JAVA PROGRAMMEREN – LES 3:

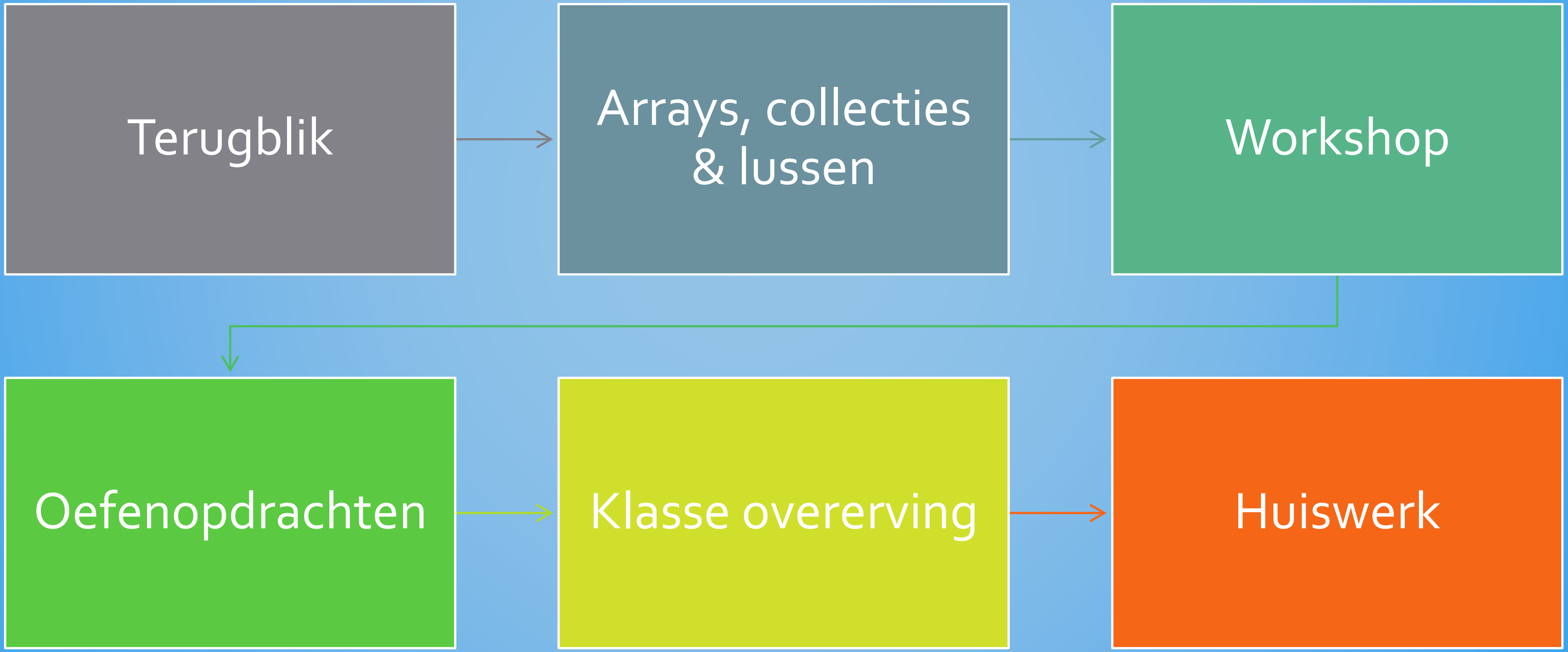
ARRAYS
COLLECTIES
LUSSEN
OVERERVING

Robert-Jan Elias

robert-jan.elias@novi-education.nl

NOVI

AGENDA



Terugblik

Arrays, collecties & lussen

Workshop

Oefenopdrachten

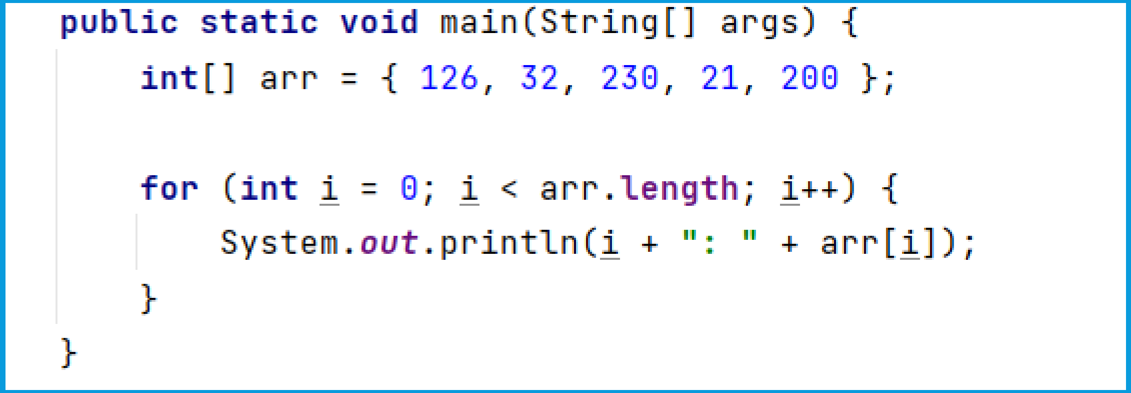
Klasse overerving

Huiswerk

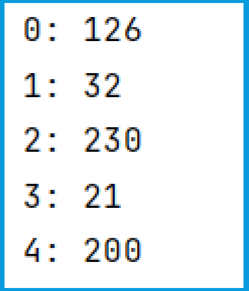
TERUGBLIK

- Klassen & objecten
- Constructors
- Getters & setters
- Relaties tussen klassen

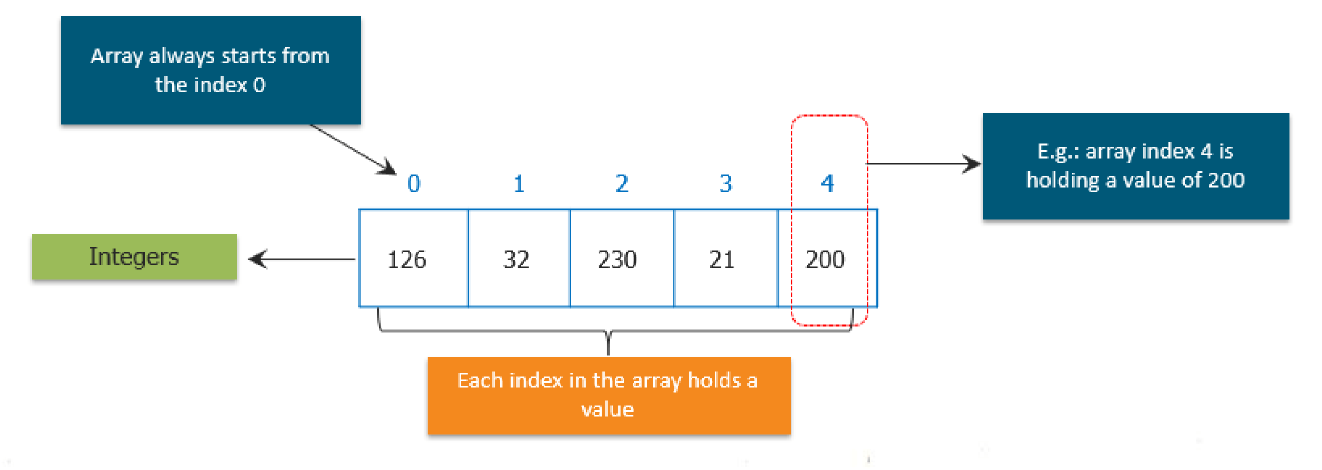
ARRAYS



```java
public static void main(String[] args) {
    int[] arr = { 126, 32, 230, 21, 200 };

    for (int i = 0; i < arr.length; i++) {
        System.out.println(i + ": " + arr[i]);
    }
}
```



```
0: 126
1: 32
2: 230
3: 21
4: 200
```



Array always starts from the index 0

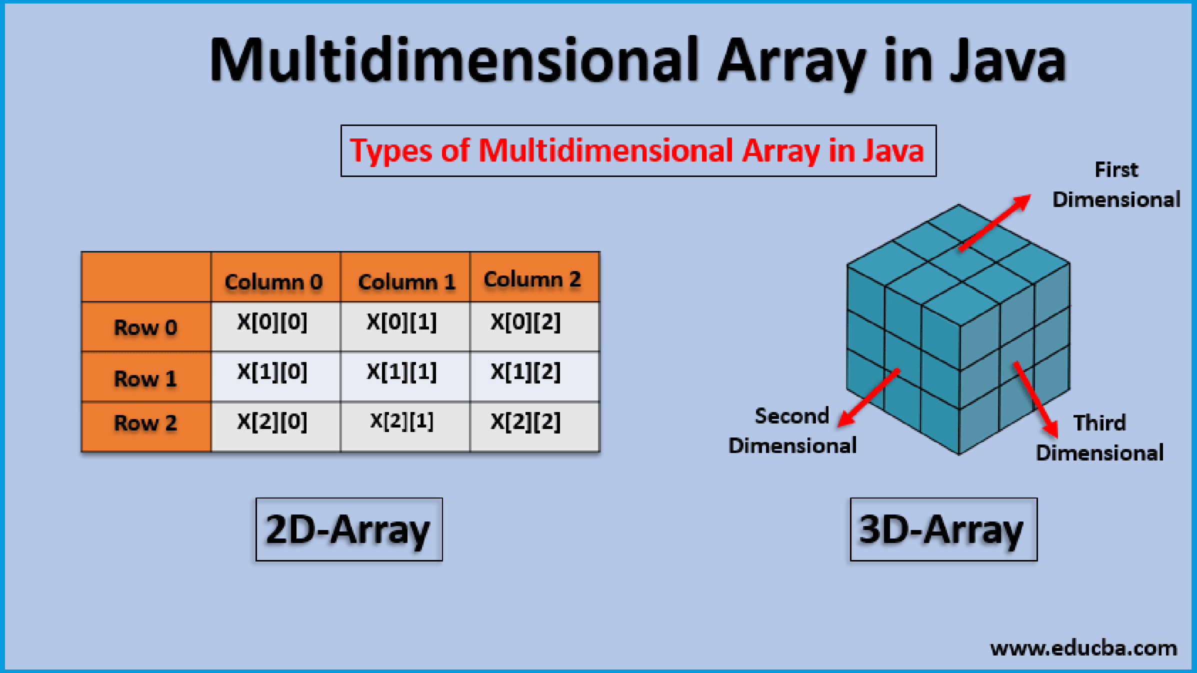
0	1	2	3	4
126	32	230	21	200

E.g.: array index 4 is holding a value of 200

Integers

Each index in the array holds a value

MULTIDIMENSIONAL ARRAYS



Multidimensional Array in Java

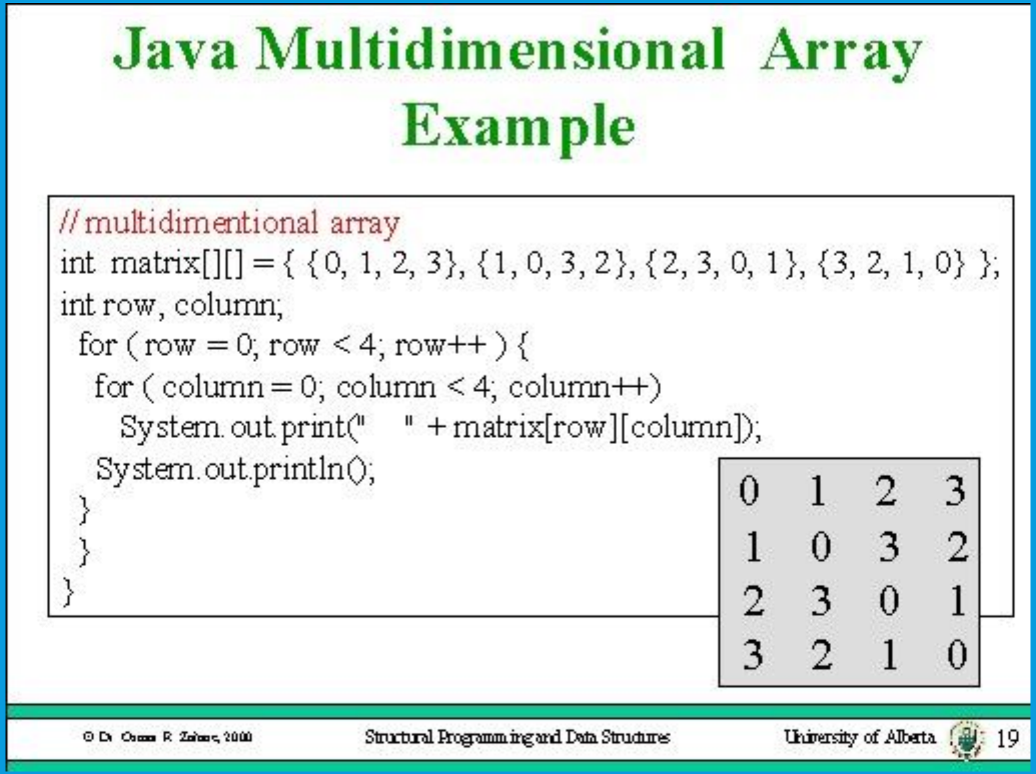
Types of Multidimensional Array in Java

	Column 0	Column 1	Column 2
Row 0	X[0][0]	X[0][1]	X[0][2]
Row 1	X[1][0]	X[1][1]	X[1][2]
Row 2	X[2][0]	X[2][1]	X[2][2]

2D-Array

3D-Array

www.educba.com



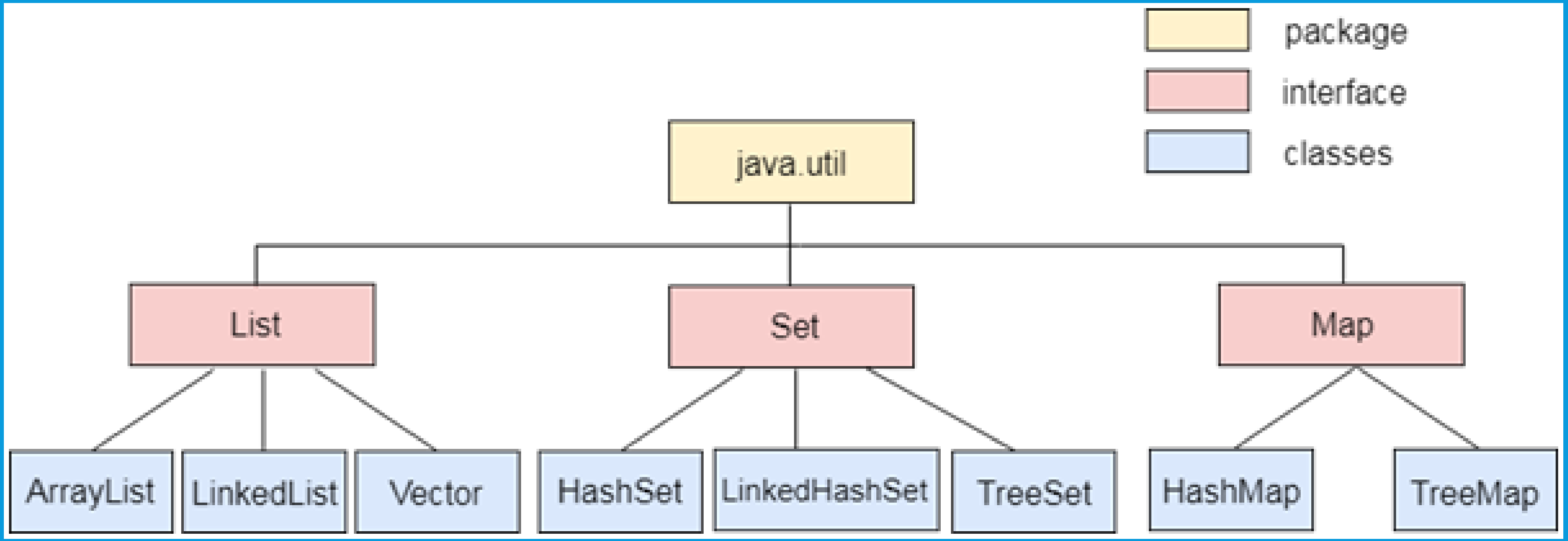
Java Multidimensional Array Example

```
//multidimentional array
int matrix[][] = { {0, 1, 2, 3}, {1, 0, 3, 2}, {2, 3, 0, 1}, {3, 2, 1, 0} };
int row, column;
 for ( row = 0; row < 4; row++ ) {
  for ( column = 0; column < 4; column++)
     System.out.print("    " + matrix[row][column]);
   System.out.println();
 }
 }
}
```

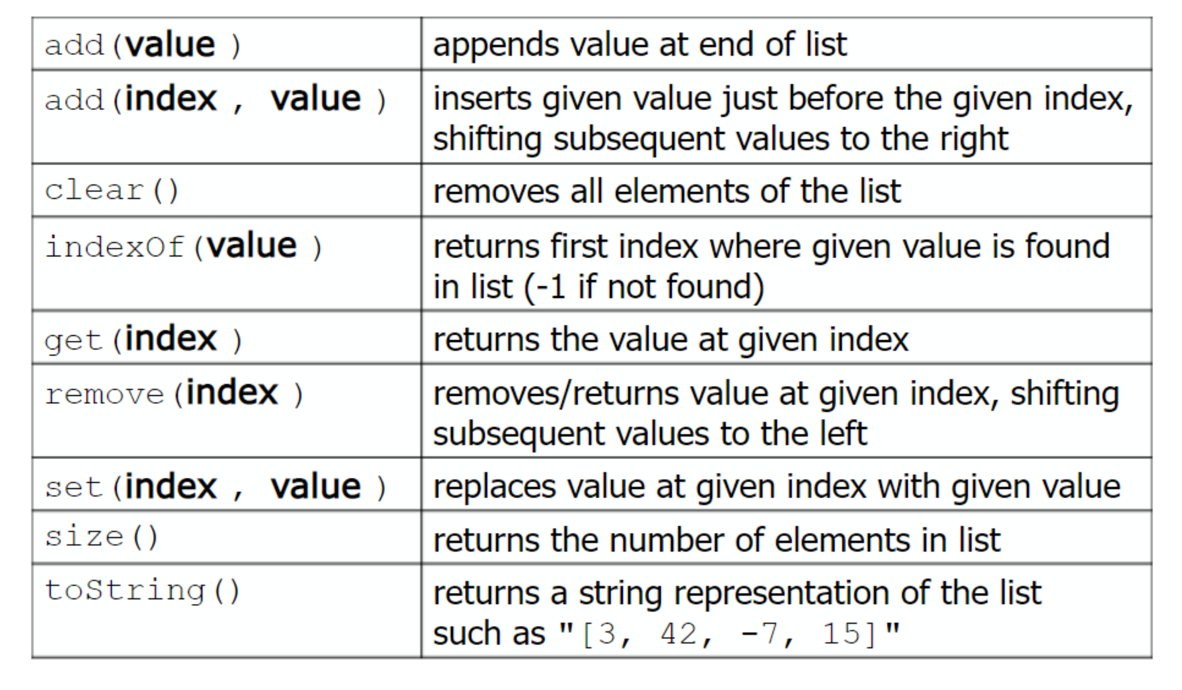
```
0  1  2  3
1  0  3  2
2  3  0  1
3  2  1  0
```

© Dr. Osmar R. Zaïane, 2000 Structural Programming and Data Structures University of Alberta 19

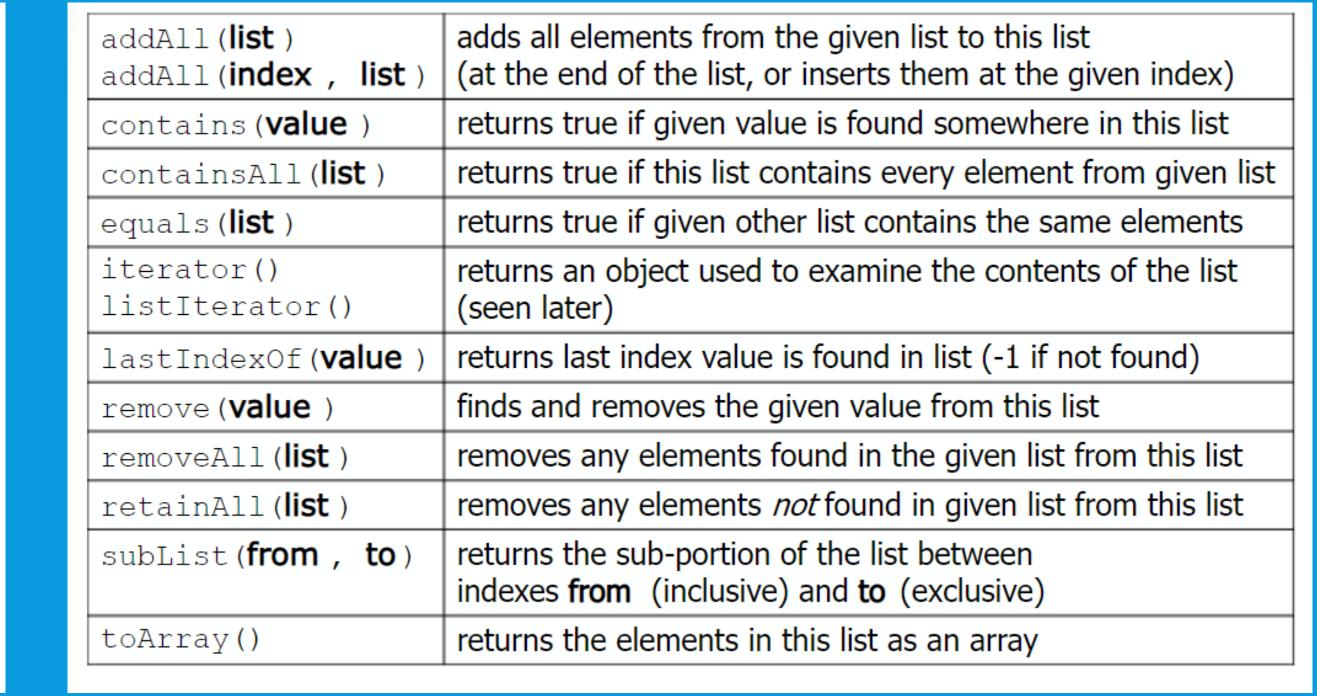
COLLECTIES


package
interface
classes
java.util
List
Set
Map
ArrayList
LinkedList
Vector
HashSet
LinkedHashSet
TreeSet
HashMap
TreeMap

ARRAYLIST METHODS



`add(`**value**`)`	appends value at end of list
`add(`**index**`,` **value**`)`	inserts given value just before the given index, shifting subsequent values to the right
`clear()`	removes all elements of the list
`indexOf(`**value**`)`	returns first index where given value is found in list (-1 if not found)
`get(`**index**`)`	returns the value at given index
`remove(`**index**`)`	removes/returns value at given index, shifting subsequent values to the left
`set(`**index**`,` **value**`)`	replaces value at given index with given value
`size()`	returns the number of elements in list
`toString()`	returns a string representation of the list such as `"[3, 42, -7, 15]"`

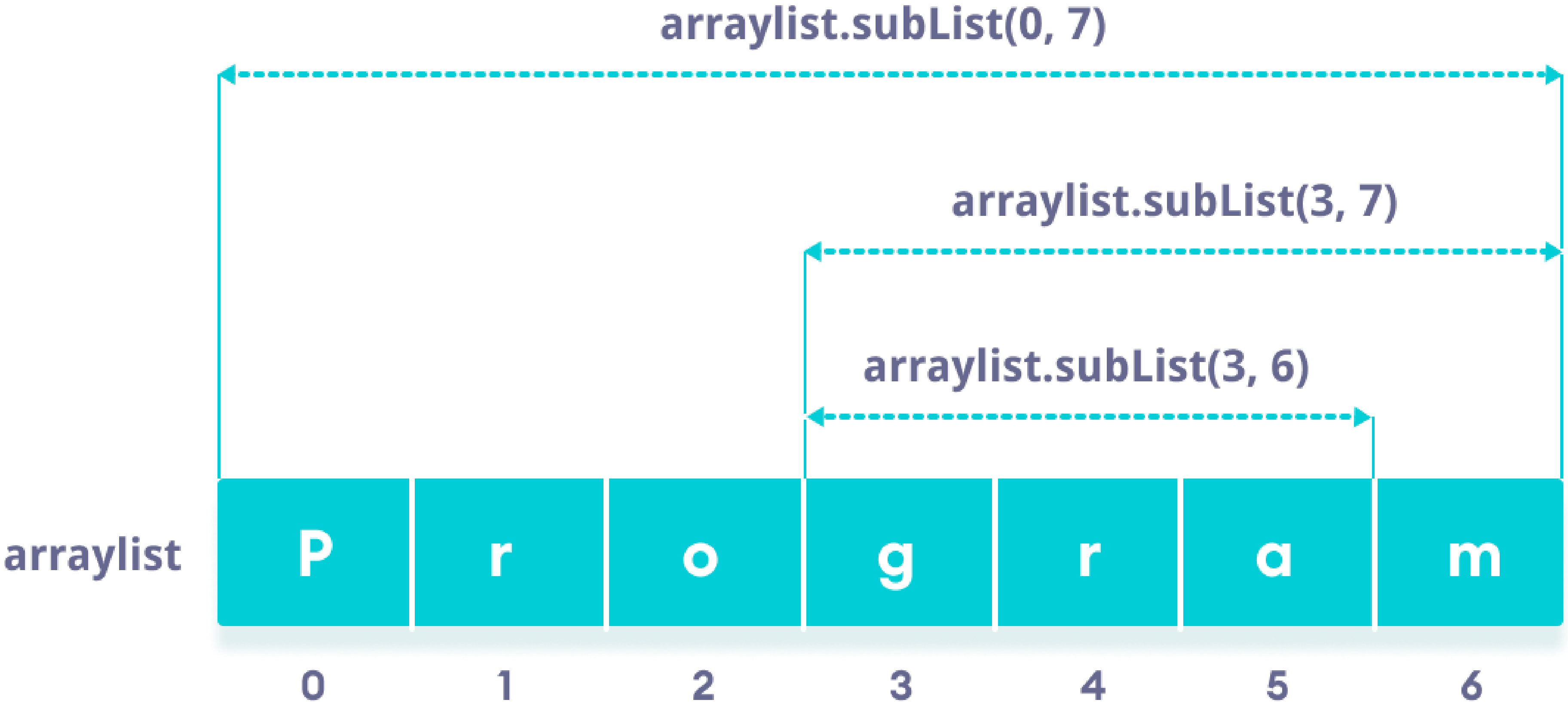


`addAll(`**list**`)`
`addAll(`**index**`,` **list**`)`
	adds all elements from the given list to this list (at the end of the list, or inserts them at the given index)

`contains(`**value**`)`	returns true if given value is found somewhere in this list
`containsAll(`**list**`)`	returns true if this list contains every element from given list
`equals(`**list**`)`	returns true if given other list contains the same elements

`iterator()`
`listIterator()`
	returns an object used to examine the contents of the list (seen later)

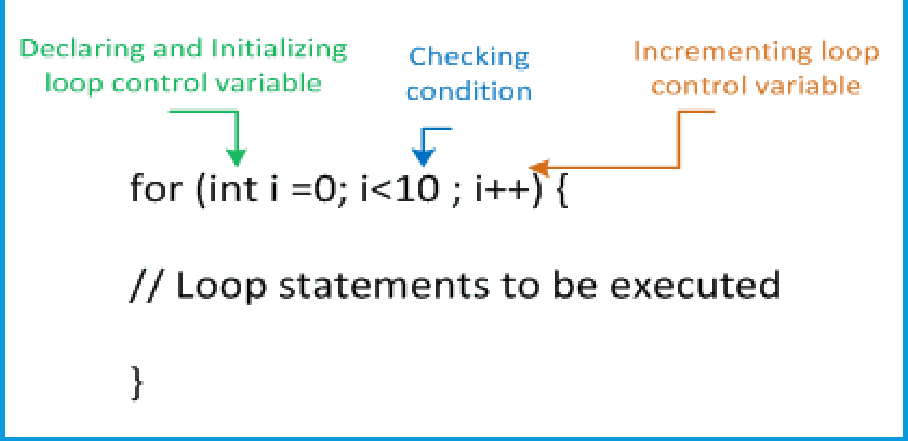
`lastIndexOf(`**value**`)`	returns last index value is found in list (-1 if not found)
`remove(`**value**`)`	finds and removes the given value from this list
`removeAll(`**list**`)`	removes any elements found in the given list from this list
`retainAll(`**list**`)`	removes any elements *not* found in given list from this list
`subList(`**from**`,` **to**`)`	returns the sub-portion of the list between indexes **from** (inclusive) and **to** (exclusive)
`toArray()`	returns the elements in this list as an array



ARRAYLIST SUBLIST

FOR LOOP

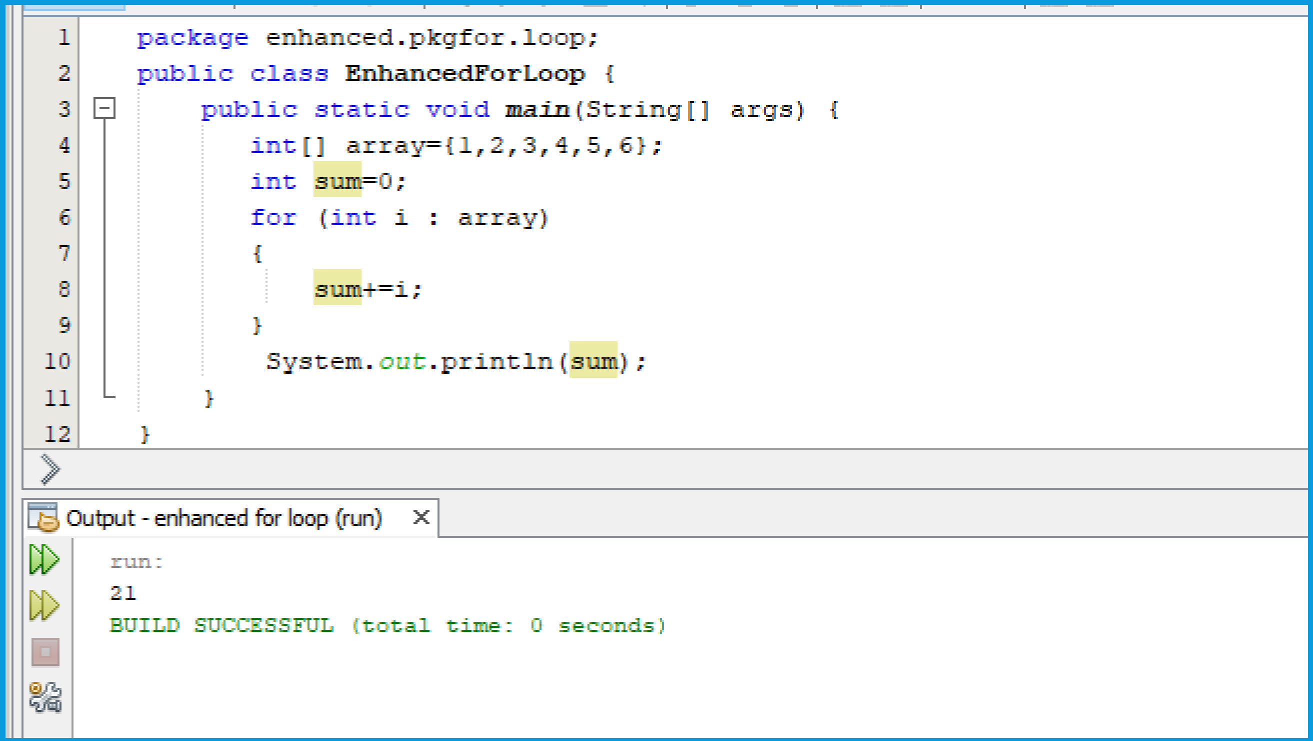
'normal'



Declaring and Initializing loop control variable
Checking condition
Incrementing loop control variable

```
for (int i =0; i<10 ; i++) {

// Loop statements to be executed

}
```

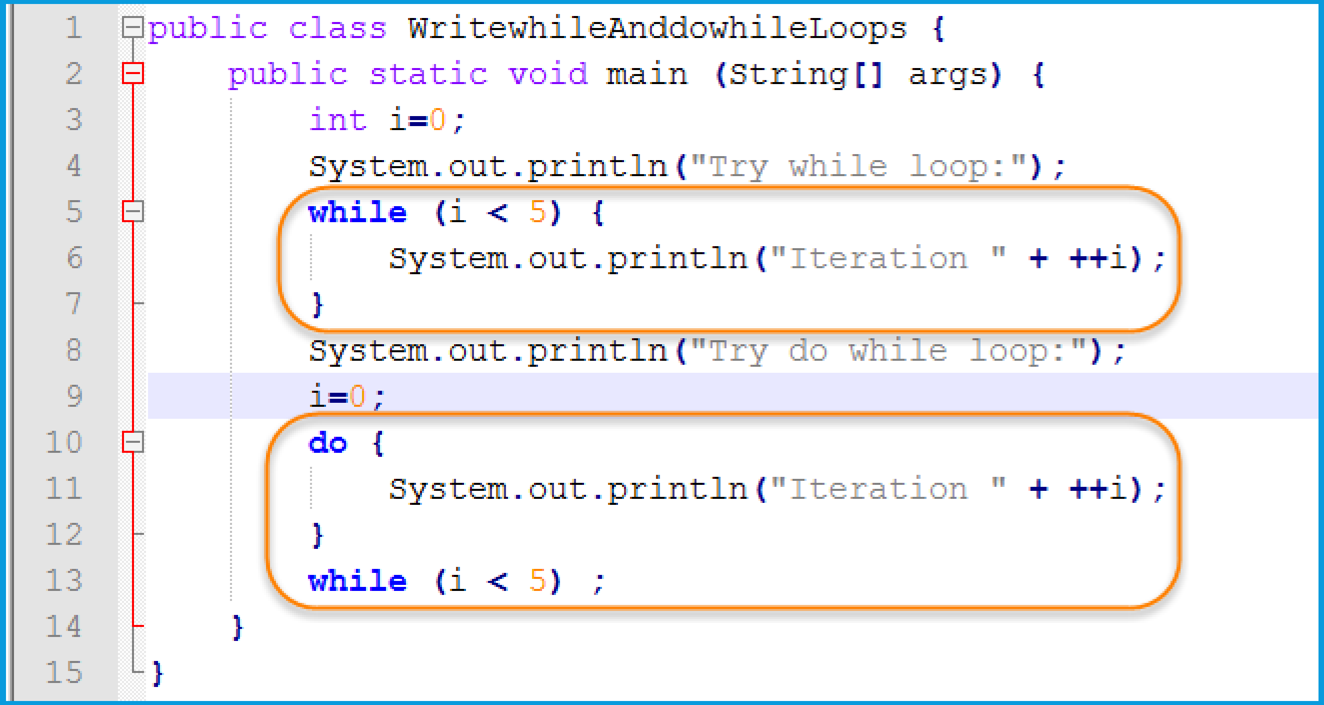
'enhanced'



```
package enhanced.pkgfor.loop;
public class EnhancedForLoop {
    public static void main(String[] args) {
        int[] array={1,2,3,4,5,6};
        int sum=0;
        for (int i : array)
        {
            sum+=i;
        }
         System.out.println(sum);
    }
}
```

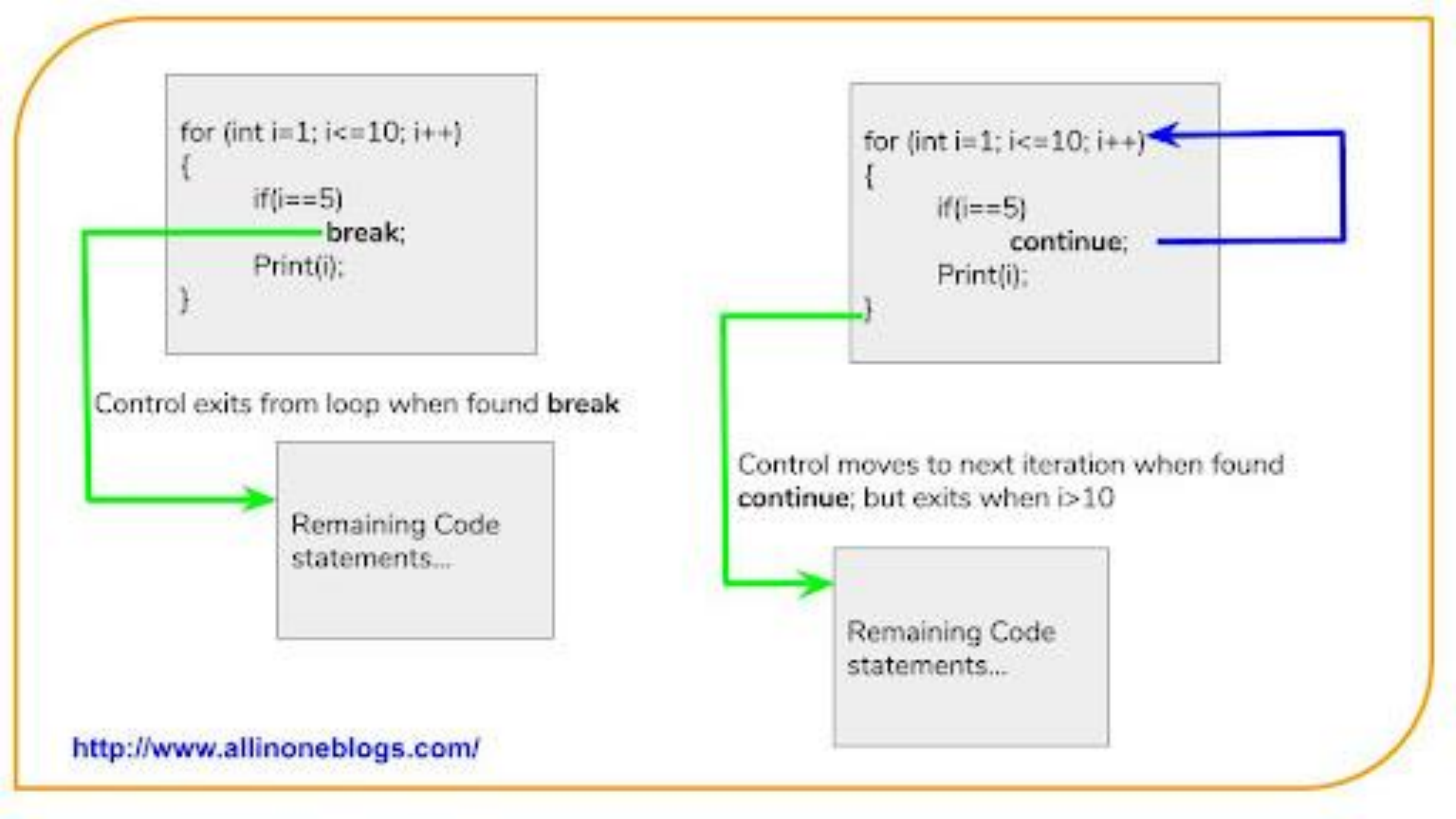
```
run:
21
BUILD SUCCESSFUL (total time: 0 seconds)
```

WHILE LOOP



```
public class WritewhileAnddowhileLoops {
    public static void main (String[] args) {
        int i=0;
        System.out.println("Try while loop:");
        while (i < 5) {
            System.out.println("Iteration " + ++i);
        }
        System.out.println("Try do while loop:");
        i=0;
        do {
            System.out.println("Iteration " + ++i);
        }
        while (i < 5) ;
    }
}
```

BREAK & CONTINUE



```
for (int i=1; i<=10; i++)
{
        if(i==5)
                break;
        Print(i);
}
```

Control exits from loop when found **break**

Remaining Code statements...

```
for (int i=1; i<=10; i++)
{
        if(i==5)
                continue;
        Print(i);
}
```

Control moves to next iteration when found **continue**; but exits when i>10

Remaining Code statements...

http://www.allinoneblogs.com/

WORKSHOP

ArrayList loop (3 ways)

Boter, kaas en eieren

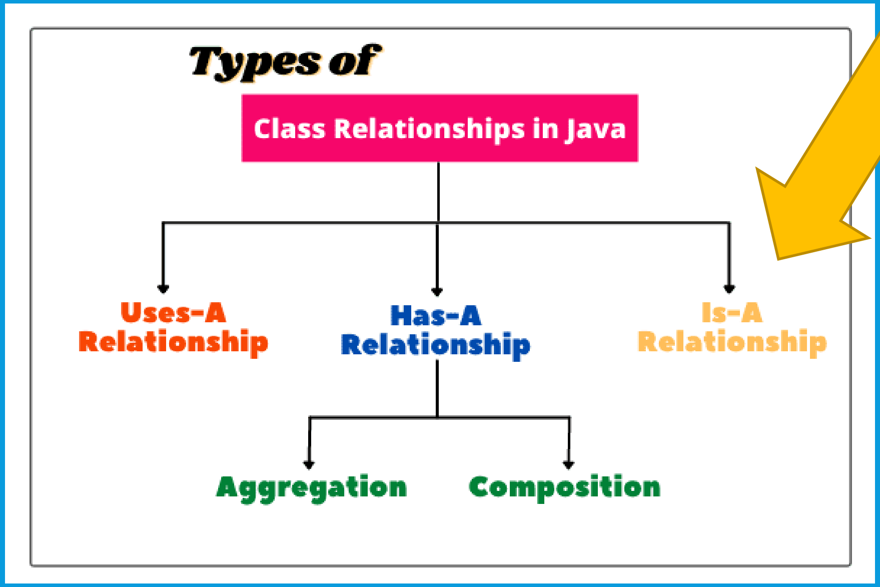
OEFENOPDRACHTEN

https://github.com/hogeschoolnovi/SD-BE-JP-oefenopdrachten/tree/master/src/nl/novi/opdrachten

Lees readme.md voor moeilijkheid. Maak 1 oefening per categorie (dus totaal 4)

- Beslissingsstructuren
- Lijsten
- Methoden
- While-lussen

KLASSE OVERERVING



Types of

Class Relationships in Java

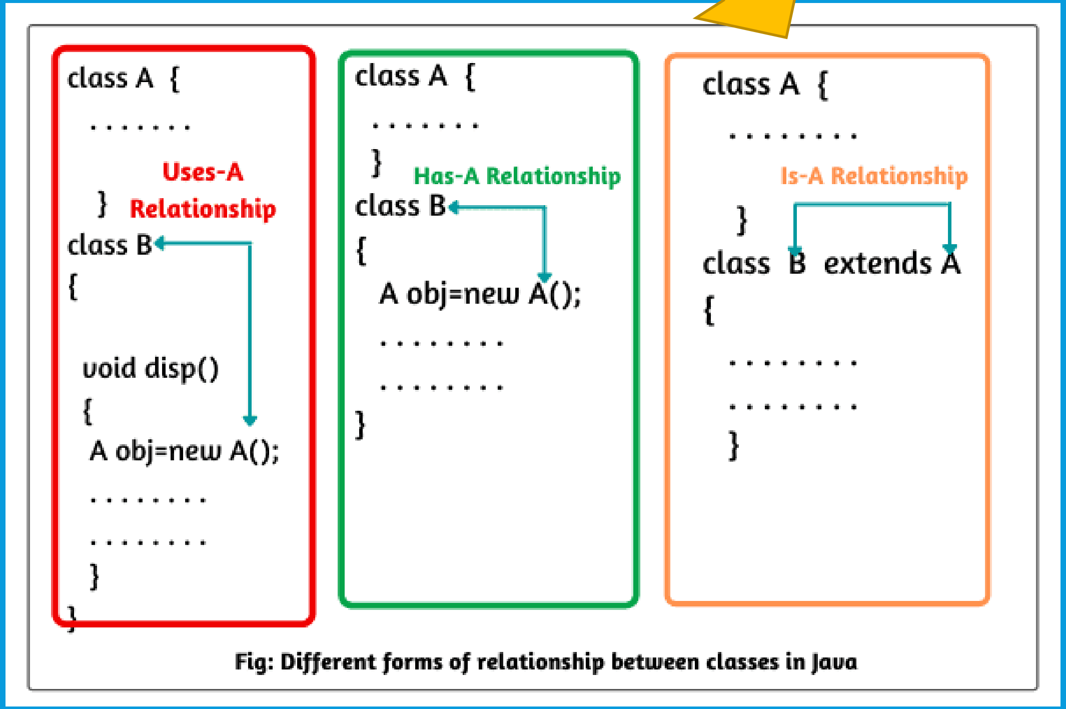
Uses-A Relationship

Has-A Relationship

Is-A Relationship

Aggregation

Composition



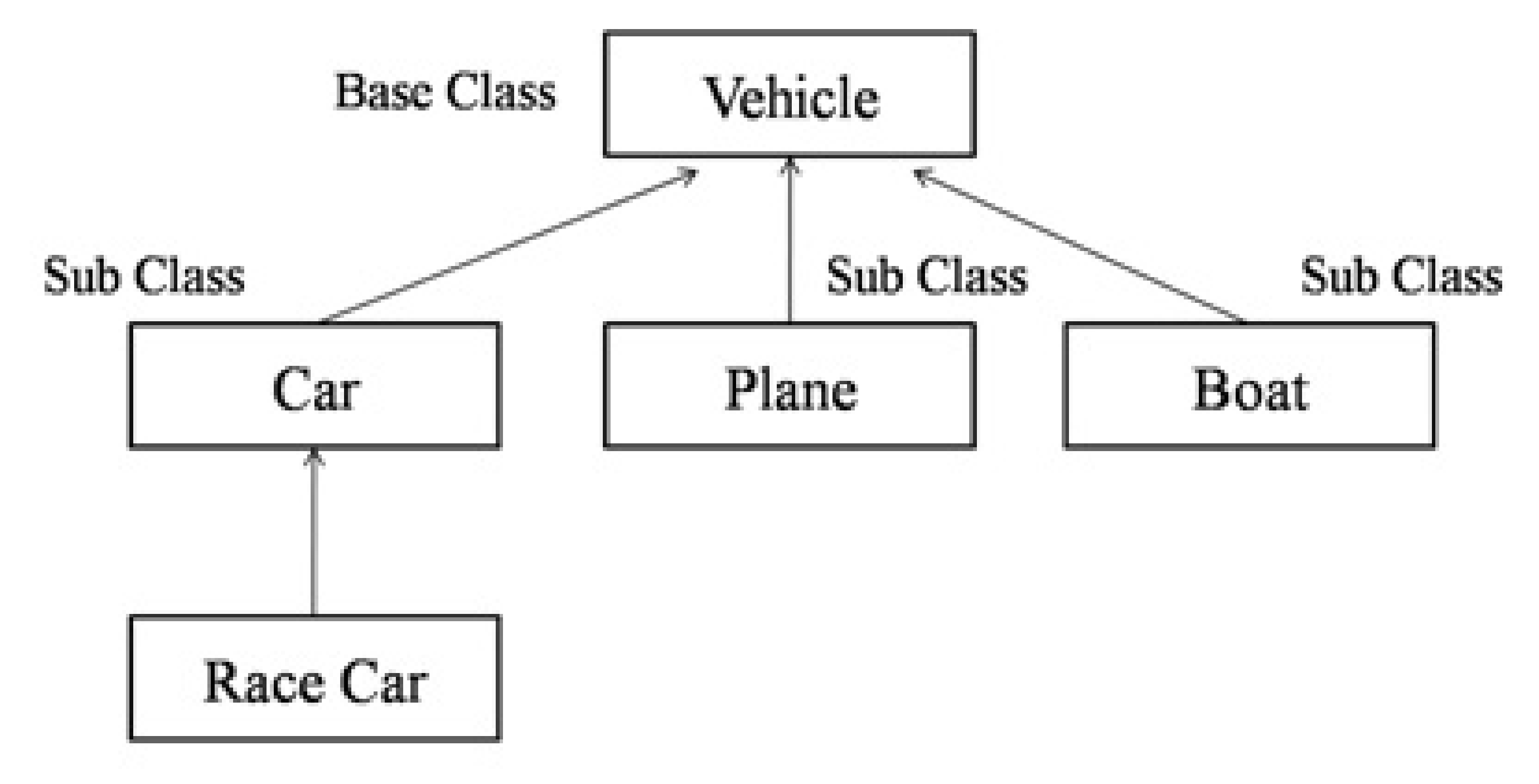
```
class A {
  .......
        Uses-A
  }  Relationship
class B
{

 void disp()
 {
  A obj=new A();
  ........
  ........
 }
}
```

```
class A {
  .......
 }  Has-A Relationship
class B
{
  A obj=new A();
  ........
  ........
}
```

```
class A {
   ........
      Is-A Relationship
  }
class B extends A
{
   ........
   ........
  }
```

Fig: Different forms of relationship between classes in Java

KLASSE OVERERVING



OPDRACHT ‘KLASSE OVERERVING’

- Werk in tweetallen

- Kies een onderwerp voor een klasse structuur

- Uit welke klassen bestaat de structuur? Maak een schets.

- Welke velden horen bij welke klassen?

HUISWERK

- Maak nog (minstens) 4 oefenopgaven uit SD-BE-JP-oefenopdrachten

- Bestudeer Java Programmeren EdHub:
  - Hoofdstuk 3.1 t/m 3.4 : Relaties
  - Hoofdstuk 4.1 t/m 4.7 : Overerven