
21          Reto_normal.setText(text: retoAlAzar);
22
23          String shotAlAzar = reto.obtenerShotAlAzar();
24          shot1.setText(text: shotAlAzar);
25      }
26
27       private void mostrarInformacionRetos(String[] retos) {
28          StringBuilder mensaje = new StringBuilder(str: "Retos:\n");
29          for (String reto : retos) {
30              mensaje.append(str: reto).append(str: "\n");
31          }
32          Reto_normal.setText(text: mensaje.toString());
33      }
34
35       private void mostrarInformacionShots(String[] shots) {
36          StringBuilder mensaje = new StringBuilder(str: "Shots:\n");
37          for (String reto : shots) {
38              mensaje.append(str: reto).append(str: "\n");
39          }
40          shot1.setText(text: mensaje.toString());
41      }
42
43      /**
44       * This method is called from within the constructor to initialize the form.
45       * WARNING: Do NOT modify this code. The content of this method is always
46       * regenerated by the Form Editor.

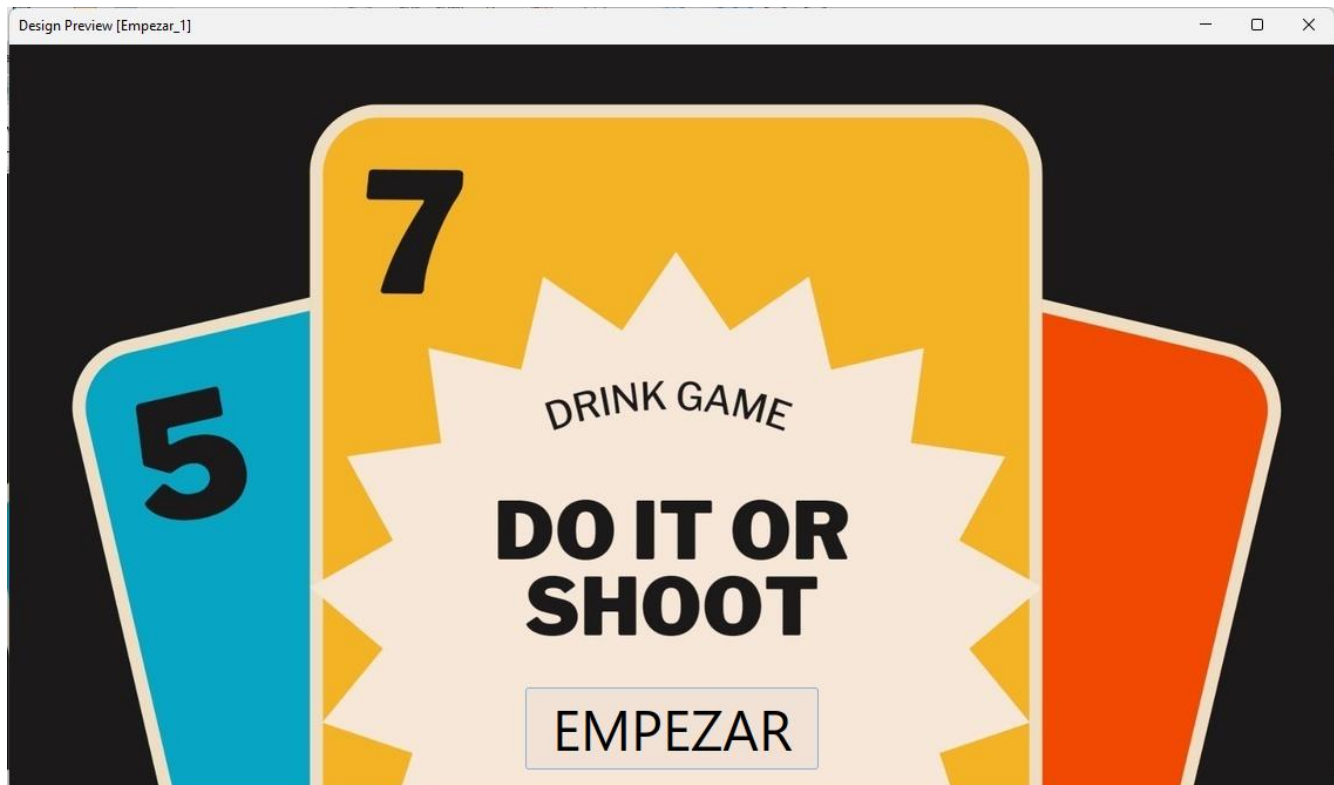
```

```

47 L      */
48      @SuppressWarnings("unchecked")
49      + Generated Code
81
82      - /**
83      |  * @param args the command line arguments
84      |  */
85      - public static void main(String args[]) {
86      |  /* Set the Nimbus look and feel */
87      |  + Look and feel setting code (optional)
108
109      |  /* Create and display the form */
110      |  java.awt.EventQueue.invokeLater(new Runnable() {
111      |  |  public void run() {
112      |  |  |  new Reto().setVisible(b: true);
113      |  |  }
114      |  }
115      |  });
116
117      // Variables declaration - do not modify
118      private javax.swing.JLabel Reto_normal;
119      private javax.swing.JLabel jLabel1;
120      private javax.swing.JLabel shot;
121      private javax.swing.JLabel shot1;
122      // End of variables declaration
123      }

```

## PRUEBAS Y RESULTADOS









Design Preview [Comodin]

— □ ×



**COMODIN**

Design Preview [Dados]

— ×



**Nuevo Juego**

Design Preview [Regla]

— □ ×



# Bomba

## Nueva Regla

Design Preview [Reto]

— □ ×



# RETO

## Shots

# ANÁLISIS DE RESULTADOS

1. Presentación del Proyecto: Nuestro proyecto se centra en el desarrollo de un juego de retos o "shots" que promete llevar la diversión y la intriga a un nuevo nivel. A través de la combinación de programación hábil y diseño de interfaz intuitiva, buscamos proporcionar a los jugadores una experiencia única y atractiva.

2. División de Bloques de Programación: Para alcanzar el éxito en nuestro proyecto, hemos organizado la programación en diversos bloques que se complementan entre sí. Estos bloques incluyen:

- **Interfaz Gráfica con JFrame:** La base de nuestro juego descansa en una interfaz gráfica desarrollada con Java y JFrame. Este componente es esencial para proporcionar una experiencia visual atractiva y amigable para el usuario.
- **Diseño de Cartas:** Cada detalle del juego se refleja en la creación de cartas visualmente atractivas que presentan los diferentes retos, comodines y otros elementos del juego. La atención al diseño contribuye significativamente a la inmersión del jugador.
- **Lógica del Juego:** La esencia del juego reside en la programación de la lógica detrás de cada movimiento, regla y desafío. La coherencia y fluidez del juego dependerán en gran medida de la precisión de esta parte del desarrollo.
- **Gestión de Eventos:** Implementamos una gestión eficiente de eventos para garantizar respuestas adecuadas a las acciones del jugador. La interactividad es clave, y nuestra programación se enfoca en brindar una experiencia fluida y receptiva.
- **Conexión entre Componentes:** La integración armoniosa de la interfaz, las cartas y la lógica del juego es crucial. La eficiente comunicación entre estos componentes garantiza un funcionamiento sin problemas y una experiencia de usuario cohesionada.

En resumen, nuestro proyecto no solo representa un desafío técnico en la programación de un juego interactivo, sino también una oportunidad para explorar la creatividad y la innovación en el diseño de experiencias lúdicas.

## CONCLUSIONES

Al culminar este proyecto de programación hemos alcanzado diversas conclusiones que reflejan no solo el éxito técnico, sino también el aprendizaje y la colaboración que han sido fundamentales en este proceso.

1. A lo largo de este proyecto, hemos fortalecido y ampliado nuestras habilidades técnicas en programación Java y en el uso de la biblioteca JFrame. La creación de una interfaz gráfica atractiva y funcional, la implementación de la lógica del juego y la gestión eficiente de eventos son aspectos que han contribuido significativamente al desarrollo de nuestras competencias como programadores.
2. La importancia de la estética y el diseño se ha vuelto evidente en la creación de nuestro juego. La integración de cartas visualmente atractivas con una lógica de juego sólida ha demostrado ser esencial para ofrecer una experiencia de usuario completa y envolvente. Este enfoque equilibrado entre diseño y funcionalidad es una lección valiosa que llevaremos adelante en futuros proyectos.
3. La resolución de problemas ha sido una constante, fomentando el pensamiento crítico y la búsqueda de soluciones creativas. Este proceso de superar obstáculos ha sido enriquecedor y nos ha permitido crecer como desarrolladores.
4. Este proyecto ha servido como plataforma de lanzamiento para futuros emprendimientos en el ámbito de la programación de juegos. Las lecciones aprendidas, las habilidades adquiridas y la experiencia acumulada nos posicionan para enfrentar con confianza y entusiasmo nuevos desafíos en el vasto mundo del desarrollo de software.

En conclusión, este proyecto no solo representa el logro de un producto final, sino también el crecimiento personal y profesional de cada miembro del equipo.