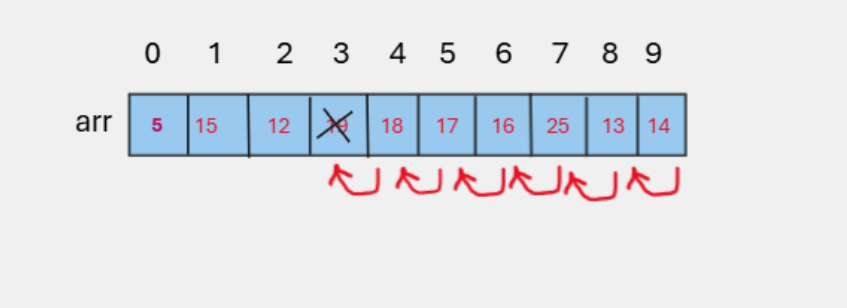
هنتكلم النهاردا عن ال **arrayList** بس قبل م نبدأ محتاجين نراجع بعض المعلومات عن ال array العادي 👍

ال array :-

قولنا ان دا عباره عن pointer بيشير لمكان معين ف الذاكره (عنوان اول عنصر ف الاراي) المكان دا والاماكن اللي جمبه حسب ال size اللي انا مدياه للاراي بيبقوا محجوزين اقدر اخزن فيهم ال values اللي انا عاوزاها ولازم القيم دي تكون من نفس الdatatype ومش بقدر اغير في ال size بالزياده ,علشان امسح حاجه او اضيف حاجه لازم همر بعمليه ال shifting (انقل مكان العناصر)

الحذف هشفت ناحيه اليمين لو العنصر ف اليمين بحيث ان اكتب عليه وبكدا تبقي القيمه القديمه مش موجوده , والاضافه هشفت برضو بس في مكان فاضي بحيث اوسع مكان للعنصر الجديد وابدا اعمله اضافه .

Remove



Adding

➕ نلم لمه بقا :

-pointer

-fixed size (الزياده ممنوعه والنقص بيعمل هدر ف الميموري)

-zero based

-the same datatype

-Contiguous places in memory

—> dataType name[]=new datatype[size];

Int arr[]= new int[5];

دي كدا نبذه سريعه فكرتنا باهم خصائص ال Array.

ملحوظه صغيره نركنها علي جمب بما اننا اخدنا ال object

لما كنا بنحب نشغل دماغنا ونضحك علي الاراي نخليه يشيل اكتر من داتا تايب كنا بنعمل اي ؟؟

برافوو 😂 نخليه يشيل objects يعني نعمله array of objects

ArrayList

- اول حاجه انها class تبع java.util.package
- من اهم مميزاتها انها dynamic or resizable array بمعني ان انا اقدر اتحكم ف ال size بتاعها عادي لانها مش fixed size زي ال array العادي
- وبالتالي لما اجي اضيف او امسح هي هتتعامل مع الحته دي بقا لوحدها وتعمل شيفت براحتها منغير م اعمل كلاسيز تقوم بالمهمه دي
- عباره عن array of objects من الكلاس اللي اسمه arrayList انا بس كل اللي عليا ان اعرف منه اوبجيكت واشتغل

علشان اعرف اي اوبجيكت كنا بنعمل اي ؟

```
Classname objectname = new classname ();
ArrayList list = new arraylist();
```

دا كدا list هتشيل عناصر من النوع objects واحنا عارفين ان الاوبجيكت دا عباره عن بوينتر(new : بتحجز مكان في ال heab - المحتوي نفسه - , و مكان في ال stack -العنوان - , واسم الاوبجيكت بيشاور علي العنوان)

يعني هو هيشيل non primitive data type

يعني ميقدرش يشيل int , float , double ... لانهم premitive يعني بيشيل فاليو وبيشاور علي الفاليو دي مش شايل عنوان

طب لو عايزاه يشيلهم ؟؟ اخليه يبقي اوبجيكت واخزن فيه اللي انا عايزاه عادي برااافووو 👏

ومن هنا جت فكره ال rabberclass كل نوع ليه كلاس بيعبر عنه وفي نفس الوقت بقدر اخزن فيه الpremitive عادي و هيبقي عادي استخدمه ف ال list و في نفس الوقت هو شايل الحاجه اللي انا عايزاها

Int → integer

Float→ float

Double → double

Bool → Boolean

Char → Character

هعمل انجل براكتس كدا اكتب فيها النوع بالظبط وانا بعرف زي كدا :

ArrayList < الرابير كلاس للنوع اللي انا عايزاه > list = new arraylist();

ArrayList < Integer >list = new arraylist();

كدا الاراي ليست فيها اوبجيكتس الاوبجيكت دا شايل قيم int

وفي كمان طرق تعريف زي كدا

```
//ArrayList list = new ArrayList();
//ArrayList <Integer> list = new ArrayList();
//ArrayList <Integer> list = new ArrayList<>();
//ArrayList <Integer> list = new ArrayList<Integer>();
List <Integer> list = new ArrayList<Integer>();
```

اخر واحده هنشرها بالتفصيل تحت .

ملحوظه بالنسبه لل string

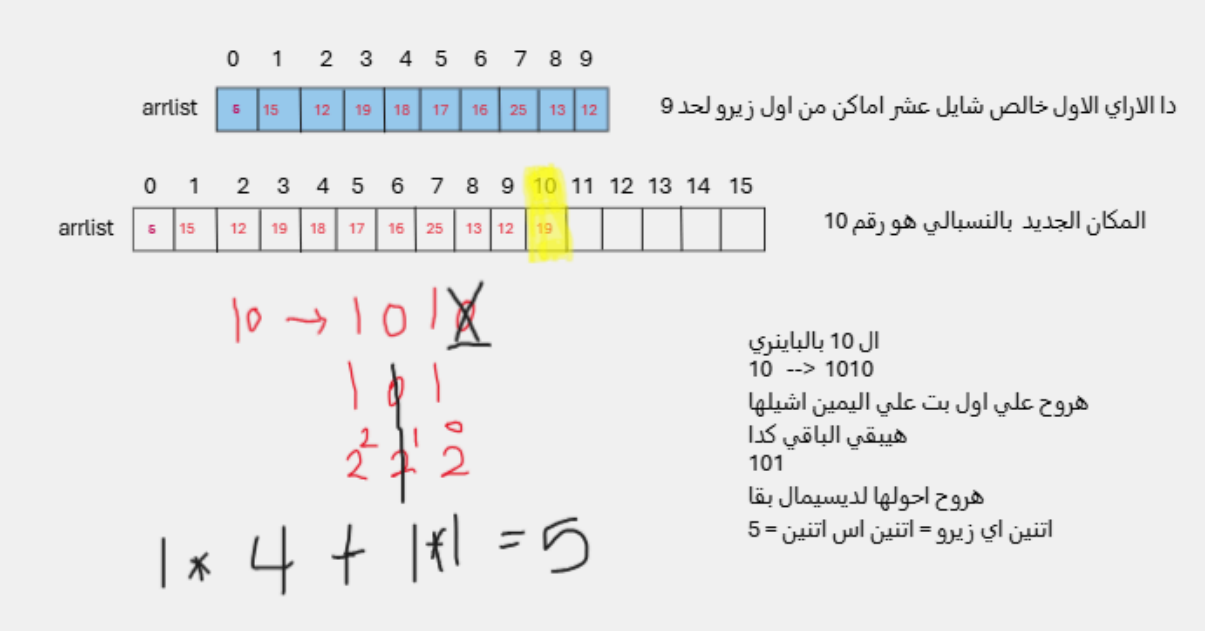
الاسترنج معروف انه كلاس ف عادي استخدمه في الاراي ليست علطول .

تكنيك شغل ال arraylist

اول م بعرف ليست بتبدا تحجز ف الميموري عشر اماكن (زي م متعرف ف ال no argument constructor) , ببدا انا بقا اخزن فيهم اللي انا عايزاه استخدم مكان اتنين تسعه ... استخدم العشر اماكن عادي براحتي بس اول م احب ازود عنصر تاني اللي هو هيبقي ال 11 هتروح الليست عامله ليست تانيه تكبر حجمها شويهيمقدار معين تخزن فيها العناصر اللي ف الاراي القديمه وتبدا تستقبل الجديد وتمسح القديمه بقا خلاص .

طب هتزود المساحه بمقدار اي ؟؟

بمقدارال (bitwise right shift operator) ل رقم عنوان اول مكان جديد بمعني



بعد م فهمنا بقا نثبت قاعده وهي لو عندي اوريدي عشر اماكن يبقي الحجم الجديد هيزيد بمقدار النص يعني هزود خمسه ف يبقي الجديد 15

- لو 15 ف النص 7.5 معندناش فكه ف يبقي 7 اذن الاجمالي 22
- لو خمسين يبقي هزود 25 ف يبقي الاجمالي 75 وهكذا ....

Arraylist hierarchy :-

- Arraylist —> abstractlist —> list (interface)

- وبالتالي ممكن اعرف بالطريقتين دول بس الفرق ان اللي فوق هيبقي متاح ليها فانكشنز اكتر
- Arratlist list = new Arraylist(); //empty constructor
- List list2 = nuw Arraylist();
- الحاجات اللي arraylist هتوفرها :-
- Access ✔️
- الوصول لاول اندكس هياخد نفس وقت الوصول لخر اندكس
- Add , remove ❌
- بتعتمد علي ال shifting ودي بتاخد وقت ف مش افضل اختيار بالنسبالي لو الجزء دا مهم في البروجرام عندي والافضل في الحته دي ال data structure linkedlist وغيرها .

🎉نلم لمه :

اتكلمنا ان الlist مش حجم معين وانها كويسه ف حته الشيفتينج لو همسح او اضيف وانها بتشيل اوبجيكتس بس وعرفنا بنعرفها ازاي طب لو عايزاها تشيل premitive datatype وعرفنا هي بتشتغل ازاي وبتزود حجمها بناء علي اي

وعرفنا الهايراركي انها بتورث من ابستراكت اللي هو اصلا بيتكون من انترفيس List .

Methods

trimToSize()

لو انا عندي 1000 مكان ف الليست وجيت اضيف ال 10001 الطبيعي ان الليست هتزود الكاباسيتي بمقدار النص يعني هتزود 500 زي م اتفقنا فوق وتبقي الكباسيتي الجديدة 1500 , انا هنا مستخدمتش غير مكان واحد من ال 500 الجداد والباقي بقي محجوز وهدر ف الميموري ف الفانكشن دي كل مهمتها انها تمسح الزياده المحجوزه وانا مش بستخدمها من خلال اي ؟؟ ال garbage collector .

Parameterized constructor (int capacity)

دا ببعتله انا عايزه كام مكان علطول علشان ازود من السرعه وال preformance بدل م كل شويه يروح يحسب هيزود كام ويعمل ليست جديده ويرجع يشيل الزياده وكل الخطوات اللي بتاخد وقت دي .

add(int num - object -)

دي بتضيفلي العناصر لليست من اليمين للشمال عادي

get(int num)

كنا ف الاراي لما نحب نمسك عنصر كنا بنكتب كدا arr[1] هنا بقا بكتب get() بدل [] البراكتس دي

```
import java.util.ArrayList;
import java.util.List;
public class Main {
public static void main(String[] args) {
//ArrayList list = new ArrayList();
ArrayList <Integer> list = new ArrayList();
list.add(10);
list.add(20);
list.add(30);
➕System.out.println(list);          // [ 10, 20, 30 ]
➕Int s = list.size();       // بيحسب عدد المشغول فعليا وبيخلي الاداء افضل
    for (int i = 0; i < s; i++) {
    System.out.println(list.get(i)); //10 20 30
     }
➕System.out.println (list[1000]) ;   //بيدعم ال random access
```

Add (int index , Integer element)

دي بنبعتلها المكان والقيمه

addAll(collection)

ممكن اباصي list او شويه عناصر جمب بعض

بس لو هحط عناصر لازم احولها ل list الاول من

Arrays.asList(واحط العناصر عادي)

list2.add(2); //2

List2. addAll(Arrays.asList(66, 88, 99) ; //2 ,66 ,88 ,99

addAll(indix , collection)

مش بس هبعت العناصر لا هبعت مكانها كمان

List2. addAll(3 , Arrays.asList(16, 18) ;

// 2 , 66 ,88 ,16 , 18 ,99 عمل شيفت لل 99 عادي

ArrayList <Integer> list = new ArrayList(Arrays.asList(16, 18));

كوبي كونستراكتور

معناه بكل بساطه ان انا بباصي الارقام علطول واقوله اعملهم لست مع بعض ودا وانا بعرف ال list اصلا

Clone () بتعمل نسخ لعناصر ا ل list وبترجع اوبجيكت

List2=(ArrayList) list1.clone();

انسخ العناصر اللي ف الليس الاولي وحطهم ف الليست التانيه بعد م تعمل كاستينج للاوبجيكتس وتخليها اراي ليست

وهما في اماكن تانيه اتحجزت ف الميموري لل list 2 وبقا ليها عنوان خاص بيها (shallow copy)

ف لو حصل تاثسر علي الليست الاولي التانيه مش هتحس بيسه عكس م هعمل كدا

List2 = list 1;

هنا بقا copy by referance يعني هيخلي ال list2 تشاور علي نفس مكان الاولي وبالتالي لو حصل تعديل علي الاولي التانيه هتحس بيه والعكس لو عدلت علي التاني الاول هيحس بيه برضو

set(int indix , object)

لو عايزه اعمل اوفررايت علي مكان معين بمعني لو عندي الاندكس 3 فيه 5 وانا عايزه اغير ال5 اخليها 9 بستخدم الفانكشن دي .

ومن هنا نروح لعمليات الحذف remove

remove(object) ببعتله الحاجه اللي عايزه امسحها علطول

List1. remove(10); //error

list1.remove(new integer(10)); //عملت اوبجيكت وباصيت الرقم ف الكونستراكتور

list1.remove((Integer) 10);// عملت كاستينج وبقي اوبجيكت من الكلاس دا

list1.remove((Integer.valueof(10)); //الميثود هترجع اوبجيكت

ملحوظه امتي اباصي الرقم بقا وميدينيش ايرور ؟

لو كانت الlist من النوع string لانه اوريدي كل رقم عباره عن اوبجيكت

```
ArrayList < string > list1 = new ArrayList( Arrays.asList( "16", "18" ) );
List1. remove("18");   //هيشتغل عادي
```

Remove (int index) ببعت العنوان علطول

Remove All(collection)

قولنا الكوليكشن عباره عن ليست او مجموعه ارقام متحولين لليست من خلال

Arrays.asList();

و دا معناه انه هيمشي علي الكولكشن اللي انا بعته يحذف منه الاوبجيكت اللي انا محدداه.

نشوف مثال

```
ArrayList < string > list = new ArrayList( Arrays.asList( "16", "18" ,"12"  , "13" , "16" ) );
List. removeAll(Arrays.asList(" 16" , " 12")); // [18  ,13 ]
```

contains(Object o)

ودي بتشوف لو الاوبجيكت دا موجود عندي ف ال list ولا لا وبترجع قيمه ترو او فولس

arrayList.contains(element); بتتكتب كدا

مثال 👍

```
ArrayList<String> names = new ArrayList<>();
names.add("Ali");
names.add("Sara");
```

```
System.out.println(names.contains("Sara"));  // true
System.out.println(names.contains("Omar"));  // false
```

retainAll(Collection)

الميثود دي بقا ببعتلها list فهي تشوف العناصر اللي انا بعتها وتقارن بال list الاصليه اللي موجود في الاصليه من اللي انا بعتها تخليه واللي يتبقي في الالصليه مش في اللي انا بعتها تمسحه .

مثال 👍

```
ArrayList<String> list1 = new ArrayList<>(List.of("A", "B", "C"));
ArrayList<String> list2 = new ArrayList<>(List.of("B", "C", "D"));

list1.retainAll(list2);  // يحتفظ فقط بـ B و C
System.out.println(list1);  // [B, C]
```

subList(int fromIndex, int toIndex)

دي بقا اكني بقص جزء من ال list الاصليه بقوله قصلي الlist من الاندكس الفلاني لحد ما قبل الاندكس الفلاني

مثال 👍

```
ArrayList<String> items = new ArrayList<>(List.of("One", "Two", "Three", "Four"));
List<String> sub = items.subList(1, 3);  // يأخذ العناصر من index 1 إلى قبل 3

System.out.println(sub);  // [Two, Three]
```

clear()

دي بقا اسهل واحده بتحذف كل الاوبجيكتس اللي ف ال list .

مثال 👍

```
ArrayList<Integer> numbers = new ArrayList<>(List.of(1, 2, 3));
numbers.clear();
System.out.println(numbers);  // []
```

ensureCapacity(int minCapacity)

فاكرين لما كنا بنحدد عدد الاماكن في الليست واحنا بنعملها انشياليزيشن , هنا هنعمل نفس الكلام اكني بقوله اتاكد ان الليست دي تقدر تشيل لحد 100 عنصر علشان اقلل من الخطوات اللي اتكلمنا فيها لما الليست تحب تكبر نفسها وبالتالي يبقي الاداء ف البرنامج احسن .

مثال 👍

```
ArrayList<Integer> numbers = new ArrayList<>();
numbers.ensureCapacity(100);
```

sort(Comparator)

دي هنرتب العناصر تصاعدي او تنازلي لو تصاعدي هستدعي الفانكشن عادي واباصي ال list عادي لو تنازلي هرتبهم برضو تصاعدي واعملهم reverse

ف ك كودينج هستخدم الفانكشن دي(Collections.reverseOrder

مثال 👍

```
ArrayList<Integer> nums = new ArrayList<>(List.of(5, 2, 8, 1));
Collections.sort(nums);  من كلاس الكولكشن استدعيت الفانكشن وباصيت الليست عادي
System.out.println(nums);  // [1, 2, 5, 8]
Collections.sort(nums, Collections.reverseOrder());  // [8, 5, 2, 1]
```

هنا باصيت الليست والميثود ال reverse

اخر حاجه خلاص 😂

Collections.min(Collection)

من اسمها بترجع القيمه الاصغر

```
ArrayList<Integer> nums = new ArrayList<>(List.of(10, 3, 7));

int minVal = Collections.min(nums);
```

```
System.out.println(minVal);  // 3
```

Collections.max(Collection)

من اسمها برضو بترجع قيمه اكبر عنصر
نفس المثال 👍

```
int maxVal = Collections.max(nums);
System.out.println(maxVal);  // 10
```