1. **Identificación del problema**

   a. **Contexto del Problema**

   En 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.

   b. **Definición del problema**

   Se desea incorporar un sistema donde se pueda manejar el flujo de toda la operación del hipódromo.

   c. **Identificación de requerimientos funcionales**

   i. **R1:** El sistema debe permitir el registro entre siete y diez jinetes con sus respectivos caballos. Para el registro es necesario introducir el nombre del jinete y del caballo.
   ii. **R2:** El sistema ordenará a los jinetes con sus caballos en el carril correspondiente según el orden de inscripción.
   iii. **R3:** El sistema antes de iniciar una carrera mostrará el nombre del jinete, el nombre del caballo y el carril.
   iv. **R4:** El sistema registrará las apuestas de cada carrera durante 3 minutos. para registrar una apuesta es necesario que cada usuario digite su cédula, su nombre, caballo por el que apuesta y monto apostado.
   v. **R5:** El sistema debe arrancar la carrera una vez se haya cerrado el periodo de apuestas.
   vi. **R6:** Al finalizar la carrera el sistema <debe mostrar al podio ganador.
   vii. **R7:** Al finalizar la carrera el sistema debe permitir al usuario introducir su cédula para saber si ganó la apuesta o no con su caballo.
   viii. **R8:** El sistema debe permitir una opción de revancha en la cual el primer y último jinete en llegar en la carrera anterior, cambiarán de carriles entre ellos.
   ix. **R9:** El sistema debe permitir crear una nueva carrera en la cual se repite todo el flujo.

2. Recopilación de la información necesaria

Hipódromo

Es una arena apta para disputar carreras de caballos. El interior tiene gradas en el perímetro, y el centro está formado de tierra o hierba. En el centro se dispone un óvalo bordeando las gradas que forma la pista. En la pista se disputan carreras de caballos.

Las pistas pueden ser de tierra (arena) o de hierba (césped). En las de tierra se disputan carreras de galope o de trotón mientras que las de hierba son para carreras de galope, con o sin saltos. En el recinto, la gente que entra a ver las carreras tiene la posibilidad de hacer apuestas.

Sistemas de apuestas de carreras

Un método de juego que permite a los apostadores apostar por el resultado incremental de una carrera. Una raza seleccionada se divide en una pluralidad de segmentos. Se aceptan apuestas de los apostadores que seleccionan los números correspondientes a un tiempo o fracción del mismo de un corredor que tiene el tiempo transcurrido más rápido para cada uno de los segmentos seleccionados de la carrera. El ganador se determina después de completar la carrera en función de los tiempos de carrera reales de los corredores a través de cada uno de los segmentos seleccionados de la carrera. Si hay uno o más apostadores ganadores, a los apostadores ganadores se les paga una cantidad proporcional a la apuesta total dividida por el número de apostadores ganadores.

(https://patents.google.com/patent/US20110250938A1/en)

Estructura de datos

En ciencias de la computación, una **estructura de datos** es una forma particular de organizar datos en una computadora para que puedan ser utilizados de manera eficiente. Diferentes tipos de estructuras de datos son adecuados para diferentes tipos de aplicaciones, y algunos son altamente especializados para tareas específicas.
(https://es.wikipedia.org/wiki/Estructura_de_datos#Estructuras_de_datos_en_programaci%C3%B3n)

Pila

Es una lista ordenada o estructura de datos que permite almacenar y recuperar datos, siendo el modo de acceso a sus elementos de tipo **LIFO** (del inglés *Last In, First Out*, «último en entrar, primero en salir») . Esta estructura se aplica en multitud de supuestos en el área de informática debido a su simplicidad y capacidad de dar respuesta a numerosos procesos.
(https://es.wikipedia.org/wiki/Pila_(inform%C3%A1tica))

Colas

Es una estructura de datos, caracterizada por ser una secuencia de elementos en la que la operación de inserción *push* se realiza por un extremo y la operación de extracción *pull* por el otro. También se le llama estructura FIFO (del inglés *First In First Out*), debido a que el primer elemento en entrar será también el primero en salir. (https://es.wikipedia.org/wiki/Cola_(inform%C3%A1tica))

Fuentes:

Comunidad de wikipedia. (2020). Wikipedia: Hipódromo. Recuperado de https://es.wikipedia.org/wiki/Hip%C3%B3dromo

Bassignana, M. E., & Jean, J. R. R. (2011). *U.S. Patent Application No. 12/903,631*.

Ferguson, T. S. (1965). Betting systems which minimize the probability of ruin. *Journal of the Society for Industrial and Applied Mathematics*, *13*(3), 795-818.

Pheng, S., & Verbrugge, C. (2006, June). Dynamic data structure analysis for Java programs. In *14th IEEE International Conference on Program Comprehension (ICPC'06)* (pp. 191-201). IEEE.

3. Soluciones creativas

De acuerdo a la especificación del problema y lo investigado escogimos la metodología de lluvia de ideas para plantear las siguientes posibles soluciones:

Alternativa 1:

3.1.1 Se plantea resolver la carrera de los jinetes con la creación de un ArrayList de los jinetes que estaban participando, crear un arreglo de números enteros y crear un cola de resultado. El objetivo es sacar un número al azar, con este número se buscaba en el arraylist de jinetes al jinete seleccionado.

3.1.2 Para las apuestas se plantea utilizar un hashtable con exploración lineal

3.1.3 Para el sistema de revancha se propone manejar una pila para invertir la posición de los jines según el orden de llegada.

Alternativa 2:

3.2.1 Se plantea resolver la carrera de los jinetes con la creación de dos colas: temporal 1 y temporal 2 y una resultado cola, la idea era que, para solucionar el problema anterior, si por ejemplo salía el caballo 4, los caballos 1, 2 y 3 se guardarán en temporal 2 para no perder esa información.

3.2.2 Para las apuestas se plantea utilizar un hashtable con exploración lineal

3.2.3 Para el sistema de revancha se propone manejar una pila para invertir la posición de los jines según el orden de llegada.

Alternativa 3:

3.3.1 Se plantea resolver la carrera de los jinetes con la creación de una sola cola que sería el resultado y otra cola temporal que nos daría los jinetes que estaban participando, se sacaría un número al azar y se compararía con el número de carril de los caballos inscritos.

3.3.2 Para las apuestas se plantea utilizar un hashtable con exploración lineal

3.3.3 Para el sistema de revancha se propone manejar una pila para invertir la posición de los jines según el orden de llegada.

4. Transición de ideas a diseños preliminares

Alternativa 1

3.1.1 Esta alternativa permite escoger al jinete seleccionado de forma precisa sin pérdida de información de jinetes y sin necesidad de recorrer toda la lista.

Alternativa 2

3.3.1 Esta alternativa presenta pérdida de información de los jinetes ya que al ser una cola van saliendo de la misma.

Alternativa 3

3.3.1 Esta alternativa agrega complejidad al algoritmo solución y resulta de difícil entendimiento.

5. Evaluación y selección de la mejor solución

Criterio A: La velocidad del algoritmo.

[3] La solución es la más rápida

[2] La solución es relativamente rápida

[1] La solución es extremadamente lenta

Criterio B: Confiabilidad de información

[3]: El algoritmo es confiable al no perder la información.

[2]: El algoritmo pierde cierta información

[1]: El algoritmo pierde mucha información

Criterio C: Dificultad de implementación

[3]: El algoritmo es fácil de entender e implementar

[2]: el algoritmo es moderadamente difícil de entender e implementar

[1]: el algoritmo es muy difícil de entender e implementa

Alternativa (Algoritmo)	Criterio A	Criterio B	Criterio C	Total
Alternativa 1 (3.1.1)	2	3	3	8
Alternativa 2 (3.2.1)	2	1	2	5
Alternativa 3 (3.3.1)	2	3	1	6

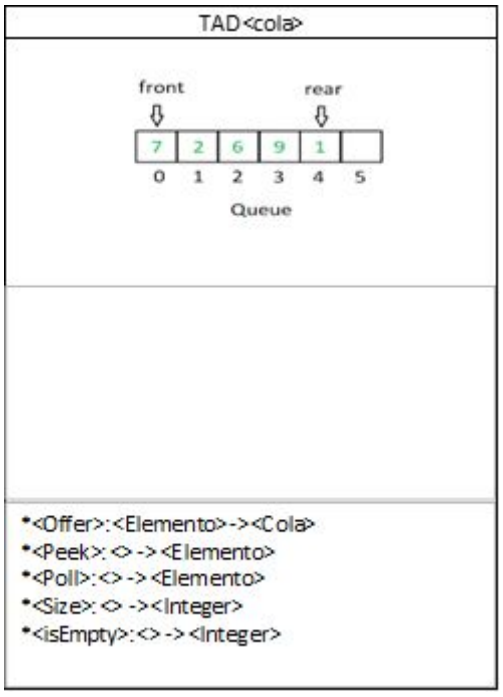
Elección final

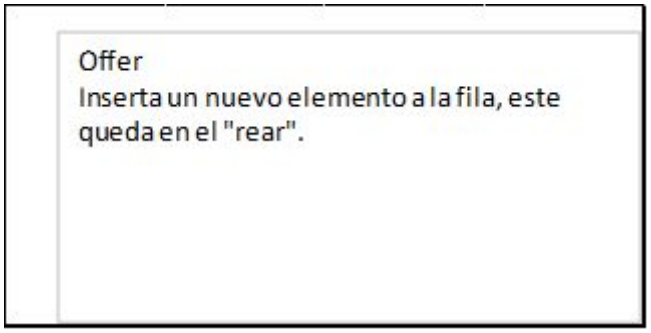
Con la comparación de los criterios es fácil ver que la alternativa 1 es la mejor opción ya que no tiene pérdida de información, el algoritmo es de fácil entendimiento y da solución a la necesidad del requerimiento.

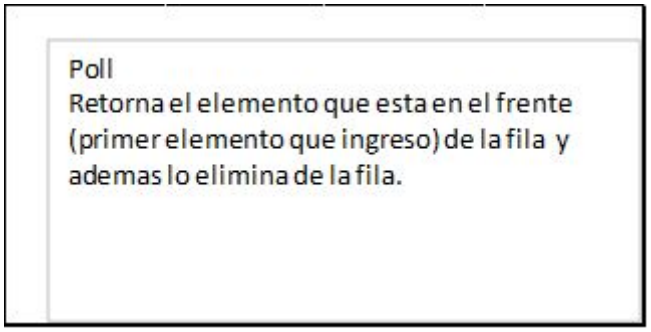
6. **Preparando el reporte**
    a. **Diagrama de flujo**

        Se adjunta en la carpeta

b. TAD's


TAD<cola>
front
rear
7
2
6
9
1
0
1
2
3
4
5
Queue
*<Offer>:<Elemento>-><Cola>
*<Peek>:<> -><Elemento>
*<Poll>:<> -><Elemento>
*<Size>:<> -><Integer>
*<isEmpty>:<> -><Integer>


Offer
Inserta un nuevo elemento a la fila, este queda en el "rear".

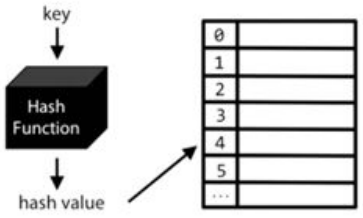

Poll
Retorna el elemento que esta en el frente (primer elemento que ingreso) de la fila y ademas lo elimina de la fila.

Poll
Retorna el elemento que esta en el frente (primer elemento que ingreso) de la fila y ademas lo elimina de la fila.

Size
Retorna el tamaño de la fila.

isEmpty
Retorna rue si la fila esta vacia de lo contrario retorna false

TAD <tabla hash>



*$(k1, v) \in T \rightarrow \forall w\,(k1, w) \notin T$

*<TableInsert>: <Elemento> -> <Tabla Hash>
*<TableRetrieve>: <Key> -> <Elemento>
*<Hash>: <Key, Integer> -> <Integer>
*<CreateTable>:<> -> <Tabla Hash>

TableInsert

Inserta un nuevo elemento a la tabla hash.

TableRetrieve

Retorna el elemento que tenga la clave que se ingreso, si no existe se retorna nulo.

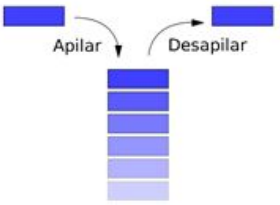
CreateTable

Crea una nueva tabla hash con todas las posiciones vacias.

Hash

Con la clave y el integer se forma un nuevo integer utilizando la función hash.

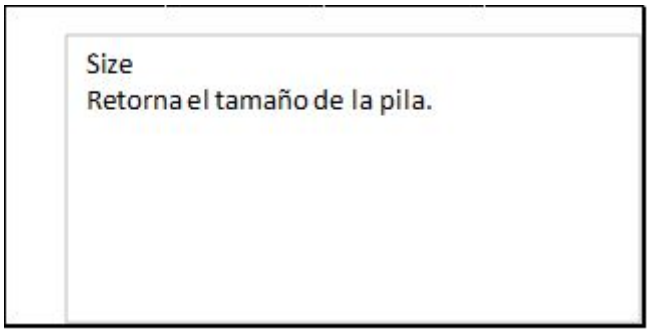
TAD <Pilas>

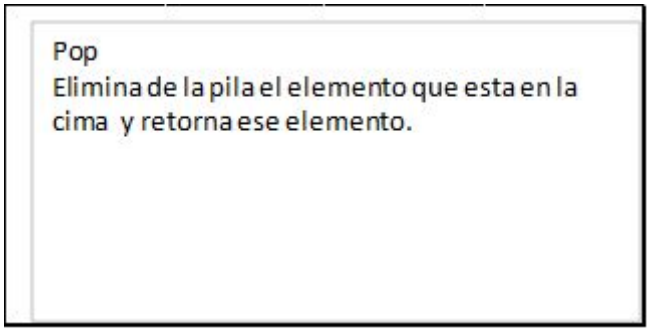


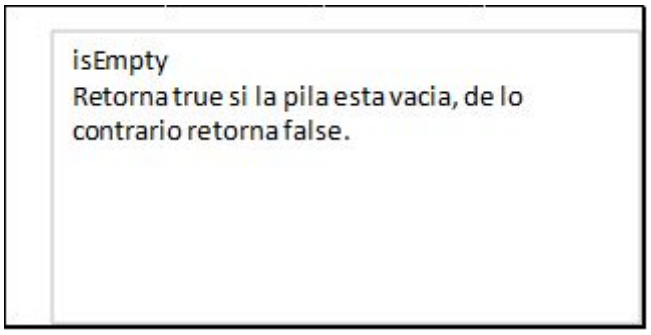
*<Push>: <Elemento> -> <Pila>
*<Pop>: <> -> <Elemento>
*<Peek>: <> -> <Elemento>
*<isEmpty>: <> -> <Boleano>
*<Size>: <> -> <Integer>

Push
Inserta un nuevo elemento a la pila, este queda en la "Cima".

Peek
Retorna el elemento que esta en la cima.







c. Diseño de pruebas

Clases	Metodos	Valores de entrada	Resultado	Escenario
Hippodrome	public void addJockey(String jockey, String horse) public String namesJockey()	Se agregan 2 jockey con el nombre camilo y camilo2 y sus caballos son pony y pony 2 respectivamente.	son equivalentes, es decir se agregan los jockeys de forma correcta y se imprime la información de la manera deseada.	Escenario 1

Clases	Metodos	Valores de entrada	Resultado	Escenario
Hippodrome	public void addJockey(String jockey, String horse) public QueueGeneric<Jockey> outcome	Se agregan 2 jockey con el nombre camilo y camilo2 y sus caballos son pony y pony 2 respectivamente.	son equivalentes, es decir la cola de resultados si posee todos los jockeys que participaron	Escenario 1

Clases	Metodos	Valores de entrada	Resultado	Escenario
Hippodrome	public void addJockey(String jockey, String horse) public String podio( )	Se agregan 3 jockey con el nombre camilo, camilo2 y camilo 3 y sus caballos son pony, pony2 y pony 3 respectivamente.	son equivalentes, se retorna el reporte del podio esperado, es decir el método funciona correctamente	Escenario 1

Clases	Metodos	Valores de entrada	Resultado	Escenario
Hippodrome	public void addJockey(String jockey, String horse) public void addBettor(String id, String name, int money, String horse)	Se agrega el jockey Felipe con el caballo pony, y se agrega el apostador Jaime con el id 1006015105, apostará $1000 por el caballo pony, se usa el método retrieve de la tabla de apostadores, se buscará eliminar el apostador Jaime.	son equivalentes, es decir el apostador se agregó de forma correcta en la tabla.	Escenario 1

Clases	Metodos	Valores de entrada	Resultado	Escenario
Hippodrome	public void addJockey(String jockey, String horse) public Jockey search(String horse)	Se agregan 2 jockey con el nombre camilo y camilo2 y sus caballos son pony y pony 2 respectivamente.	son equivalentes, es decirel método buscar retorna el jockey esperado	Escenario 1

Clases	Metodos	Valores de entrada	Resultado	Escenario
QueueFunction	public boolean offer(T t)	Se agregan los jockeys camilo y camilo2 con caballo pony y pony 2 respectivamente.	son equivalentes, es decir el método offer si esta agregando los jockeys a la cola.	Escenario 1

Clases	Metodos	Valores de entrada	Resultado	Escenario
QueueFunction	public T peek()	Se agregan los jockeys camilo y camilo2 con caballo pony y pony 2 respectivamente.	Son equivalentes, es decir se retorna el primer elemento que se agregó.	Escenario 1

Clases	Metodos	Valores de entrada	Resultado	Escenario
QueueFunction	public T pool()	Se agregan los jockeys camilo y camilo2 con caballo pony y pony 2 respectivamente.	son equivalentes, es decir si se elimina el primer elemento que se agrego a la cola.	Escenario 1

Clases	Metodos	Valores de entrada	Resultado	Escenario
QueueFunction	public T pool()	Se agregan los jockeys camilo y camilo2 con caballo pony y pony 2 respectivamente.	son equivalentes, es decir si se retorna le primer elemento que se agrego a la cola.	Escenario 1

Clases	Metodos	Valores de entrada	Resultado	Escenario
Stack	public Jockey peek()	Se agregan 2 jockeys a la pila el jockey camilo con caballo pony y el jockey Jaime con el pony2.	son equivalentes, es decir se retorna el último elemento en ingresar.	Escenario 1

Clases	Metodos	Valores de entrada	Resultado	Escenario
Stack	public Jockey pop()	Se agregan 2 jockeys a la pila el jockey camilo con caballo pony y el jockey Jaime con el pony2.	son equivalentes, es decir se retorna el ultimo elemento que se agregó. En este caso se hicieron 2 pop y por eso se retorna el primer elemento que se agrego	Escenario 1

Clases	Metodos	Valores de entrada	Resultado	Escenario
Stack	public Jockey pop()	Se agregan 2 jockeys a la pila el jockey camilo con caballo pony y el jockey Jaime con el pony2.	son equivalentes, lo que quiere decir que el método elimina el ultimo elemento que se agrego	Escenario 1

Clases	Metodos	Valores de entrada	Resultado	Escenario
HashTable	public void tableInsert(K key, V value) public V tableretrieve(K key)	Se ingresan dos apostadores, el primero es Camilo, con id 3 y $1000, el segundo tiene de nombre nno funciono, id 4 y $1000, los dos se ingresan a la tabla con la clave 3.	son equivalentes, el nombre camilo con el elemento que fue retirado de la tabla (elemento clave 3), esto quiere decir que se agrego a camilo y no al otro ya que camilo ya tenía es posición. El método funciona correctamente	Escenario 1

Clases	Metodos	Valores de entrada	Resultado	Escenario
HashTable	public int hash(K key, int i)	Los valores de entrada son 3 como clave y 0 como el elemento int	son equivalentes, es decir la función hash esta arrojando los resultados correctos.	Escenario 1

Clases	Metodos	Valores de entrada	Resultado	Escenario
HashTable	public void tableInsert(K key, V value) public V tableretrieve(K key)	Se añaden 3 nuevos apostadores Camilo, Felipe y Jaime con las identificaciones 3, 2 y 5 respectivamente y cada uno con $1000. Estos elementos son insertados en la tabla (la clave es el id de cada apostador), se pide eliminar de la tabla el elemento que tenga la clave 2	son equivalentes, Es decir se elimina de la tabla el elemento con clave 2 y ese es Felipe. El método funciona correctamente.	Escenario 1

d. Diagrama de clases

Anexo en la carpeta