Universidad Nacional
Facultad de Ciencias Exactas y Naturales
Escuela de Informática

EIF201 – Programación I

**Proyecto #2 – Valor 15%**

Objetivos

- Desarrollar una aplicación utilizando técnicas de programación orientadas a objetos
- Implementar relaciones de herencia entre objetos
- Implementar clases abstractas y polimorfismo
- Implementar listas simplemente enlazadas de objetos
- Interpretar el diseño de una aplicación basado en un diagrama de clases UML
- Consolidar técnicas algorítmicas para el desarrollo de la solución
- Comprender la importancia de la reutilización de código fuente en el desarrollo de aplicaciones
- Promover la colaboración en el desarrollo de un proyecto de software

**Descripción**

El proyecto consiste en desarrollar una versión simplificada del juego de cartas llamado “Blackjack”, mediante la utilización del lenguaje de programación C++ y utilizando técnicas de programación orientada a objetos, fundamentada en buenas prácticas de diseño e implementación. El programa deberá ejecutarse en un ambiente de consola y en buena medida con una interfaz de usuario sencilla, intuitiva y fácil de utilizar.

**La Baraja (Mazo)**

El “Blackjack” es un juego de cartas muy popular y bastante sencillo de comprender. En el juego se utiliza una baraja inglesa de 52 cartas. Las 52 cartas se dividen en 4 palos (2 de color rojo y 2 de color negro):

 Espadas (también llamado picas o bastos)

 Corazones

 Diamantes

 Tréboles

Cada palo está formado por 13 cartas, que ordenadas de menor a mayor forman la siguiente secuencia:

**A 2 3 4 5 6 7 8 9 10 J Q K**

**¿En qué consiste el juego?**

En este juego participan hasta un máximo de 7 jugadores y el “*dealer*” (conocido como la casa).

A cada jugador se le reparte una carta (una carta cada vez en el sentido de las agujas del reloj, el “dealer” recibe la última carta de la ronda), para un máximo de dos cartas por jugador (esto es lo que se conoce como “la mano”. Las dos cartas que reciben los jugadores están vueltas hacia arriba, por

lo que es posible ver el juego. Mientras tanto, el “dealer” coloca la primera carta hacia arriba y la segunda carta hacia abajo (oculta a los jugadores).

El objetivo del juego es alcanzar un valor lo más próximo a 21 pero sin exceder ese número. El “dealer” compite contra todos los jugadores, los jugadores no compiten entre sí (la idea del juego es derrotar a la casa).

Una vez que los jugadores reciben las dos cartas, cada jugador tiene la oportunidad de tomar una carta adicional si así lo desea, por turnos -en el mismo orden en el que se repartieron las cartas-. No necesariamente un jugador tiene que pedir una carta extra, en cuyo caso puede “pasar”. En ese caso, el jugador queda con la mano actual y se pasa el turno al siguiente jugador. Cada jugador puede pedir cartas adicionales, pero solamente lo puede hacer en su turno hasta que ya decida no pedir más cartas. Cuando ya todos los jugadores tomaron la oportunidad de pedir cartas extra, la casa revela la carta oculta (la pone hacia arriba). La casa debe tomar cartas adicionales mientras el total de la suma de las cartas sea 16 o menor. Si la casa se pasa de 21, entonces todos los jugadores que no pasaron de 21 ganan automáticamente. De otra manera, se debe comparar el total de la suma de las cartas de cada jugador contra el de la casa.

- Si la suma de las cartas de jugador es mayor al de la casa – El jugador gana
- Si la suma de las cartas de jugador es menor al de la casa – El jugador pierde
- Si la suma es igual para la casa y el jugador, hay un empate

Valor de las cartas

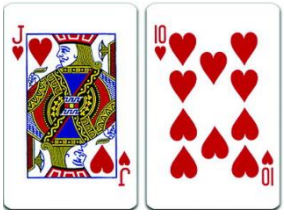
Para obtener la suma de la mano, los valores numéricos de las cartas se toman de la siguiente manera:

- Las cartas con valores numéricos (del 2 al 10), toman su valor numérico
- La carta A (llamado ***AS***), puede tomar dos valores: 1 (uno) u 11 (once) – decisión del jugador
- Las cartas **J, Q y K**, valen 10 (diez)

Ejemplos de manos

A continuación, se presentan algunos ejemplos de manos y sus respectivos valores en el juego

La mano es **19** (si se toma A como 11) o 9 (si se toma A como 1)



La mano es **20** (J = 10)

Puede consultar los siguientes enlaces para obtener información acerca de las reglas del juego y la metodología:

https://es.wikipedia.org/wiki/Blackjack

http://www.casino.es/blackjack/como-jugar-blackjack/

**Flujo de la aplicación**

La aplicación debe permitir la participación de varios jugadores en el mismo juego (máximo 7 jugadores por juego). Además de los jugadores, siempre jugará el "dealer" – que será el CPU.

Pantalla inicial

El juego iniciará con una pantalla de bienvenida al juego, y se le presentará al usuario dos opciones:

1) Nuevo Juego: Inicia un juego nuevo de Blackjack
2) Cargar Partida: Carga una partida guardada de un juego previo de Blackjack
3) Salir: Termina el programa

Nuevo Juego

- Se le presentará al usuario una pantalla en donde se podrá seleccionar la cantidad de jugadores. Una vez seleccionada la cantidad de jugadores, se solicitará un "nickname" para cada jugador . los "nicknames" no se pueden repetir. Luego pasa a la pantalla de juego.

  **Opcional**: Puede incorporar una opción en donde se cargue un archivo con los "nicknames" de los jugadores, para no digitarlos.

Cargar Partida

- Se cargará un archivo con la información de la partida guardada. Una vez cargado el archivo, se muestra inmediatamente la pantalla de juego con los "nicknames" de los jugadores y su respectiva mano, además de mantener el turno que seguía.

Pantalla de Juego

Antes de repartir las cartas a cada jugador, debe inicializar y barajar el mazo (es decir, acomodar todas las cartas de manera aleatoria). Esta acción se debe realizar antes de iniciar cada partida.

La pantalla de juego muestra el "nickname" de cada jugador con su respectiva mano. Además, inicialmente se muestra el juego del "dealer" con una carta visible y la otra oculta.

Para representar la mano del jugador, puede utilizar el siguiente formato:

**AC 10D** (representa un *A de Corazones*, y un *10 de Diamantes*)

**2T KE** (representa un *2 de Tréboles* y una *K de Espadas*)

Para cada turno de los jugadores, se solicitará al usuario que digite una opción si desea pedir carta o si desea pasar. También se incorporarán opciones que permitan guardar la partida actual del juego y salir. Por ejemplo:

```
JosePablo888:
                AD  KC
    (D)eme carta  -  (P)asar  -  (G)uardar Partida  - (S)alir
```

Deme carta: Toma una carta del mazo y se agrega a la mano del jugador. Si se pasa de 21 pierde y el turno pasa al siguiente jugador.

Pasar: El jugador queda con la misma mano y pasa el turno al siguiente jugador.

Guardar Partida: Almacena el estado actual del juego en un archivo (se sugiere solicitar el nombre del archivo al usuario en caso de querer guardar varias partidas)

Salir: Termina el juego. Retorna a la pantalla inicial

Una vez que termina el turno de todos los jugadores, se mostrará la carta oculta del "dealer". Se seguirán las reglas descritas anteriormente (si el valor de la mano del dealer es menor a 16, debe pedir carta).

Finalmente, se comparará el resultado final de la mano de cada jugador y el "dealer", y se mostrará en pantalla el resultado para cada jugador (ganó, perdió o empate). Luego, podrá preguntar si se desea repetir el juego con los mismos jugadores o retornar a la pantalla inicial.

Tome en consideración que solamente se implementarán las siguientes jugadas del blackjack: ***pedir carta o quedarse (check)***. No es necesario desarrollar las siguientes reglas: doblar, separar, asegurar la apuesta o rendirse. Por lo tanto, el juego solamente deberá, repartir las cartas, solicitarle al usuario si desea pedir carta o quedarse y finalmente mostrar el resultado final. Para esta versión del juego ***no debe solicitar una apuesta***.

Elementos del programa

Tome en consideración los siguientes aspectos que debe incorporar en la aplicación:

- Debe utilizar una lista enlazada simple de jugadores. En tiempo de ejecución, se determina cuál tipo de jugador debe realizar la jugada. La lista almacena apuntadores a los jugadores. También puede implementar listas en otras partes del programa que considere necesario.

- En el caso de utilizar vectores dentro de su implementación, debe utilizar asignación de memoria dinámica. También asegúrese de almacenar apuntadores a objetos dentro de los vectores. No se permite el uso de vectores estáticos.

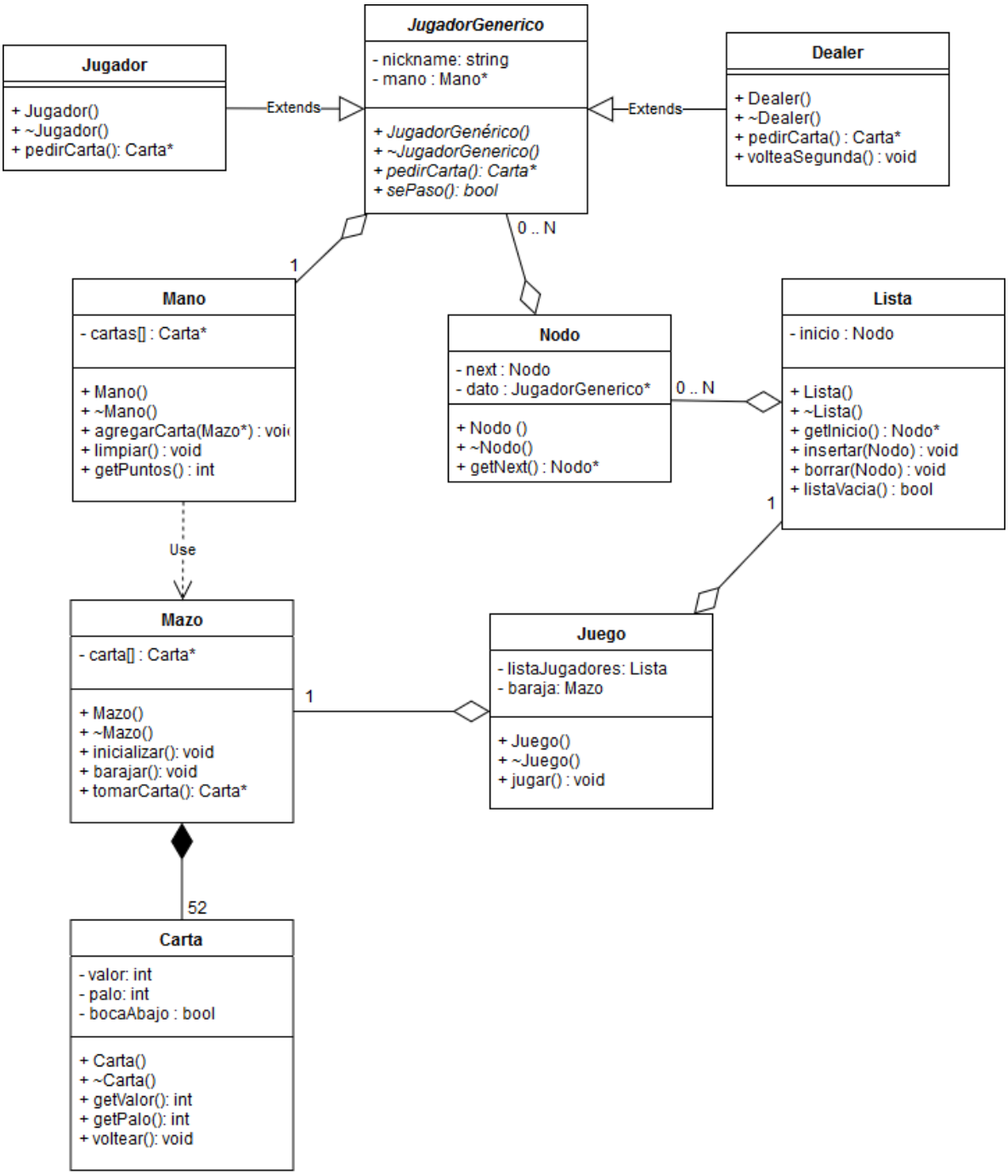
- Una herramienta que puede ser de utilidad para manejar las cartas, es el uso de ***Enumeraciones***. Considérelo como una posibilidad para implementar el valor y palo para cada carta.

- Identifique posibles condiciones de error que puedan hacer que el programa termine de manera abrupta durante su ejecución y asegúrese de validar las entradas y operaciones apropiadamente.

- Documente el código apropiadamente. Por ejemplo, entradas y salidas de los métodos y una descripción general de lo que realizan.

**Diagrama de clases**

Para el desarrollo del proyecto, puede utilizar como base el siguiente diagrama de clases UML:




Los métodos y atributos son sugeridos. Usted puede realizar cualquier cambio en el diseño si lo considera necesario, así como agregar más funciones, clases, etc. Note que en diagrama no están contempladas las funciones para trabajar con los archivos.

**Observaciones Generales**

- Utilice el lenguaje de programación C++ en Visual Studio 2017 para el desarrollo del proyecto.
- La tarea debe ser realizada en grupos de dos personas. No se permite trabajar de manera individual (a excepción de casos calificados y discutidos con anterioridad con el profesor).
- Se debe entregar el proyecto con el código fuente completo, así como el diagrama UML final de su proyecto.
- La fecha límite para la entrega es el día sábado 26 de enero a las 11:55pm.
- En caso de copia (dos proyectos o más con mucha similitud que lo demuestren) o plagio (códigos descargados de Internet, libros o cualquier otro material), la nota del proyecto es automáticamente 0.
- Debe indicar todos los recursos consultados para le elaboración del proyecto.

| **Tabla de Evaluación** | |
|---|---|
| Validaciones | 10 pts |
| Uso adecuado de listas simplemente enlazadas, herencia, clases abstractas, polimorfismo | 40 pts |
| Funcionamiento adecuado del sistema | 35 pts |
| Uso adecuado de archivos | 15 pts |
| **Total** | **100pts** |