

Les 9A

De klasse ArrayList

De klasse ArrayList<T>

[java.security.cert](#)
[java.security.interfaces](#)
[java.security.spec](#)
[java.sql](#)
[java.text](#)
[java.util](#)
[java.util.jar](#)
[java.util.logging](#)
[java.util.prefs](#)
[java.util.regex](#)
[java.util.zip](#)
[javax.accessibility](#)
[javax.crypto](#)

Constructor Summary

[ArrayList\(\)](#)

Constructs an empty list with an initial capacity of ten.

[ArrayList\(Collection c\)](#)

Constructs a list containing the elements of the specified collection, in the order they are returned by the collection's iterator.

[ArrayList\(int initialCapacity\)](#)

Constructs an empty list with the specified initial capacity.

Method Summary

void [add](#)(int index, [Object](#) element)

Inserts the specified element at the specified position in this list.

boolean [add](#)([Object](#) o)

Appends the specified element to the end of this list.

boolean [addAll](#)([Collection](#) c)

Appends all of the elements in the specified Collection to the end of this list, in the order that they are returned by the specified Collection's Iterator.

boolean [addAll](#)(int index, [Collection](#) c)

Inserts all of the elements in the specified Collection into this list, starting at the specified position.

void [clear](#)()

Removes all of the elements from this list.

[Object](#) [clone](#)()

Returns a shallow copy of this ArrayList instance.

boolean [contains](#)([Object](#) elem)

Returns true if this list contains the specified element.

void [ensureCapacity](#)(int minCapacity)

Increases the capacity of this ArrayList instance, if necessary, to ensure that it can hold at least the number of elements specified by the minimum capacity argument.

[Object](#) [get](#)(int index)

Returns the element at the specified position in this list.

int [indexOf](#)([Object](#) elem)

Searches for the first occurrence of the given argument, testing for equality using the equals method.

[java.util](#)

Interfaces

[Collection](#)

[Comparator](#)

[Enumeration](#)

[EventListener](#)

[Iterator](#)

[List](#)

[ListIterator](#)

[Map](#)

[Map.Entry](#)

[Observer](#)

[RandomAccess](#)

[Set](#)

[SortedMap](#)

[SortedSet](#)

Classes

[AbstractCollection](#)

[AbstractList](#)

JAVA SE6

[ArrayList](#)

[Arrays](#)

[BitSet](#)

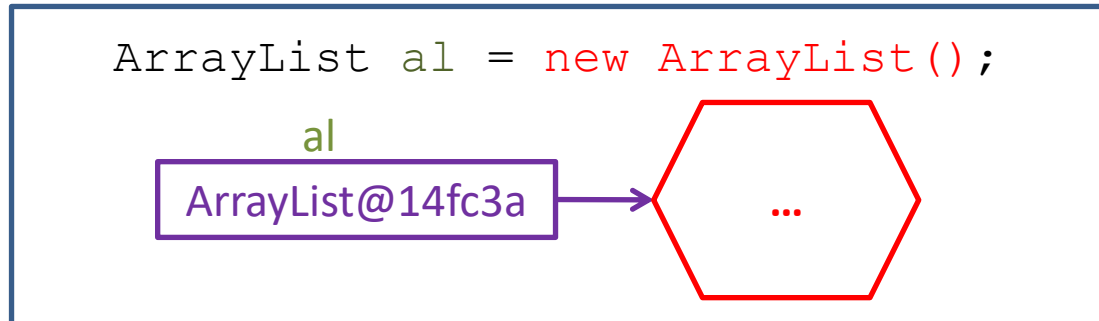
[Calendar](#)

[Collections](#)

[Currency](#)

[Date](#)

Klassieke ArrayList als datatype

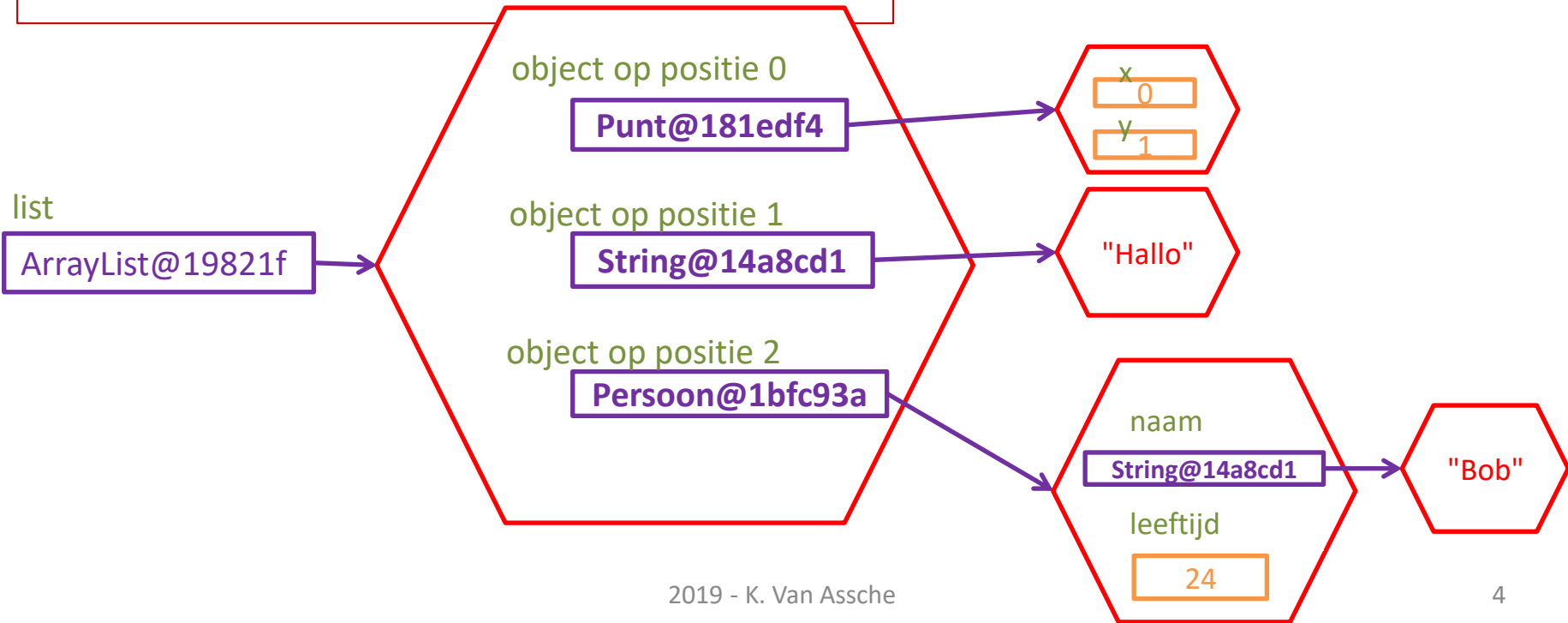


- Elementen in een ArrayList = van **het type Object** !
 - => Elementen kunnen van gelijk **welk objecttype** zijn
 - => ArrayList kan objecten van **verschillende objecttypes** bevatten !
 - => De elementen zijn referenties (naar objecten)
- Aantal elementen in een ArrayList kan **groeien** !

Type van objecten in ArrayList

JAVA SE6

```
ArrayList list = new ArrayList();  
  
list.add(new Punt(0,1));  
list.add(new String("Hallo"));  
list.add(new Persoon("Bob" 24));
```



Polymorfisme

```
public class Console {  
    public static void main(String[] args) {  
        ArrayList lijst = new ArrayList();  
        lijst.add(new Punt(0,1));  
        lijst.add(new String("Hallo"));  
        lijst.add(new Persoon("Bob", 24));  
  
        drukInfo(lijst);  
    }  
  
    public static void drukInfo(ArrayList lijst) {  
        for (Object o : lijst) {  
            System.out.println(o);  
        }  
    }  
}
```

Upcast van Punt naar Object
Upcast van String naar Object
Upcast van Persoon naar Object

Late binding
Downcast naar respectievelijk Punt, String en Persoon
Oproep van de resp. toString() methode

```
(0,1)  
Hallo  
Bob (24 jaar)
```

Al dan niet generisch

```
ArrayList list = new ArrayList();  
  
list.add(new Punt(0,1));  
list.add(new String("Hallo"));  
list.add(new Persoon("Bob" 24);
```


JAVA SE6
(niet-generisch)

```
ArrayList<Punt> list = new ArrayList<Punt>();  
  
list.add(new Punt(0,1));  
list.add(new Punt(1,4));  
list.add(new Punt(7,11));
```

JAVA SE7
(generisch)

java.util

Class **ArrayList**<E>



java.lang.Object

java.util.AbstractCollection<E>

java.util.AbstractList<E>

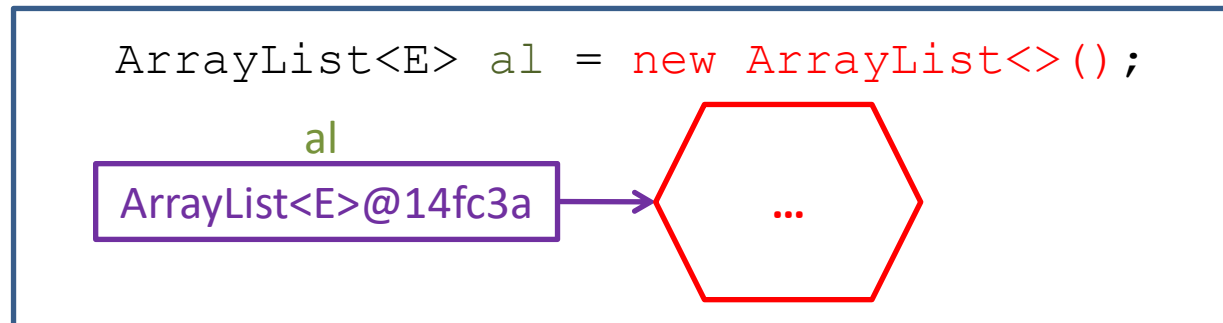
java.util.**ArrayList**<E>



All Implemented Interfaces:

Serializable, Cloneable, Iterable<E>, Collection<E>, List<E>, RandomAccess

ArrayList<E> als datatype



- Alle elementen zijn van **type E**
- Alle elementen zijn referenties (naar objecten)
- Aantal elementen in een ArrayList kan **groeien** !
- Enkele voorbeelden:
 - `ArrayList<int>` Hierbij worden de int-waarden geboxt (i.e. copy-object op heap)
 - `ArrayList<Werknemer>` evt. gevuld met bedienden en arbeiders (specialisaties van klasse `Werknemer`)
 - `ArrayList<Kat>`
 - `ArrayList<Bankfiliaal>`

Nuttige `ArrayList<E>`-methoden

- `public boolean add(E element)`
- `public void add(int index, E element)`
- `public int size()`
- `public E get(int index)`
- `public void clear()`
- `public boolean contains(Object o)`
- `public boolean isEmpty()`
- `public E remove(int index)`
- `public boolean remove(Object o)`

Voorbeeld

Cf. API:
public ArrayList<E>()

```
ArrayList<Punt> coordinaten;
```

coordinaten

null

```
coordinaten = new ArrayList<>();
```

coordinaten

ArrayList<Punt>@19821f

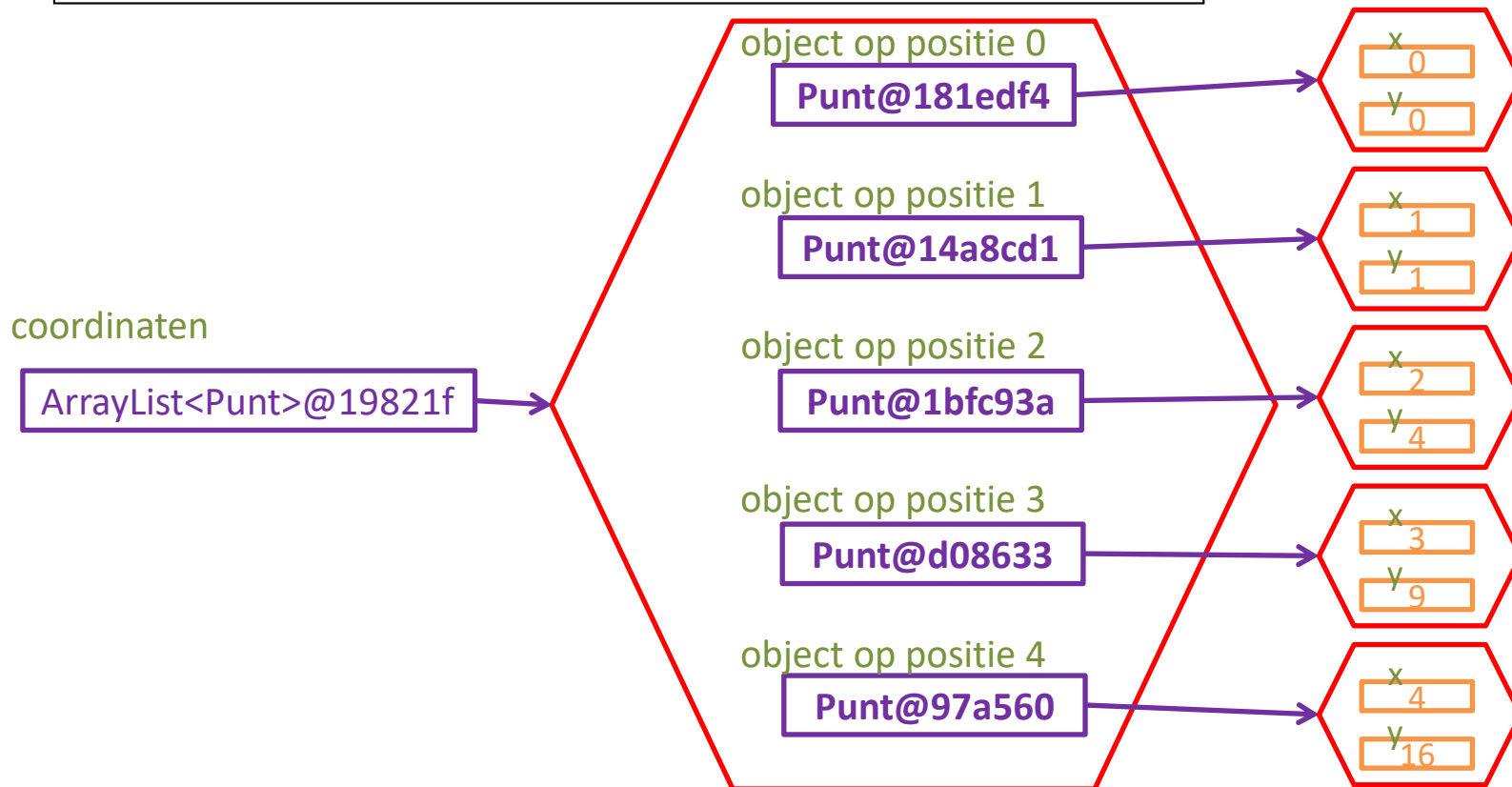


```
ArrayList<Punt> coordinaten = new ArrayList<>();
```

```
int y;
```

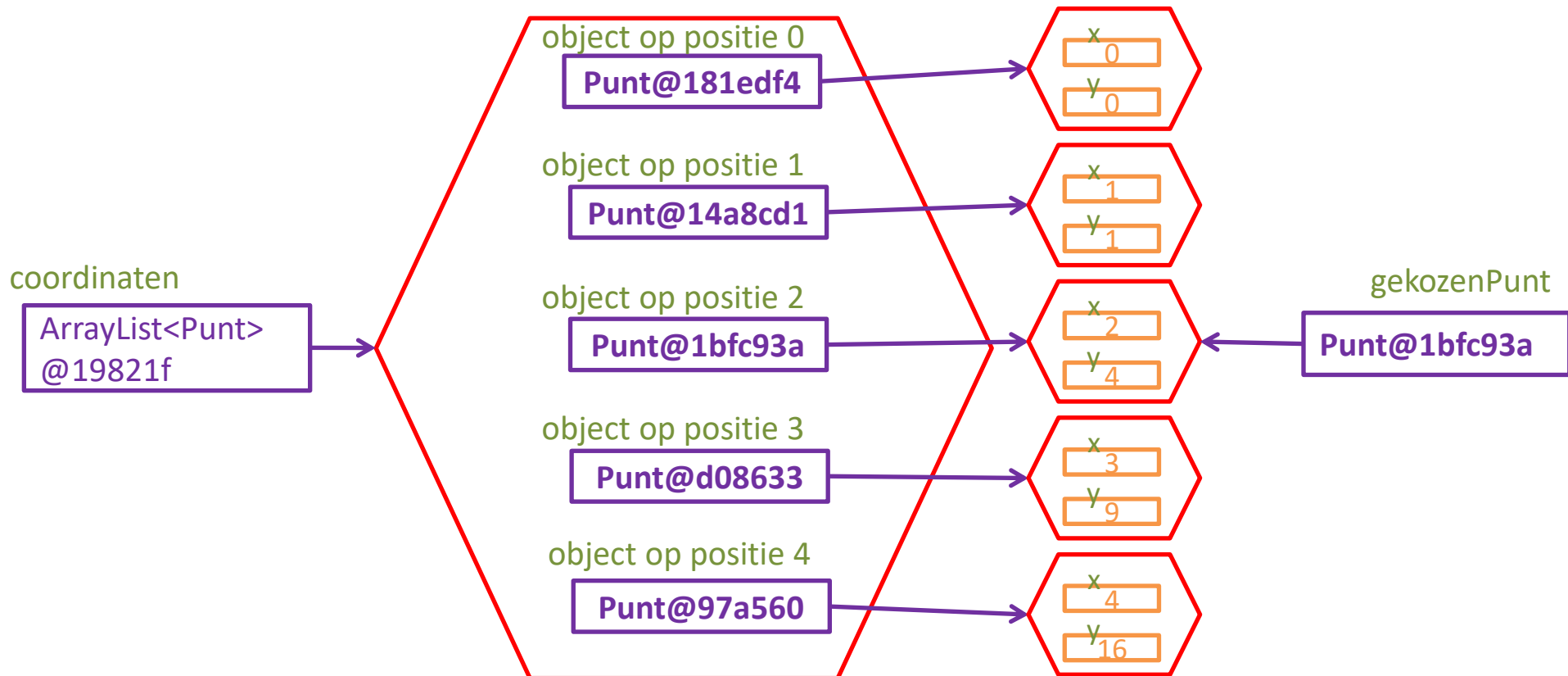
```
for (int x = 0; x < 5; x++) {  
    y = kwadraat(x);  
    coordinaten.add(new Punt(x, y));  
}
```

Cf. API:
public boolean add(E element)



```
Punt gekozenPunt = coordinaten.get(2) ;
```

Cf. API:
public E get(int index)

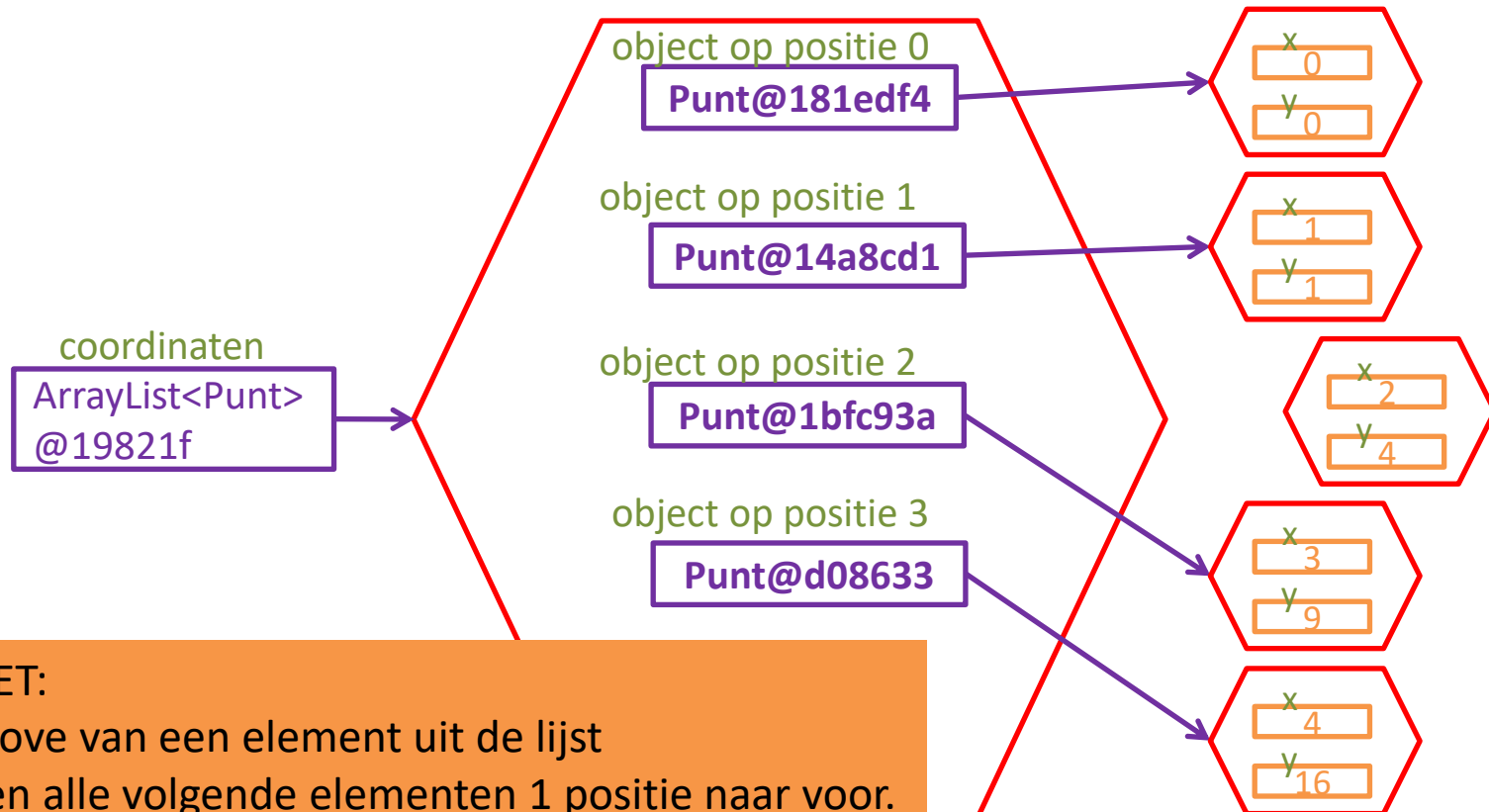


```
System.out.println(coordinaten.size()); //5  
coordinaten.remove(2);  
System.out.println(coordinaten.size()); //4
```

Cf. API:

public int size()

public **E** remove(int index)



OPGELET:

Bij remove van een element uit de lijst
schuiven alle volgende elementen 1 positie naar voor.

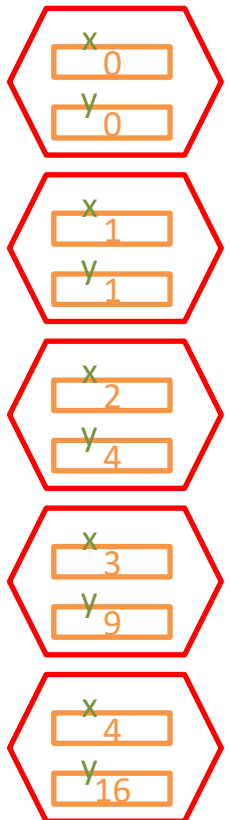
```
if (!coordinaten.isEmpty()) {  
    coordinaten.clear();  
}
```

Cf. API:

public boolean isEmpty()

public void clear()

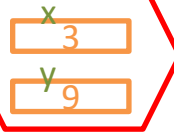
coordinaten
ArrayList<Punt>
@19821f



```
Punt teZoekenPunt = new Punt(3, 9);
```

teZoekenPunt

Punt@4cdf3ab



```
if (!coordinaten.contains(teZoekenPunt)) {  
    ...  
}
```

Cf. API:

public boolean contains(**Object** o)

OPGELET:

Hier gebeurt impliciete oproep van de 'equals' implementatie in klasse Punt !

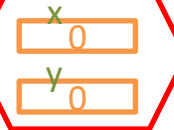
→ **Zorg ervoor dat je een zinvolle override voor de 'equals' methode geschreven hebt !!!!!**

coordinaten

ArrayList<Punt>
@19821f

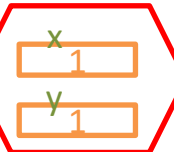
object op positie 0

Punt@181edf4



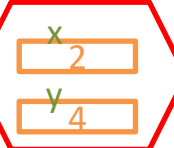
object op positie 1

Punt@14a8cd1



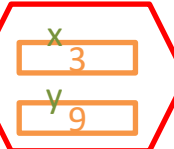
object op positie 2

Punt@1bfc93a



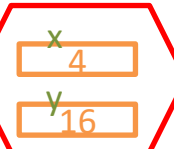
object op positie 3

Punt@d08633



object op positie 4

Punt@97a560



onderliggende werking

- De methode `contains` zal het meegegeven `Punt`-object vergelijken met elk element in de `ArrayList`. Via een *autonome* oproep van de `equals`-methode van de klasse `Punt` wordt uitgezocht of het meegegeven object al dan niet in de `ArrayList` vervat zit.
- Noot: Is er in de klasse `Punt` geen `equals` methode bepaald, dan wordt er teruggevallen op de `equals` implementatie van de root klasse `Object`.

Concreter voorbeeld

```
public void start() {  
    ArrayList<Punt> list = new ArrayList<Punt>();  
  
    for (int i = 0; i < 100; i++) {  
        list.add(new Punt(i, i*i));  
    }  
  
    int getal = (int)(Math.random() * 100);  
    System.out.println("Geef kwadraat van : " + getal);  
    int antwoord = Input.readInt();  
  
    if (list.contains(new Punt(getal, antwoord)) ) {  
        System.out.println("Goed geraden!");  
    }  
    else {  
        System.out.println("Niet correct");  
    }  
}
```

Bevat o.a. punt (5, 25)

Bv. getal = 5

Bv. antwoord = 25

←?!

←?!

Uitvoer zonder equals methode in klasse Punt

```
class Punt {  
    private int x;  
    private int y;  
  
    public Punt(int x, int y) {  
        this.x = x;  
        this.y = y;  
    }  
}
```

**Geef kwadraat van : 60
3600
Niet correct**

Uitvoer mét equals methode in klasse Punt

```
class Punt {  
    private int x;  
    private int y;  
  
    public Punt(int x, int y) {  
        this.x = x;  
        this.y = y;  
    }  
  
    @Override  
    public boolean equals(Object o) {  
        if (!(o instanceof Punt)) {  
            return false;  
        }  
  
        return this.x == ((Punt)o).x  
            && this.y == ((Punt)o).y;  
    }  
}
```

Geef kwadraat van : 60
3600
Goed geraden!

Projects	Ctrl+1
Files	Ctrl+2
Favorites	Ctrl+3
Services	Ctrl+5
Navigator	Ctrl+7
Action Items	Ctrl+6
Tasks	Ctrl+Shift+6
Output	Ctrl+4
Editor	Ctrl+0
Debugging	
Profiling	
Web	
IDE Tools	
Configure Window	
Reset Windows	
Close Window	Ctrl+W
Close All Documents	Ctrl+Shift+W
Close Other Documents	
Document Groups	
Documents...	Shift+F4

Variables	Alt+Shift+1
Watches	Alt+Shift+2
Call Stack	Alt+Shift+3
Loaded Classes	Alt+Shift+4
Breakpoints	Alt+Shift+5
Sessions	Alt+Shift+6
Threads	Alt+Shift+7
Sources	Alt+Shift+8
Debugging	Alt+Shift+9
Analyze Stack	

Call Stack venster

```

17      @Override
18      public boolean equals(Object o) {
19          if (!(o instanceof Punt)) {
20              return false;
21          }
22
23          return this.x == ((Punt) o).x && this.y == ((Punt) o).y;
24      }

```

Name
Punt.equals:19
Hidden Source Calls
ArrayList.indexOf:317
ArrayList.contains:300
Console.start:28
Console.main:13