**Vragen H11**

**1. Welke stelling(en) zijn waar?**

1. Overerving word gebruikt voor het hergebruiken van code
2. Overerving word **niet** gebruikt voor het hergebruiken van code
3. Overerving word gebruikt voor uitbreiding van nieuwe klassen op basis van bestaande klassen
4. Overerving word **niet** gebruikt voor uitbreiding van nieuwe klassen op basis van bestaande klassen
5. Stelling 1 en 4 zijn waar
6. Stelling 2 en 3 zijn waar
7. Stelling 1 en 3 zijn waar 🡪 goede antwoord
8. Stelling 2 en 4 zijn waar

**2. Wat word er bedoel met methoden overladen?**

A. Een methoden in een klasse overschrijven

B. Een methoden in een klasse gebruiken

C. Een methoden met dezelfde methoden naam hergebruiken in een klasse. 🡪Goede antwoord

**3. welke stelling(en) zijn waar?**

Bij overladen word er een nieuwe methoden aangemaakt met dezelfde naam, bij het overschrijven word er een methode ge-erfd en wijzigen als het nodig is.

1 bij Overladen maak je een method die al bestaat met de zelfde naam zodat je extra parameters kan meegeven.

2. een final methode kun je overschrijven.

1. Bij de stellingen zijn waar
2. Alleen stelling 1 is waar 🡪 Goede antwoord
3. Beide stellingen zijn fout
4. Alleen stelling 2 is waar

**4. Wat doet het sleutel word “extends”?**

A. Op deze manier include je een klasse

B. Hiermee overerf je alle eigenschappen en gedrag van de gegeven klasse. 🡪 Goede antwoord

C. Dit moet bij elke klasse voor Java.

D. Hierdoor kun je met GUI’s werken.

1. **Leg het verschil uit tussen “this” en “super”.**

“this” word gebruikt voor het aanroepen van de parameter in een methoden, “super” word gebruikt voor het aanroepen uit de superklasse.

**Vragen H12**

1. Final static int x = 3;

Static int y = 4;

Final static int z = 5;

**Welke van de volgende stukjes code kan niet gecompileerd worden? (Het bovenste stukje code hoort bij deze vraag.) (GEMAAKT DOOR JELLE)**

* 1. y += z;
  2. y += x + z;
  3. x = 3; 🡨 Goede antwoord
  4. y = x + z + 2\*x;

1. **java.lang.Math is?**  
     
   a. een abstracte klasse  
   b. een interface   
   c. een normale klasse  
   d. een final klasse 🡨 Goede antwoord
2. **.**

**Hoofd klasse**

package toetsVraag3;

public class Vraag3 {

public static void main(String[] args) {

SuperKlasse sk = new SuperKlasse();

sk.increment(5);

System.*out*.println(sk.krijgX());

SubKlasse s = new SubKlasse();

s.increment(5);

System.*out*.println(s.krijgX());

}

}

**superklasse**

package toetsVraag3;

public class SuperKlasse {

int x = 0;

public final void increment(int incr) {

x+=incr;

}

public int krijgX() {

return this.x;

}

}

**subklasse**

package toetsVraag3;

public class SubKlasse extends SuperKlasse{

@override

public final void increment(int incr) {

super.x \*= incr;

}

}

**Gegeven zijn de bovenstaande 3 klassen wat is hierbij de output?**

1. 5 en een exception omdat je een final methode probeert de overschrijven 🡨 Goede antwoord
2. 25
3. Een foutcode.
4. 5
5. **Een final variabele is?**
   1. Een variabele waar je de waarde niet van kunt wijzigen dus een constant. 🡨 Goede antwoord
   2. Een primitieve type zoals een int of een float.
6. **Welke stelling(en) zijn waar?**

1 een final klasse kun je overerven.

2 een final variable kun je niet overschrijven.

1. Alleen 1 is juist
2. Alleen 2 is juist
3. Beide stellingen zijn fout
4. Beide stellingen zijn goed

**Vragen H13**

1. **Welke stelling(en) zijn waar?**

1 Je kunt een abstracte klasse maken en gebruiken

2 Je kunt een abstracte methode maken en gebruiken

1. 1 is juist
2. 2 is juist
3. Allebei zijn juist
4. Geen zijn juist 🡨 goede antwoord | je kan abstracte klasses en methodes maken maar niet gebruiken, alleen overschrijven.
5. **Wat is hier waar?**
6. Je kunt een object maken van een abstracte klasse
7. Abstracte methoden kunnen een body hebben
8. Een klasse kan abstract zijn zelfs zonder abstracte methoden 🡨 goede antwoord
9. Als een klasse meer dan 1 abstracte methode bevat moet de klasse zelf ook abstract zijn.
10. **Lees het stukje code en beantwoord de vraag.**

public class MijnAbstract(){

MijnAbstract(){

System.out.print(“Kaas”)

}

Abstract void printDit(char w)

}

**Moet dit een abstracte klasse zijn of niet?**

1. Ja, dit moet een abstracte klasse zijn want er staat een abstracte methode in. 🡨 goed antwoord
2. Ja, dit moet een abstracte klasse zijn want er staat een abstracte constructor in.
3. Nee, dit moet geen abstracte klasse zijn want er staat maar 1 abstracte methode in.
4. Nee, dit moet geen abstracte klasse zijn want de constructor is niet abstract.
5. **Waarvoor kan abstract handig zijn? (Meerdere antwoorden kunnen goed zijn)**
6. Abstracte klasse maar alleen apart van de main klasse. 🡨 goed antwoord
7. Abstract kan goed gebruikt worden met final variabelen.
8. Het neerzetten van een vaste variabele. 🡨 goed antwoord
9. Abstract kan goed gebruikt worden met static variabelen.
10. Abstract kan goed gebruikt worden met superklassen. 🡨 goed antwoord
11. **Wat wordt er geprint bij het volgende stukje code?**

Import java.util.Calendar;

public class time(){

public static void main(String[] args){

Calendar c = new Calendar.getInstance();

System.out.print(c.getTime());

}

}

**Ga er van uit dat het vandaag vrijdag is op 14 juni 2019 en het is 10 voor half 3 en 52 seconden.**

1. Fri 14-6-2019 14:20:52
2. Fri 6-14-2019 14:20:52 CEST
3. Fri Jun 14 2019 14:20:52
4. Fri Jun 14 14:20:52 CEST 2019 🡨 goede antwoord.

**Toetsvragen H14**

1. **Welke stelling(en) zijn waar?**
2. Een interface is een abstracte klasse
3. In een interface declareer je alleen methods met een lege body.
4. Beide stellingen zijn waar 🡪 goede antwoord
5. Beide stellingen zijn fout
6. Alleen stelling 1 is waar
7. Alleen stelling 2 is waar
8. **Welke stelling is fout?**
9. Je hoeft niet alle methods te overschrijven als je een interface implementeert. 🡪 goede antwoord
10. Je moet alle methods overschrijven als je een interface implementeert.
11. Je kan meerde interfaces implementeren.
12. **Welke stelling is waar**
13. Je kan maar per klasse een andere klasse overerven via extends. 🡪 goede antwoord
14. Je kan per klasse maar een interface implementeren