

Proyecto programado

Debe implementar una versión simplificada del juego de cartas Blackjack o 21. Al inicio del programa se mostrará un menú con las siguientes opciones:

- 1) Iniciar un nuevo juego
- 2) Instrucciones de juego.
- 3) Salir

Al escoger la **opción 3**, se le muestra un mensaje al usuario consultando si realmente quiere salir, si responde afirmativamente se mostrará un mensaje de agradecimiento y se finalizará el programa. Si responde negativamente se mantendrá en el menú. Lo mismo si presiona la X de la ventana de menú.

Al escoger la **opción 2**, se le deberán mostrar las reglas del juego, incluyendo el valor de las cartas y las descripciones de cómo gana la casa o el jugador.

Al escoger la **opción 1**, se solicitará el nombre al usuario, el mismo no podrá tener espacios, números o caracteres especiales, solo letras del alfabeto en mayúscula o minúscula (incluyendo ñ y tildes), en caso de que no cumpla la validación se deberá mostrar un mensaje de error y solicitarle nuevamente el nombre, esto se repetirá hasta que indique un nombre válido.

Posteriormente se iniciará el juego, para lo cual se repartirá una carta aleatoria al jugador y una carta aleatoria a la casa (computadora) y luego otra carta al jugador y otra a la casa. Cada vez que se reparte cualquier carta en el transcurso del juego se muestra individualmente su información en una ventana y a quién es repartida, excepto la primera carta de la casa para lo cual puede mostrar algo similar a “Carta inicial de la casa, no visible”. Tome en cuenta que el juego es visible desde la perspectiva del jugador, no de la computadora, por lo cual la casa tampoco sabe, o no toma en cuenta, el valor de la carta inicial del jugador.

Luego, se mostrará el estado del juego, este corresponde a cada carta del jugador y su puntaje total y cada carta de la casa (para la primera con el mismo mensaje utilizado en el punto anterior) y su puntaje total sin sumar la primera carta, seguido de la consulta al jugador de si quiere quedarse con la puntuación actual o pedir otra carta.

En caso de que el jugador pida otra carta, se mostrará la carta y luego el estado del juego. En caso de que el puntaje sea superior a 21 se le deberá indicar que ha ganado la casa y que termina el juego. Si no es superior a 21 se le consulta nuevamente si quiere quedarse o pedir otra carta. Se repite el proceso hasta que el jugador decida quedarse o pierda.

En el caso de que el jugador decida quedarse, se deberán repartir cartas a la casa, mientras el puntaje total de la casa (incluyendo la primera carta) sea inferior a 19. Si tiene 19 o más, no se reparten más cartas. Si el puntaje de la casa es mayor a 21 o inferior al del jugador, se muestra el estado final del juego (incluyendo la carta inicial de la casa y el puntaje total) junto con la indicación de que el jugador ha ganado. Si el puntaje de la casa es igual o superior, se muestra el estado final del juego junto con la indicación de que la casa ha ganado.

El estado del juego (jugador y casa) seguido de los mensajes de consulta o de gane o pérdida deben ir en una sola ventana.

Una vez que finaliza el juego se vuelve a mostrar el menú inicial de forma continua hasta que el usuario decida salir.

Detalles de implementación:

1. El programa debe realizarse utilizando el lenguaje Java y mediante el editor NetBeans
2. Deberá implementar al menos las siguientes clases:
 - **GestorEntradaSalida:** contiene los métodos necesarios para interactuar con el usuario, tanto para solicitar información como para mostrarla. Los métodos de esta clase deberían ser estáticos. La interfaz gráfica de usuario (GUI) deberá realizarse únicamente mediante métodos de la clase JOptionPane.
 - **Jugador:** representa un jugador con el nombre y su puntaje.
 - **Carta:** representa una carta de una baraja de naipes, con su figura y su valor. El valor de una carta puede ser un número (del 2 al 10) pero también letras (J, Q, K y A), para las figuras pueden utilizar caracteres especiales de las figuras vacías y llenas. **Nota:** se debe idear la forma de cuantificar el valor de las cartas con letras, pero sin sustituir las letras. **Por ejemplo:** el valor de una “J, Q, K y A” debe ser: “J, Q, K y A”, respectivamente, pero debe tener la posibilidad de generar un valor numérico de 10 para J, Q, K y 11 para A.
 - **Juego:** debe tener como atributos dos instancias de Jugador para el jugador y la casa y una instancia de RepartidorDeCartas. Debe tener un método jugar que tiene toda la estructura y lógica de juego. Por ejemplo: solicitar los nombres de los jugadores, llamar a repartir las cartas, evaluar el fin del juego, etc.
 - **RepartidorDeCartas:** debe tener un método para crear y retornar una instancia de una carta con sus valores y figuras de forma aleatoria. Nota: no es necesario controlar que los valores de la carta que se generen no se repitan. Esta clase no tiene atributos.
 - **Menú:** tiene un método estático para desplegar el menú principal, en el cual se debe llamar a iniciar jugar cuando corresponda. Esta clase no tiene atributos.
 - **Main:** tiene el método main que llama a iniciar el menú del programa.
- Los atributos tipo objeto deben ser inicializados en los constructores de la clase.
3. En todas las consultas al usuario, si digita opciones inválidas se le mostrará un mensaje de error y se le volverá a solicitar la opción hasta que digite una correcta.
4. No está permitido utilizar estructuras de datos, arreglos, colecciones ni excepciones, u otras clases de Java no vistas en el curso.
5. Deberá seguir las convenciones vistas en clase, más adelante se da un resumen de las mismas.

Convenciones de programación:

- Nombres de clases iniciando en mayúscula y representativo. Por ejemplo: Estudiante, EmpleadoAsalariado.
- Nombres de variables y métodos en minúscula. Por ejemplo: saldoDeCuenta, debitarCuenta().

- Nombres de constantes en mayúscula. Por ejemplo: CANTIDAD_MAXIMA_DE_USUARIOS.
- Nombres significativos. Por ejemplo: saldoCuentaCorriente, en lugar de la palabra dinero.
- Nombres de métodos que denoten acción cuando corresponda. Por ejemplo: calcularSalario(), desplegarMensaje().
- Iniciar cada palabra significativa con mayúscula cuando se tengan nombres compuestos por varias palabras (excepto la primera), o separados por el carácter “_” solo para el caso de constantes. Por ejemplo: saldoDeCuentaCorriente, CANTIDAD_MAXIMA_DE_USUARIOS.
- Identación de forma adecuada y legible: Alineamiento apropiado entre las líneas de código que se encuentran a cada nivel de anidamiento. (Algunos se refieren a esto como Tabulación). Espaciado adecuado entre llaves, operadores, paréntesis y métodos.
- Separación de declaraciones de variables en diferentes líneas cada una.
- Uso correcto de constantes (final) para no tener valores literales dentro de los métodos del programa.
- Uso correcto de los modificadores de acceso public, private, etc.
- Despliegue adecuado de hileras y ortografía en las mismas. Los mensajes a mostrar en las ventanas deben ser significativos, completos y descriptivos. Cada ventana deberá incluir un título significativo.
- Uso de modularización en el programa.
- Comentarios explicativos al inicio de cada clase que describa su funcionamiento general. En el comentario de la clase que incluye el método main debe incluir los nombres de los integrantes.

Evaluación del programa	Valor
Comentarios explicativos para las clases	2 Pts
Estándares y buenas prácticas de programación orientada a objetos *esto también puede ser evaluado en los demás rubros	5 Pts
Clases main y menu	6 Pts
Clase GestorEntradaSalida	6 Pts
Clase Jugador	6 Pts
Clase Carta	10 Pts
Clase Juego, lógica del juego e interacción entre clases y objetos	35 Pts
RepartidorDeCartas	10 Pts
Defensa (proporcional a los puntos obtenidos en la tarea, por lo que puede variar)	20 Pts
Total	100 Pts

Puntos opcionales:

- **Mostrar gráficamente las cartas (5 pts):** que cada vez que se reparte una carta, en la ventaja que se muestra la información de esta, se muestre también un ícono con la imagen **pequeña** de la carta correspondiente. Tome en cuenta que eso implica que la carpeta del proyecto debe tener 52 archivos de imagen, las cuales deben ser pequeñas y no muy alta calidad, para que el archivo no supere los 512 MB máximos de mediación. Además, de que no deben tener 52 líneas de código para íconos de esas imágenes, deben jugar con el nombre del archivo de tal forma que incorpore la figura y valor de la carta generada, una sola línea de código.
- **Expresiones regulares (5 pts):** investigar sobre expresiones regulares y utilizarlo para la validación del nombre del usuario, de tal forma que se valide los posibles caracteres en una sola expresión.
- **Reglas adicionales del juego:** incorporar una de las reglas adicionales de juego, por ejemplo que el valor de la carta A pueda tomar el valor 1 u 11 según la lógica de juego, que 5 menores o 20 y medio, le ganan a varios juegos de 21, que ciertas cartas permiten cambio de juego si el usuario lo quiere, etc.
- **Cartas no repetidas (5 pts):** validar que la carta que se va a lanzar, no haya sido lanzada previamente. Para esto no puede utilizar arreglos, estructuras de datos, colecciones, ni similares, debe utilizar lo visto hasta el momento del curso y que no implique tener 52 variables para comprobación.