1. Definicion de problema:

Hace poco en la ciudad de Cali una experimentada empresa que maneja un gran casino ha decidido incursionar en el mundo de las casas de apuestas y lo ha hecho inaugurando un novedoso hipódromo llamado El indomable Spirit. Debemos crear una aplicacion que cumpla estos requerimientos:

* + Se debe permitir el registro de entre siete y diez jinetes con sus respectivos caballos que serán quienes competirán en cada carrera y cuyo orden de salida está dispuesto por el orden de inscripción.
  + Se deben mostrar desplegados los competidores en pantalla, junto con su información respectiva: nombre del jinete, nombre del caballo y pista.
  + Se debe permitir el registro de apuestas para los usuarios, registrando su cédula, nombre, caballo por el que apuesta y monto apostado solo por un tiempo limitado (3 minutos).
  + Debe ser posible visualizar el podio de ganadores de la misma.
  + Los usuarios deben poder consultar rápidamente, a partir de su cédula, el registro de su apuesta donde además debe aparecer ahora si su caballo ganó o perdió la carrera.
  + Debe haber una opción “revancha” donde una carrera puede ser repetida por los mismos jinetes, pero en este caso el primero que llegó a la meta en la carrera anterior será el jinete del último carril y el primer carril lo ocupará el jinete que llegó de último.
  + Debe ser posible crear una nueva carrera donde se repite todo el flujo.

1. Busqueda de informacion necesaria:

**Estructura de datos:**

Es una colección de valores, la relación que existe entre estos valores y las operaciones que podemos hacer sobre ellos; en pocas palabras se refiere a cómo los datos están organizados y cómo se pueden administrar. Una estructura de datos describe el formato en que los valores van a ser almacenados, cómo van a ser accedidos y modificados, pudiendo así existir una gran cantidad de estructuras de datos.

**Stack**

Un stack o pila, soporta la recuperación ordenada de datos last-in, first-out (LIFO) o bien: el último dato en entrar, el primer dato en salir. De la misma forma en que se hace en una pila de platos limpios, si necesitamos un plato limpio vamos a la pila y tomamos el primero de la pila, que en realidad fue el último plato que agregamos a ella. Las pilas son estructuras que encontramos de muchas formas en el mundo real, siempre que podamos apilar algún objeto como libros, cazuelas, películas o la forma en la que metemos latas de refresco en el refrigerador.

**Queue**

Una queue o cola, soporta la recuperación ordenada de datos first-in, first-out (FIFO) o bien: el primer dato en entrar, es el primer dato en salir. Justo como una cola en el banco cuando vamos a realizar alguna operación con nuestra cuenta bancaria, si todos los asistentes están ocupados se genera una cola donde el primero que llegó será el primero en ser atendido y el resto esperamos en la cola. Este tipo de estructuras se utiliza mucho en el control de tiempos de espera de servicios, tiempos de ejecución en un CPU, de conexión de red, o en el mundo real en una cola para las tortillas, para entrar en una avenida rápida, etc.

**Tabla Hash:**

Es una estructura de datos no lineal cuyo propósito final se centra en llevar a cabo las acciones básicas (inserción, eliminación y búsqueda de elementos) en el menor tiempo posible, mejorando las cotas de rendimiento respecto a un gran número de estructuras.

1. **Búsqueda de Soluciones Creativas:**

Gracias a la búsqueda de información, nos dimos cuenta que java maneja varias estructuras de datos las cuales pueden dar una solución apropiada a nuestro problema:

* Alternativa 1: Usar la clase stack, la cual permite almacenar objetos y luego recuperarlos en orden inverso, “el último dato en entrar, el primer dato en salir”. Esta clase nos permite insertar elementos gracias a su método push().
* Alternativa 2: ArrayList es una colección donde cada objeto puede ser ubicado por su índice, su tamaño es flexible y nos permite agregar elementos gracias a su método add y también nos permite acceder a un elemento en específico.
* Alternativa 3: La clase linkedList se basa en la implementación de listas doblemente enlazadas, funciona con nodos y nos permite una rápida inserción al inicio o final de la lista, también implementa la clase queue, la cual nos permite almacenar objetos de la forma FIFO (Firts in- First Out)el primero en entrar, es el primero en salir.
* Alternativa 4: Tabla hash es una estructura de datos la cual nos permite la inserción, eliminación y búsqueda de elementos en el menor tiempo posible mejorando el rendimiento de nuestra estructura.

1. **Transición de las Ideas a los Diseños Preliminares**

Al hacer un análisis de nuestras ideas y de cómo podemos implementar cada una de las alternativas decidimos descartar la alternativa 2, dado que ArrayList proporciona una manipulación lenta y es ineficiente para insertar o eliminar elementos que estén en el final, por lo que requiere desplazar todos los últimos elementos para abrir o llenar un espacio.

La revisión cuidadosa de las demás alternativas nos conduce a lo siguiente:

**Alternativa 1: Stack**

Es una clase de tipo LIFO (Last in. First out) y con cinco operaciones nos permite tratar un vector a modo de pila, estas operaciones son Push (introduce un elemento en la pila), Pop (saca un elemento de la pila), peek (consulta el primer elemento), empty (comprueba si la pila está vacía) y search (busca un elemento determinado)

**Alternativa 3: LinkedList**

Nos proporciona una implementación rápida por que utiliza listas doblemente enlazadas, se puede usar como lista y cola por que implementa la interfaz de List, Deque y queue.

**Alternativa 4:** Tabla hash

1. **Evaluación y Selección de la Mejor Solución**

https://medium.com/techwomenc/estructuras-de-datos-a29062de5483

https://www.ecured.cu/Tabla\_hash

DIAGRAMA DE CLASES: <https://www.lucidchart.com/documents/edit/37d765f4-65f8-4d89-80c9-54485ee4a322/0_0?beaconFlowId=78D26FC2C0AA3E8A>