IK WIL

OCA Java SE 8 Examentraining

vijfhart IT-OPLEIDINGEN

Agenda

1 Over het examen

2 Aandachtspunten

3 Sheets

4 Nul-meting

5 Evaluatie

6 Oefening

Cursusmateriaal

- Boek OCA Java SE 8 Programmer I Study Guide
- Proefexamens

Over het examen

- Examennummer: 1Z0-808
- Titel: Oracle Certified Associate, Java SE 8 Programmer
- Prijs: 212 Euro
- Duur: 150 minute
- Aantal vragen: 77
- Minimale score: 65 % (kan worden bijgesteld)
- Certificaat verloopt niet

Tips en aandachtspunten

- Neem niks mee naar het examen
- Lees de vraag goed door:
  - Gebruik eventueel de scrollbar
  - of de Code Exhibit knop om de hele vraag te lezen
- Bedenk wat het juiste antwoord moet zijn, voor de antwoorden te lezen
- Kies altijd het gevraagde aantal juiste antwoorden
- Markeer vragen die teveel tijd kosten en ga naar de volgende vraag
- Doorloop de gemarkeerde vragen op het eind met de knop Review Marked Questions

Java Basics

- package en imports gelden voor alle klassen binnen het .java bestand
- dus: klassen in verschillende packages staan ook in verschillende .java bestanden
- Klassen uit de default package kunnen niet worden geïmporteerd vanuit een andere package
- import static java.lang.Math.*;
- import static java.lang.Math.round;

Naamgeving: gereserveerde woorden

abstract	continue	for	new	switch
assert	default	goto*	package	synchronized
boolean	do	if	private	this
break	double	implements	protected	throw
byte	else	import	public	throws
case	enum	instanceof	return	transient
catch	extends	int	short	try
char	final	interface	static	void
class	finally	long	strictfp	volatile
const*	float	native	super	while

*** worden niet gebruikt in java, maar zijn wel gereserveerd**

Working with Java data types

→ Samengestelde operatoren hoeven niet expliciet te worden gecast:

```
int x = 1;
x += 2L; // compileert
x = x + 2L; // compileert niet
```

→ Vergelijking van onvergelijkbare objecten of waarden: compileert niet

```
Integer x=1; Long y = 2L;
System.out.println(x==y); // compileert niet
int a =1; long b = 2L;
System.out.println(a==b); // compileert wel
```

→ char kan 0 t/m 65535 bevatten, grotere of kleinere gehele waarden kunnen naar char worden gecast:

```
char x = (char)-3;
```

Primitieve datatypen

Datatype	Grootte (bits)	Waarde	Bereik	Default waarde
byte	8	geheel getal	-128 t/m +127	0
short	16	geheel getal	-32.768 t/m +32.767	0
int	32	geheel getal	-2.147.483.648 t/m +2.147.483.647	0
long	64	geheel getal	-(2^63) t/m +(2^63-1)	0L
float	32	reëel getal	1.4E-45 t/m +3.4E38 (+ en -)	0.0F
double	64	reëel getal	4.9E-234 t/m +1.797E308 (+ en -)	0.0D
char	16	een Unicode karakter	0 t/m 65.535 (van '\u0000' t/m '\uffff')	\u0000 (null)
boolean	1	true of false	true of false	false

Automatische type promotie

Het resultaat van een expressie met operanden van type t kan buiten de grenzen van type t vallen:

bijvoorbeeld

```
byte a= 100, b = 100;
int   c = a * b;  // 10.000 past niet in byte
```

Automatische type promotion in expressies:

- → als een van beide getallen een double is, wordt het andere getal naar double gecast
- → anders, als een van beide getallen een float is, wordt het andere getal een float
- → anders, als een van beide getallen een long is, wordt het andere getal een long
- → anders worden beide getallen een int

`byte d = a+b;` // foutmelding bij compileren:

```
possible loss of precission
required: byte
found: int
```

Using Operators and Decision Constructs

- Use Java operators; including parentheses to override operator precedence
- Test equality between Strings and other objects using == and equals ()
- Create if and if/else and ternary constructs
- Use a switch statement

- switch accepteert ook expressies die een int opleveren
- switch(x++): vergelijk if(x++ == ...)
- case waarden moeten compile time constanten zijn
- case waarden moeten toe te kennen zijn aan het argument van switch
- switch bevat minimaal een blok { } zonder case of default
- enhanced for loop is alleen te gebruiken om elementen te doorlopen, niet om ze te wijzigen. Uitzondering: de state van objecten in de verzameling zijn mogelijk te wijzigen via methoden van die objecten.
- alle statements behalve declaraties kunnen gelabeld worden
- declaratie tussen { en } kan wel gelabeld worden
- do{...}while(...)**;**
- if(false){} compileert, while(false){} of for(;false;){} niet

Prefix en postfix bewerkingen

- Bij een prefix bewerking wordt eerst de bewerking uitgevoerd en daarna de expressie geëvalueerd.
- Bij een postfix bewerking wordt eerst de expressie geëvalueerd en daarna de bewerking uitgevoerd.

```
int x=1, y=1;
System.out.println(x++ +" "+ ++y + " " + x-- + " " + --y );
// uitvoer: 1 2 2 1
// waarde van x is hierna 1

x = x++;
System.out.println(x);

//uitvoer: 1
```

Prioriteit van operatoren

volgorde	operator	beschrijving	richting
1	**[ ] ( ) .**	array index, methode aanroep, member toegang	links naar rechts
2	**++ --**	postfix ophoging, postfix verlaging	rechts naar links
3	**++ --** **+ -** **! ~**	prefix ophoging en verlaging, positief, negatief logische NOT, bitwise NOT	rechts naar links
4	**(type) new**	type cast, object aanmaken	rechts naar links
5	*** / %**	vermenigvuldiging, deling, modulus	links naar rechts
6	**+ - +**	optellen, aftrekken, concatenatie	links naar rechts
7	**<< >> >>>**	bitwise shift	links naar rechts
8	**< <=** **> >=** **instanceof**	kleiner dan, kleiner dan of gelijk aan groter dan, groter dan of gelijk aan reference test	links naar rechts
9	**== !=**	gelijkheid, ongelijkheid (waarde of reference)	links naar rechts
10	**&**	bitwise AND / Boolean AND	links naar rechts
11	**^**	bitwise XOR / Boolean XOR	links naar rechts
12	**|**	bitwise OR / Boolean OR	links naar rechts
13	**&&**	conditional (optimized) AND	links naar rechts
14	**||**	conditional (optimized) OR	links naar rechts
15	**? :**	conditional	rechts naar links
16	**= += -= *=** **/= %= &=** **^= |= <<=** **>>= >>>=**	toekenning en samengestelde toekenningen	rechts naar links

De String klasse

- String is een referentie type
- Waarde kan niet worden gewijzigd
- Wijziging van String variabele = aanmaken nieuw String object
- Kan worden geïnitialiseerd alsof het een primitief type is
- Wordt dan mogelijk uit de "String Constant Pool" gehaald

```
String tekst1 = new String("tekst"); // nieuw object

String tekst2 = "tekst"; // mogelijk nieuw object
String tekst3 = "tekst"; // anders uit de String Constant Pool

System.out.println(tekst1==tekst2); // false
System.out.println(tekst2==tekst3); // true
System.out.println(tekst1.equals(tekst2)); // true
```

Methods and encapsulation

- let op de scope van lokale variabelen
- objecten kunnen in aanmerking komen voor garbage collection:
  - als ze null krijgen toegewezen
  - aan het einde van hun scope
  - als er geen referentie bestaat, zoals: new String();
- overloaded constructors mogen verschillende toegankelijkheid hebben
- encapsulation en information hiding worden in het examen door elkaar gebruikt
- code die alleen na een return statement kan worden uitgevoerd: compileert niet
- **Instance methods bind at runtime:** reference type bepaalt “wat” de variabele kan, object type bepaalt “hoe”.
- **All variables and static methods bind at compile time:** reference type bepaalt hoe en wat.

Modifiers voor toegangscontrole

	public	protected	Default (zonder modifier)	private
Zelfde klasse	ja	ja	ja	ja
Subklasse in zelfde package	ja	ja	ja	nee
Niet subklasse in zelfde package	ja	ja	ja	nee
Subklasse in andere package	ja	ja, *via overerving*	nee	nee
Niet subklasse in andere package	ja	nee	nee	nee

Een public *class* kan ook benaderd worden vanuit andere packages, zonder public is een klasse alleen binnen de eigen package bereikbaar.

static members

- static "members" (methoden of attributen) zijn members van een klasse, niet van een instantie
- direct te gebruiken vanuit de klasse (geen object nodig)
- geen overerving: static members horen alleen bij de klasse waarin ze zijn gedefinieerd, niet bij subklassen
- subklassen kunnen wel bij de static members van de superklasse, maar krijgen daar geen eigen versie van
- static methoden kunnen niet abstract zijn
- static methoden kunnen alleen bij andere static members

final

- final klasse:
  - hiervan kan geen subklasse worden gemaakt
  - een final klasse kan dus ook niet abstract zijn
- final variabelen:
  - waarde kan na initialisatie niet worden gewijzigd
  - final reference variabele: kan niet naar ander object verwijzen, object zelf kan wel wijzigen
  - final instance variabele: waarde moet direct bij declaratie, in een "instance initializer", of in de constructor worden geïnitialiseerd
  - final static variabele: waarde moet direct bij declaratie, of in een "static initializer" worden geïnitialiseerd (initializers worden in het volgende hoofdstuk behandeld)
- final methoden:
  - kunnen niet worden overschreven in een subklasse
- final parameters van methoden:
  - compiler controleert of parameterwaarden niet worden gewijzigd in de methode

String, StringBuilder

- Een String object wordt intern als final char-array opgeslagen (immutable)
- s.substring(1,3):
- van 2e positie (inclusief) tot 4e positie (exclusief)
- s.indexOf(char c) of s.indexOf(String s):
  -1 indien niet gevonden
- pas op voor += in combinatie met Strings en getallen:
  - evalueert van links naar rechts,
  - zodra een String waarde komt, wordt het voorgaande en volgende naar String omgezet
- pas op voor "void-aanroepen", zoals s.toLowerCase() ipv s=s.toLowerCase();
- StringBuilder heeft default een capaciteit van 16 karacters
- sb.append(Object o): voegt o.toString() toe
- sb.substring() geeft een String object, geen StringBuilder
- StringBuilder heeft geen trim(), String wel

Arrays en ArrayList

- dit kan: int [] [] x; of int x[][]; of int [ ] x [ ];
- x[-3]: compileert, maar geeft runtime ArrayIndexOutOfBoundsException
- een array is een Object met:
  - attribuut length
  - overschreven clone() methode (zonder exception)
- clone() van array en ArrayList met object referenties: referenties worden gecloned, objecten zelf niet:

```
String x[] = { new String () }, y[]=x.clone();
System.out.println(x==y); // uitvoer: false
System.out.println(x[0]==y[0]); // uitvoer: true
```

Arrays initialiseren

```
int[] nummers;
nummers = new int[10]; // lengte opgeven
of:
int[] nummers = new int[10];
of
int[] nummers = new int[] { 1, 3, 5, 7, 9 ,11, 13, 15, 17, 19 };
of
int[] nummers =  { 21, 23 , 25, 27, 29 , 31, 33, 35, 37, 39 };

// op deze manier vullen mag alleen tijdens het initialiseren,
// dus dit mag niet:

int[] nummers = new int[10];
nummers = { 1, 3, 5, 7, 9 ,11, 13, 15, 17, 19 }; // werkt niet!
nummers = new int[]{ 1, 3, 5, 7, 9 ,11, 13, 15, 17, 19 };//werkt wel!
```

Multidimensionale arrays initialiseren

- lengtes van sub-arrays hoeven niet gelijk te zijn
- lengtes van sub-arrays hoeven niet bekend te zijn

Voorbeelden:

```
int[][][] a = new int[4][][]; // correct
int[][][] b = new int[][4][]; // incorrect
int[][][] c = {{{1},{2,3},{4,5,6}},{{7,8}}};
```

Varargs - overwegingen

- een vararg argument is vrijwel gelijkwaardig aan een array argument

Dit mag ook:

```
public static void main (String ... args){}
```

Maar:

- een varargs argument kan alleen als laatste argument aan een methode worden meegegeven
- er mag maar één varargs argument aan een methoden worden meegegeven
- een methode met array argument kan niet worden overschreven door een methode met vararg argument en andersom
- een methode met array argument kan alleen worden aangeroepen met een array, niet met losse element-waarden

Working with inheritance

- constructor kan niet worden overschreven
- "base class" is "super class" is "parent class“
- "derived class" is "subclass" is "extended class" is "child class“
- super en this zijn niet beschikbaar in static context
- een concrete superklasse kan een abstracte subklasse hebben
- de abstracte subklasse kan een methode overschrijven (@Override) met een abstracte methode
- variables bind at compile time, methods bind at runtime
- pas op voor overloaded methoden in de hierarchie: die doen niet mee in polymorfisme

Exception handling

- overridden methoden mogen zelfde checked exceptions als methode van de superklasse opwerpen, of **subklasse** daarvan (overweging: SuperKlasse a = new SubKlasse(); a.methode();)
- daarnaast mogen ze RuntimeExceptions en Errors opwerpen (worden niet gecontroleerd door compiler)
- constructors van een subklasse moeten zelfde checked exceptions opwerpen als die van superklasse, of **superklasse** daarvan (overweging: eerste aanroep van super() kan niet worden afgevangen in try-catch-blok)
- Een static initializer:
  - mag geen checked exceptions opgooien.
  - Unchecked exceptions mogen wel worden opgegooid, maar kunnen niet worden afgevangen.
  - Een runtime exception in een static block veroorzaakt een ExceptionInInitializerError
- Het finally block kan geen primitieve waarde van een variabele wijzigen voordat deze in catch of try wordt geretourneerd. De state van een te retourneren reference variabele kan wel worden gewijzigd.

return in finally

```
public static void main (String args[]) {
    System.out.println(sb());
 }

 static int sb(){
   int x=0;
   try{
                return x++;
              }
              finally{
                 ++x;
              }
 }

// uitvoer: 0 (met return in finally: 2)
```

Exceptions en overriding

```
class A {
  void info() throws FileNotFoundException {
    System.out.println("in A");
  }
}

class B extends A {
  @Override
  public void info() throws ArithmeticException {
    System.out.println("in B");
  }
}

try {
  A obj = new B();
  a.info(); // in B
catch(IOException ioe){}
```

Working with Selected classes from the Java API

- Manipulate data using the StringBuilder class and its methods
- Creating and manipulating Strings
- Create and manipulate calendar data using classes from java.time.LocalDateTime, java.time.LocalDate, java.time.LocalTime, java.time.format.DateTimeFormatter, java.time.Period
- Declare and use an ArrayList of a given type
- Write a simple Lambda expression that consumes a Lambda Predicate expression

Time en Date - now(), of(), parse()

```
LocalDate vandaag = LocalDate.now();
LocalTime nu = LocalTime.now();
LocalDateTime nuVandaag = LocalDateTime.now();

LocalDate datum = LocalDate.of(2015,9,25) // 1-based
LocalDate dagLater = LocalDate.of(2015,Month.SEPTEMBER, 26);
LocalTime lunch = LocalTime.of(12,0);

LocalDate datumUitTekst = LocalDate.parse("2015-09-27");
LocalTime theeTijd = LocalTime.parse("14:30:00");
```

DateTimeFormatter - format()

```
DateTimeFormatter dtf1 = DateTimeFormatter
  .ofPattern("dd MMMM yyyy HH:mm");
DateTimeFormatter dtf2 = DateTimeFormatter
  .ofLocalizedDate(FormatStyle.SHORT); // MEDIUM / LONG / FULL

System.out.println(nu.format(dtf1));
System.out.println(nu.format(dtf2));
```

Periodes

```
class Periods {
   public static void main(String args[]){
     Period tweeWekelijks = Period.ofWeeks(2);
     Period vierWekelijks = Period.ofWeeks(4);
     kalender(LocalDate.of(2016,1,12), tweeWekelijks, "restafval");
     kalender(LocalDate.of(2016,1,5),  tweeWekelijks, "GFT");
     kalender(LocalDate.of(2016,1,28), vierWekelijks, "papier");
   }
   public static void kalender (LocalDate start,
                                Period periode,
                                String rolemmer){
       System.out.println(rolemmer);
       LocalDate dag = start;
       while(dag.isBefore(LocalDate.of(2017,1,1))){
        System.out.println(dag);
        dag = dag.plus(periode);
       }
   }
}
```

Lambda's

- Vullen enige method van een functional interface in
- java.util.function.Predicate<T>: boolean test(T t);

```
 public static void main (String args[]){
     System.out.println(filter(args, s -> s.length()>3));
     System.out.println(filter(args, s -> s.indexOf("i")>=0));
   }
   private static List<String> filter(String[] woorden,
Predicate<String> p){
      List<String> lijst = new ArrayList<>();
      for(String s: woorden){
        if(p.test(s)){
          lijst.add(s);
        }
      }
      return lijst;
   }
```

Eerste proefexamen

- Houd tijd in de gaten
- Markeer lastige vragen
- Neem na afloop alle foutief beantwoorde en de gemarkeerde vragen door (uitleg bestuderen)
- Zijn er categorieën die er in positieve of negatieve zin uitspringen? Die hoofdstukken nog eens goed bestuderen.

- sandra