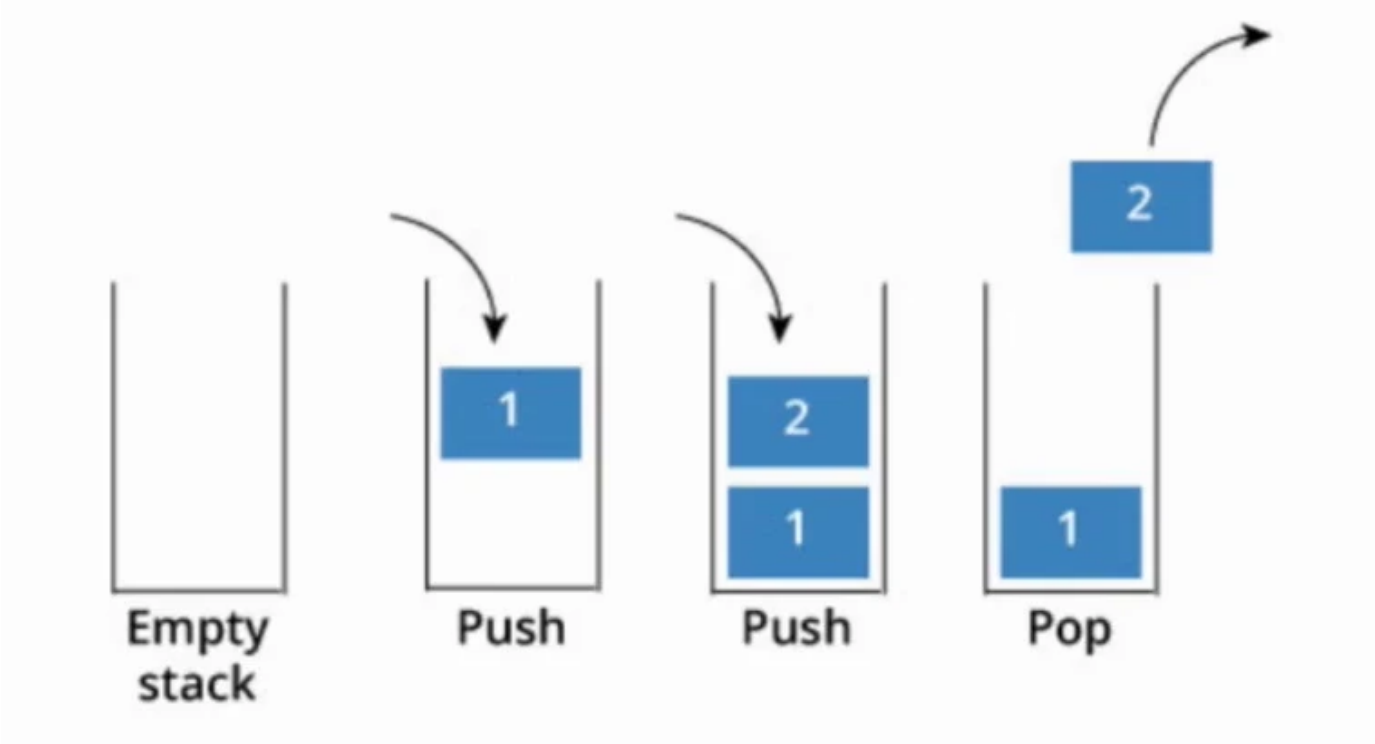
Actividad Collections. ArrayList. Pila

Actividad Collections. ArrayList. Pila

Una pila es una estructura de datos con un único punto de entrada/salida. También es llamada estructura LIFO (*Last In First Out*), es decir, el último elemento en entrar es el primero en salir.



Las pilas tienen muchas aplicaciones prácticas, ya que permiten resolver problemas compuestos. Por ejemplo:

$3 * [5 + (7 - 2)]$

Esta operación es compuesta, ya que requiere resolver subproblemas en un orden concreto.

Existen muchas situaciones similares que señalan la pila como solución idónea.

Un tipo de datos *Pila* cuenta con los siguientes métodos:

- Un método de creación. En Java, nos referimos al constructor de la clase. La operación de creación de la pila inicia la pilacomo vacía.
- *push*: pondrá un nuevo elemento en la parte superior de la pila.
- *pop*: extrae el elelemento superior de la pila.
- *top*: devuelve el valor del elemento en la cima de la pila, pero no lo extrae.
- *pilaVacia*: Esta operación es necesaria paraverificar la existencia de elementos de la pila.

Vamos a implementar una estructura pila de números enteros sin utilizar Collections, llamada *PilaArray* y una estructura pila utilizando ArrayList, llamada *PilaCollections*.

Para probar que cada pila es correcta, hacer las siguientes pruebas:

- Una pila donde no se ha insertado nada, está vacía (pilaVacia == true).
- Al insertar (push) los valores 1, 2 y 3:
  1. La cima (top) devuelve 3
  2. La cima (pop) devuelve 3
  3. La cima (top) devuelve 2
  4. La cima (pop) devuelve 2
  5. La cima (pop) devuelve 1
  6. la pila está vacía (pilaVacia == true)

Actividad Collections. Pilotos

La clase RankingPilotos mantiene una lista de pilotos participantes en una carrera automovilística. Dicha clase implementa los siguientes métodos:

```
void añadirPiloto(Piloto piloto, int posicionDeSalida)
boolean adelantarPiloto(String nombrePiloto)
boolean retrasarPiloto(String nombrePiloto)
void descalificarPiloto(String nombrePiloto)
void ordenarPilotosPorPosicionDeSalida()
void ordenarPilotosPorNombre()
String toString()
```

La clase Piloto incluye varios campos:

```
nombre
escudería
posicionDeSalida
descalificado
```

> Cuando un piloto es descalificado es enviado al final de la lista de pilotos y no puede ser adelantado.

> Ningún piloto no descalificado puede estar por detrás de un piloto descalificado.

Implementa un programa que mantenga la lista de pilotos según las operaciones realizadas sobre RankingPilotos. Los datos iniciales sobre los pilotos son los siguientes:

Posición de salida	Nombre	Escudería
1	Bottas V.	Mercedes
2	Vettel S.	Ferrari
3	Verstappen M.	Red Bull

Posición de salida	Nombre	Escudería
4	Leclerc C.	Ferrari
5	Hamilton L.	Mercedes
6	Albon A.	Red Bull
7	Sainz Jr. C.	McLaren
8	Norris L.	McLaren
9	Ricciardo D.	Renault
10	Gasly P.	Toro Rosso
11	Hülkenberg N.	Renault
12	Magnussen K.	Haas
13	Kvyat D.	Toro Rosso
14	Stroll L.	Racing Point
15	Grosjean R.	Haas
16	Giovinazzi A.	Alfa Romeo
17	Räikkönen K.	Alfa Romeo
18	Russel G.	Williams
19	Kubica R.	Williams
20	Pérez S.	Racing Point

Los sucesos que deben representarse correctamente son los siguientes:

- Los pilotos son ordenados según su nombre.
- Los pilotos vuelven a ser ordenados según su orden de salida.
- El piloto Hamilton L. adelanta dos posiciones hasta la tercera posición y Magnussen K. tres posiciones hasta la novena.
- Los pilotos Albon A. y Grosjean R. son descalificados.
- Los pilotos vuelven a ser ordenados según su orden inicial de salida, teniendo en cuenta que los decalificados no aparecen.

Tras cada suceso debes mostrar un informe con el contenido de *RankingPilotos*. En los informes del contenido de RankingPilotos debe indicarse: a) La posición actual de cada piloto. b) Su posición inicial de salida. b) Sus datos básicos (nombre y escudería).