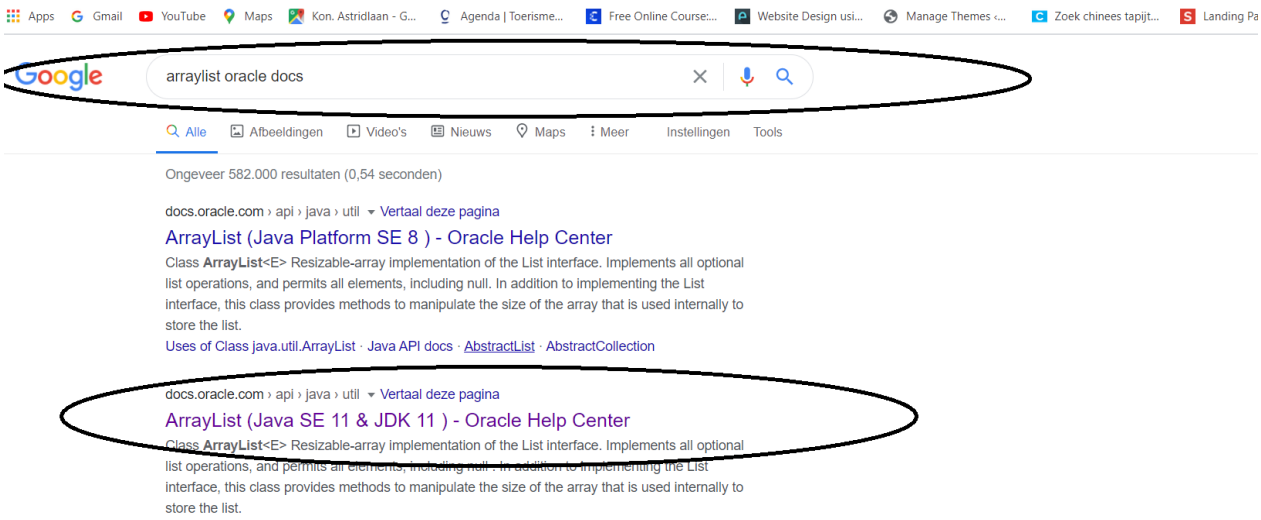
Bibliotheekklassen.

Het Java platform heeft een uitgebreid assortiment bibliotheken waarvan ervaren programmeurs continu gebruik maken.

Zo gebruikten we reeds de package java.util om gebruik te kunnen maken van de klasse ArrayList.
De belangrijkste methodes van deze klassen zijn add, get, iterator, remove en size.
De methods add, get en size hebben jullie al veel gebruikt.

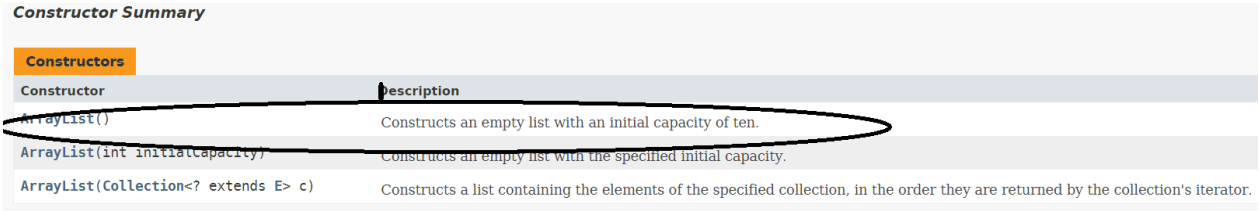
Hoe kan je deze info nu makkelijk opzoeken op het internet?
Belangrijk bij het vinden van info op het internet is, dat je weet waar je naar op zoek gaat.
Daarom is het belangrijk om basiskennis programmeren te hebben en een dosis parate kennis.

TO DO : Zoek op *arraylist* en geef als extra keywords mee : *oracle docs.*
Kies de hoogste versie.



Vervolgens kom je op de help-pagina's van de klasse ArrayList.
Op dit moment zal je zeker niet alles begrijpen wat hier staat, maar dat geeft niet.
Wat interessant is, zijn de methods die je kan gebruiken van deze klasse.

In een eerste overzicht staan alle mogelijke constructoren. Wij gebruikten in Programming Fundamentals steeds de eerste.

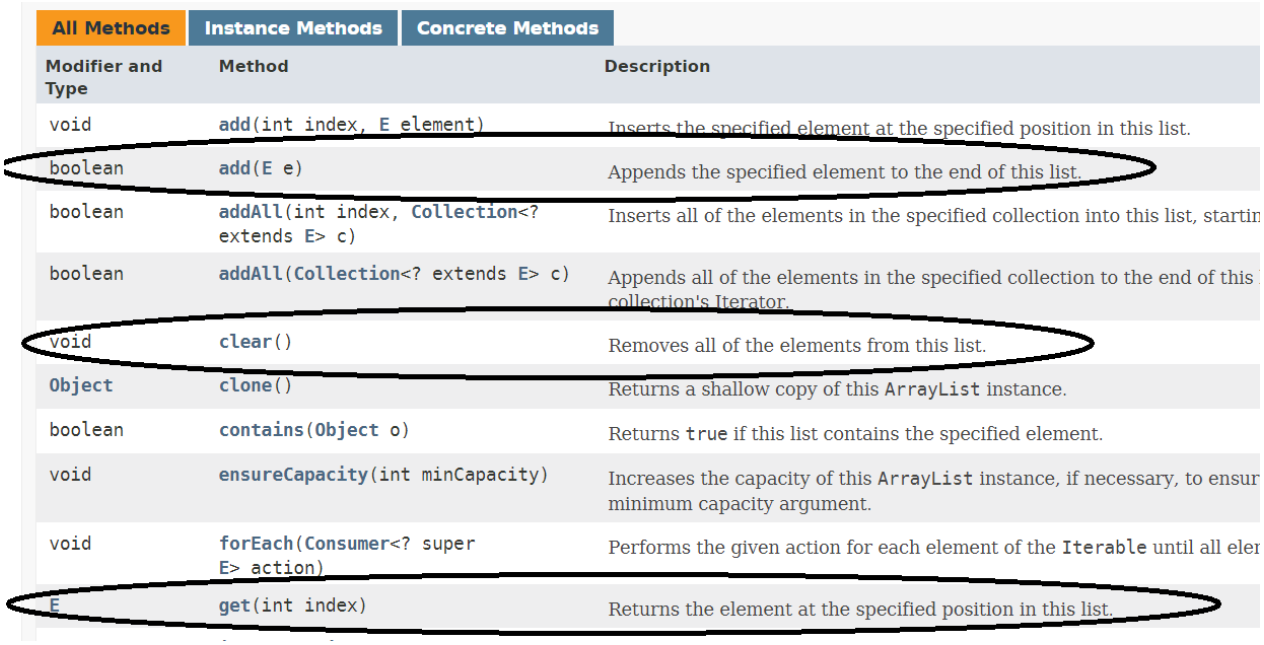


In een tweede overzicht staan alle methoden in alfabetische volgorde die je kan gebruiken om een ArrayList te manipuleren.

Zo kennen we reeds de volgende functies.

add => deze functie heeft als parameter een object van klasse E om toe te voegen aan de ArrayList.
De functie add returnt een boolean, true indien gelukt.

Bekijk ook get en clear.



All Methods | Instance Methods | Concrete Methods

Modifier and Type	Method	Description
void	add(int index, E element)	Inserts the specified element at the specified position in this list.
boolean	add(E e)	Appends the specified element to the end of this list.
boolean	addAll(int index, Collection<? extends E> c)	Inserts all of the elements in the specified collection into this list, startin
boolean	addAll(Collection<? extends E> c)	Appends all of the elements in the specified collection to the end of this collection's Iterator.
void	clear()	Removes all of the elements from this list.
Object	clone()	Returns a shallow copy of this ArrayList instance.
boolean	contains(Object o)	Returns true if this list contains the specified element.
void	ensureCapacity(int minCapacity)	Increases the capacity of this ArrayList instance, if necessary, to ensur minimum capacity argument.
void	forEach(Consumer<? super E> action)	Performs the given action for each element of the Iterable until all eler
E	get(int index)	Returns the element at the specified position in this list.

De klasse Random

De volgende klasse die we uitvoerig bespreken en zullen gebruiken in de volgende oefeningen is de klasse Random. Ook Random maakt deel uit van de package java.util. Met deze klasse kan je willekeurige getallen laten genereren.

TO DO : zoek deze klasse op in de documentatie.

De belangrijkste methods zijn de methoden nextInt die een willekeurig positief getal leveren, eventueel binnen een bepaald bereik.

Een willekeurig getal tussen 0 en 19 maak je als volgt - je moet Random daartoe importeren.

Onderstaande code toont hoe je een willekeurig getal genereert tussen 0 en 19.

```
Random r = new Random();
int i = r.nextInt(20);
```

Oefening 1
Maak een functie die een getal genereert van 1 tot een bepaald getal. Dit getal is een parameter.

Oefening 2
Maak een functie die een getal genereert tussen een opgegeven min en max. Min en max zijn parameters.

Oefening 3
Elke woensdag- en zaterdagavond is er een Lotto-trekking.
6 nummers en 1 bonusnummer worden getrokken uit 45 genummerde ballen.
Schrijf hiervoor een klasse Lotto.

TIP
Maak een functie trekking die een ArrayList<Integer> van 7 getallen returnt.
Toon met sout het bonus getal (=het laatste getal van de ArrayList).

Je zal dus gebruik moeten maken van Wrapper-klassen.
Hieronder de volledige uitleg.

Wrapper-klassen.

Zoals we gezien hebben kan een ArrayList alle soorten objecten opslaan. Echter geen enkelvoudige datatypes zoals int, boolean, double.

Er zijn echter situaties waarin we toch integers willen opslaan in een ArrayList.
De Java-oplossing voor dit probleem zijn Wrapper-klassen.

Elk primitief datatype heeft in Java een overeenkomstige wrapper klasse die hetzelfde type vertegenwoordigt, maar een objecttype is.
Voor int is dat de wrapper-klasse Integer.

Hoe gaan we dan te werk voor bovenstaande oefening?
Voorbeeld:

```
import java.util.ArrayList;

public class Wrapper {
    public static void main(String[] args) {
        ArrayList<Integer> lijst = new ArrayList<>();
        int getal = 5;
        lijst.add(getal);
    }
}
```

Primitief type	**Wrappertype**
Boolean	Boolean
Double	Double
Char	Char
Int	Integer

Oefening 4
Maak een raad spelletje.
Werk met JOptionPane, zowel voor de input als voor de output.

Stap 1
Maak een functie **public int speelRaadspel()**

Deze functie genereert een getal tussen 1 en 6.
De speler moet dit getal raden. Hij krijgt hiervoor maximaal 3 kansen.

Heeft hij het getal geraden na 1 keer, dan krijgt hij 6 punten,
na 2 keer 5 punten,
na 3 keer 4 punten. Heeft hij het niet geraden, dan krijgt hij geen punten.

Tel hoeveel keer de speler in totaal geraden heeft en toon dit aantal.

De functie returnt de score.

Zorg er ook voor dat de functie moet nagaan of het ingegeven getal tussen 1 en 6 ligt. Indien niet, dan vraagt de functie om juist getal in te geven. Dit telt niet bij de 3 raad-kansen.

Stap 2

Maak een functie **public void spelen()** die gebruik maakt van de vorige functie.

Deze functie vraagt aan de speler of hij nog eens het raad-spel wil spelen.
Indien de gebruiker NEE ingeeft, is het spel gedaan.
Indien de speler JA ingeeft, wordt het raad-spel opnieuw gestart.
Na maximaal 10 keer raden, of wanneer de speler eerder wou stoppen, wordt er een eindstand gegeven.

Deze eindstand bevat :
Totaal aantal punten van alle beurten samen.
Per beurt het aantal punten. Maak hiervoor een ArrayList.
De beurten (beurt 1, beurt 2, beurt 3, …) met het hoogste aantal punten er uit halen.
Maak een ArrayList om de beurten met de maximale scores op te slaan.

Toon de volledige eindstand met JOptionPane.
Voorbeeld.

De klasse String

De String klasse maakt deel uit van de package java.lang.
Klassen in deze package zijn van fundamenteel belang voor de Java taal.
Daarom wordt deze package automatisch in elke klasse geïmporteerd.

De volgende oefeningen zijn er op gericht om zelf op zoek te gaan in de Java bibliotheken en gebruik te maken van bestaande functies.
Het is dus van cruciaal belang om te leren omgaan met de Java documentatie.
https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/lang/String.html

Buiten de officiële bronnen zijn er nog tal van andere sites waar je veel informatie kan vinden.
Voorbeelden.

https://beginnersbook.com/2013/12/java-strings/
https://www.geeksforgeeks.org/string-class-in-java/
https://www.w3schools.com/java/java_strings.asp

Op Stack overflow kan je heel veel info vinden.

Oefening 1
Schrijf een methode met signatuur public void printLowerUpperCase (String woord) die een String woord als parameter heeft en het woord eerst in kleine letters en daarna in grote letters op het scherm schrijft.

Regex – reguliere expressies

Om gemakkelijker te werken met String functies is het nuttig om te weten wat reguliere expressies zijn.
Een reguliere expressie is een zoekpatroon dat bestaat uit een aantal characters.
Hiermee beschrijf je wat je zoekt.
Verschillende oefeningen worden eenvoudiger indien je gebruik maakt van deze expressies.

De meest gebruikte characters zijn:

Construct	description	Vb regexp	matcht
.	Any character (may or may not match line terminators)	**h.llo**	hallo, hello, hullo, h9llo, hqllo, …

X?	`? staat na een char X: wil zeggen: X 0 of 1 keer`	**A?llo**	llo, Allo
X*	`* staat na een char X: wil zeggen:0 of meerdere keren`	**A*llo**	llo, Allo, AAllo, AAAllo, AAAAllo, …..
X+	`+ staat na een char X: wil zeggen:1 of meerdere keren`	**A+llo**	Allo, AAllo, AAAllo, AAAAllo, …..
\d	A digit: [0-9]	**\d\d:\d\d**	55:00, 99:16, …
\D	A non-digit: [^0-9]		
\s	A whitespace character: [\t\n\x0B\f\r]	Hallo\sjij	Hallo jij, Hallo jij, Hallo jij, …
\S	A non-whitespace character: [^\s]		
^	Begin lijn		
$	Einde lijn		
[abc]	a, b, or c (simple class)	[abc]111	a111, b111, c111
[^abc]	Any character except a, b, or c (negation)	[^abc]111	d111, E111, 3111, …

Een paar complexere voorbeelden:

Regexp	Matcht
Hallo, .*!	Hallo,! Hallo, Bart! Hallo, blablabla blablabala dsdfkjslfjsdf ! …
BE-\d\d\d-	BE-234- BE-765- BE-000- …
\d+	1 54 755432223 …
[abc]+\d	a1 b4 c7 …

Lijst met alle mogelijkheden: https://docs.oracle.com/javase/7/docs/api/java/util/regex/Pattern.html

Uitleg in meer detail:

https://docs.oracle.com/javase/tutorial/essential/regex/pre_char_classes.html
https://docs.oracle.com/javase/tutorial/essential/regex/quant.html
https://docs.oracle.com/javase/tutorial/essential/regex/char_classes.html

```java
public class Regex {
    // returns true if the string matches exactly "true"
    public boolean isTrue(String s) {
        return s.matches("true");
    }

    // returns true if the string matches exactly "true" or "True"
    public boolean isTrueVersion2(String s) {
        return s.matches("[tT]rue");
    }

    // returns true if the string matches exactly "true" or "True"
    // or "yes" or "Yes"
    public boolean isTrueOrYes(String s) {
        return s.matches("[tT]rue|[yY]es");
    }

    public boolean isThreeNumbers(String s) {
        //return s.matches("[0-9]{3}");
        return s.matches("[\\d]{3}");
    }

    // returns true if the string contains of three letters
    public boolean isThreeLetters(String s) {
        //return s.matches("[a-zA-Z]{3}");
        return s.matches("[\\D]{3}");
    }
}
```

https://regexr.com/
Op deze site kan je experimenteren.

De volgende oefeningen kan je naar keuze oplossen met of zonder regex oplossen.

Oefening 2
Schrijf een methode met signatuur public String schrapLeestekensSpaties (String zin) die een String zin als parameter heeft en deze retourneert zonder leestekens en spaties.

Oefening 3
Schrijf een methode met signatuur public void printKlinkerSpatie (String woord) die een woord als parameter heeft en het woord op het scherm schrijft waarbij iedere klinker vervangen is door een spatie.

Oefening 4
(werk met charAt())
Schrijf een methode met signatuur public boolean isPalindroom (String woord) die een woord als parameter heeft en controleert of dit woord een palindoom is. Palindromen zijn bijvoorbeeld: “pop”, “lepel”, “meetsysteem”, “Hannah”, …

Oefening 5
Schrijf een methode met signatuur public void printWoorden (String zin) die een String zin als parameter heeft en elk woord afdrukt op een nieuwe regel. Een spatie geldt als het einde van een woord.
Het is WK
Het
is
WK

Oefening 6
Schrijf een methode met signatuur public double printGemiddeldAantalLetters (String [ ] woorden) die een array van woorden als parameter heeft en het gemiddelde aantal letters per woord retourneert.

Static methods

Alle methoden die wij gezien en geschreven hebben zijn instantiemethoden.

Een **instantiemethode** wordt aangeroepen **voor een object** of een **instantie van een klasse**. We leerden:
1) Maak object.
2) object.method()

Een **klassenmethode** wordt opgeroepen **zonder eerst een object van de klasse** nodig te hebben. Het is voldoende om een klasse te hebben.
De oproep is
Klasse.method()

Voorbeeld.

```
import java.time.LocalDateTime; // Import the LocalDateTime class
import java.time.format.DateTimeFormatter; // Import the DateTimeFormatter
class

public class VoorbeeldStatic {
    public static void main(String[] args) {
        LocalDateTime myDateObj = LocalDateTime.now();
        System.out.println("Before formatting: " + myDateObj);
```

```
        DateTimeFormatter myFormatObj = DateTimeFormatter.ofPattern("dd-MM-
yyyy HH:mm:ss");

        String formattedDate = myDateObj.format(myFormatObj);
        System.out.println("After formatting: " + formattedDate);
    }
}
```

TO DO :
-Probeer de code uit.
-Zoek deze functies op in de klassebibliotheken en probeer het verschil te begrijpen tussen static en non-static.
-Welke functies zijn static en welke zijn non-static?

Meer weten?

Static – non static

Static methods are useful if you have only one instance (situation, circumstance) where you're going to use the method, and you don't need multiple copies (objects). For example, if you're writing a method that logs onto one and only one web site, downloads the weather data, and then returns the values, you could write it as static because you can hard code all the necessary data within the method and you're not going to have multiple instances or copies. You can then access the method statically using one of the following:

```
MyClass.myMethod();
this.myMethod();
myMethod();
```

Non-static methods are used if you're going to use your method to create multiple copies. For example, if you want to download the weather data from Boston, Miami, and Los Angeles, and if you can do so from within your method without having to individually customize the code for each separate location, you then access the method non-statically:

```
MyClass boston = new MyClassConstructor();
boston.myMethod("bostonURL");

MyClass miami = new MyClassConstructor();
miami.myMethod("miamiURL");

MyClass losAngeles = new MyClassConstructor();
losAngeles.myMethod("losAngelesURL");
```

In the above example, Java creates three separate objects and memory locations from the same method that you can individually access with the "boston", "miami", or "losAngeles" reference. You can't access any of the above statically, because MyClass.myMethod(); is a

generic reference to the method, not to the individual objects that the non-static reference created.

If you run into a situation where the way you access each location, or the way the data is returned, is sufficiently different that you can't write a "one size fits all" method without jumping through a lot of hoops, you can better accomplish your goal by writing three separate static methods, one for each location.

Generally

static: no need to create object we can directly call using

```
ClassName.methodname()
```

Non Static: we need to create a object like

```
ClassName obj=new ClassName()
obj.methodname();
```

Oefening 7

Schrijf een methode met signatuur public void printNamenAlfabetisch (String [ ] namen) die een array van namen als parameter heeft en deze alfabetisch op het scherm afdrukt. Zoek functies op die sorteren en een array afdrukken.
Maak ook een methode om het kortste woord te zoeken in de array en een methode om het langste woord te zoeken in de array. Beide functies returnen een arrayList van 1 of meerdere woorden (bij gelijke lengte).

Maak nu ook een method die een arraylist returnt waarin je elk uniek woord en het aantal keren dat dit voorkomt plaatst.
Tip : maak eerst een klasse aan met als private variabelen:
-woord
-aantalKeer

Oefening 8

Maak een functie die true of false returnt wanneer het bankrekeningnr geldig is.
public boolean isGeldigRekeningNummer(String reknr)
//format: BE00 0000 0000000 00
//gebruik matches-functie
//Wanneer je het getal bestaande uit 11 cijfers na BE00 deelt door 97, is de rest = aan de laatste twee cijfers.
//%
Tips.
Gebruik substring
Gebruik de functies matches om na te gaan of de bankrekeningnummer in het juiste formaat staat.

Zoek op hoe je String omzet naar long.
Gebruik long-datatype, met int kom je in de problemen. Zoek het bereik van int op.