HoGent
BEDRIJF
EN
ORGANISATIE

H6-8: Methoden

OEFENINGEN

Overzicht oefeningen

1. Gebruik van static methoden uit de Math klasse
2. Afleiden van de definitie van een methode a.d.h.v. de aanroep van deze methode
3. Een klasse met 3 methodes (1 static, 2 gewone) volledig implementeren
4. Oefening met 2 klassen, meerdere methodes, overloading, Stringrepresentatie van object, Exception gooien
5. Uitwerken van static methodes
6. Afleiden van de definitie van methodes a.d.h.v. de aanroep van deze methodes

HoGent 2

Overzicht oefeningen

7. Rode draad Rekening: parameter van methode is een object, 1 dim array
8. Kop of munt: volledige applicatie, objecten als parameters, 1 dim array
9. 1 dim array als parameter in een methode
10. 1 dim array als parameter in een methode
11. Verwijderen van een getal uit een array
12. Opsplitsen in methodes, willekeurig getal
13. Dynamische array van objecten, methodes

HoGent 3

Overzicht oefeningen

14. Recursie: code ontcijferen
15. Recursie: schema opstellen + uitvoer bepalen
16. Recursie: domeinklasse + applicatie
17. 2dim array van primitief datatype als parameter
18. Starten van applicatie met argumenten

HoGent 4

Oef 1: Wat is de waarde van x na evaluatie?

a) x = Math.abs(7.5);
b) x = Math.floor(7.5);
c) x = Math.abs(0.0);
d) x = Math.ceil(7.5);
e) x = Math.abs(-6.4);
f) x = Math.ceil(-6.4);
g) x = Math.ceil(-Math.abs(-8 + Math.floor(-5.5)));

HoGent 5

Oefening 2: schrijf een methode geefTemperatuurStatus

- Er zijn 2 methodes:
  - main
  - geefTemperatuurStatus

De code van de main-methode vind je terug op de volgende slide. Vanuit deze methode wordt een nieuwe methode geefTemperatuurStatus aangeroepen.

Leid uit de code volgende zaken af:

- Klasse waarin deze methode staat
- Volledige definitie van de methode

Implementeer dan ook deze methode.

HoGent 6

Oefening 2: schrijf een methode geefTemperatuurStatus

```
public static void main(String[] args)
{
  Scanner input = new Scanner(System.in);
  String schaal = "Celsius";
  System.out.printf("Geef de temperatuur in graden %s (9999 om te stoppen): ", schaal);
  int temp = input.nextInt();
  while (temp != 9999)
  {
        String resultaat = geefTemperatuurStatus(temp);
        System.out.printf("%d graden %s voelt aan als %s%n", temp, schaal, resultaat);
        System.out.printf("Geef de temperatuur in graden %s (9999 om te stoppen): ", schaal);
        temp = input.nextInt();
  }
}

/*
   methode geefTemperatuurStatus:
        Als temp de in de main-methode ingevoerde temperatuur voorstelt, dan:
        • Geeft de methode "koud" terug als temp < 10
        • Geeft de methode "lauw" terug als temp behoort tot [10,20]
        • Geeft de methode "warm" terug als temp > 20
*/
```

HoGent 7

Oefening 3. Schrijf een applicatie die …

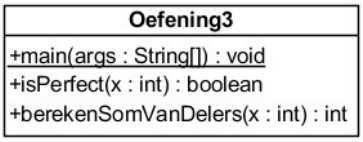
Nagaat welke gehele getallen perfect zijn uit de reeks natuurlijke getallen waarvan de kleinste en grootste waarde door de gebruiker worden ingevoerd. **Controleer of getal1 < getal 2**! Een getal is perfect als het gelijk is aan de som van zijn delers, bv: 6 = 1 + 2 + 3.

Schrijf hiervoor een non-static methode **isPerfect** die nagaat of zijn argument een perfect getal is.

De methode perfect zelf gebruikt de non-static methode **berekenSomVanDelers** die je ook uitwerkt en die de som van de delers van zijn argument aflevert.

HoGent 8

Oefening 3. Schrijf een applicatie die …




```
run:
Geef het eerste getal: 1
Geef het tweede getal (> 1): 10000
De perfecte getallen tussen 1 en 10000 zijn: 6 28 496 8128
BUILD SUCCESSFUL (total time: 7 seconds)
```

```
run:
Geef het eerste getal: 100000
Geef het tweede getal (> 100000): 10000
Geef het eerste getal: 10000
Geef het tweede getal (> 10000): 100000
Er zijn geen perfecte getallen in dit interval
BUILD SUCCESSFUL (total time: 29 seconds)
```

HoGent 9

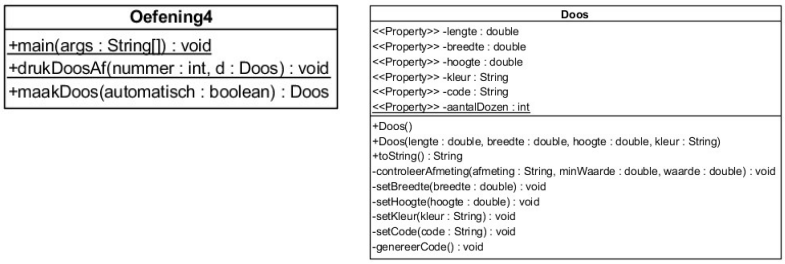
Oefening 4.

In de volgende applicatie werken we met een applicatieklasse én een domeinklasse. Beiden hebben meerdere methodes (static / niet static)

Inoefenen van:
- tekstweergave van een object
- overloading constructoren
- aanroepen methodes
- Exception gooien in de domeinklasse

HoGent 10

Oefening 4.



Uitvoer:

```
Doos 1: Een doos met lengte 1,00, met hoogte 1,00, met breedte 1,00 en kleur rood.
Deze doos heeft als unieke code D000000000000001.

Doos 2: Een doos met lengte 5,00, met hoogte 8,00, met breedte 6,00 en kleur geel.
Deze doos heeft als unieke code D000000000000002.

Dozen aanwezig in de applicatie: 2
```

HoGent 11

Oefening 4.

Domeinklasse Doos:

- Definieer de nodige attributen met nodige getters en setters (zie UML).

  lengte, **breedte**, **hoogte**: moet een strikt positief getal zijn. 3 keer zelfde controle vermijden door de aparte methode controleerAfmeting te gebruiken.

  kleur: mag niet leeg zijn

  aantalDozen: houdt bij hoeveel dozen de applicatie maakt.

  code: elke doos krijgt een unieke code (aparte methode: genereerCode. De code bestaat uit de letter D gevolgd door 15 cijfers, het aantalDozen vooraf gegaan door het juist aantal nullen, deze methode roept dan de setter van code aan om attribuut correct in te stellen).

HoGent 12

Oefening 4.

Domeinklasse Doos:

- Constructoren
  - Default constructor

    Maakt een Doos-object waarbij lengte, breedte en hoogte de waarde 1,0 krijgen en de kleur van deze doos is altijd rood.
  - Constructor met parameters voor alle attributen

    Maakt een Doos-object waarbij lengte, breedte, hoogte en kleur ingesteld worden op de waarde van de parameters.

  Opgelet: van zodra je één van beide constructoren aanroept, maak je een extra doos. Zorg dat hiervoor de nodige aanpassingen gebeuren.
- Tekstweergave van het object

  Voorzie hiervoor de correcte methode. Bekijk de uitvoer voor de correcte implementatie.

HoGent 13

Oefening 4.

Applicatie

main

- Maak een default doos d1
- Maak een specifieke doos d2
- Druk deze twee dozen af
- Druk af hoeveel dozen de applicatie gemaakt heeft.

drukDoosAf

Volg de uitvoer: genummerde doos + tekstweergave van het object

maakDoos

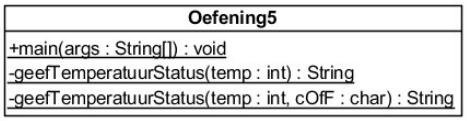
Afhankelijk van de parameter maak je een default doos door de defaultconstructor aan te roepen of maak je een doos met de vaste afmetingen: lengte=5, breedte=6, hoogte=8, kleur=geel.

HoGent 14

Oefening 5. Wijzig de methode geefTemperatuurStatus.

- Maak de methode geefTemperatuurStatus meer algemeen bruikbaar. De temperatuur moet zowel in graden Celsius (oorspronkelijke versie) als in Fahrenheit doorgegeven kunnen worden. Het eindresultaat blijft uiteraard hetzelfde.

  de formule: C = (F – 32) * 5 / 9



HoGent 15

Oefening 5. Wijzig de methode geefTemperatuurStatus.

Uitvoer:

```
run:
Geef de schaal: Celcius of Fahrenheit (C=1 of F=2): 1
Geef de temperatuur (9999 om te stoppen): 20
20 graden Celcius voelt aan als lauw

Geef de schaal: Celcius of Fahrenheit (C=1 of F=2): 2
Geef de temperatuur (9999 om te stoppen): 20
20 graden Fahrenheit voelt aan als koud

Geef de schaal: Celcius of Fahrenheit (C=1 of F=2): 1
Geef de temperatuur (9999 om te stoppen): 9999
BUILD SUCCESSFUL (total time: 12 seconds)
```

HoGent 16

Oefening 6. Schrijf een applicatie die …

```
6   public class Oefening6 {
7
8       public static void main(String args[]) {
9           Scanner scan = new Scanner(System.in);
10
11          System.out.print("Geef de zijde van het vierkant in : ");
12          int z = scan.nextInt();
13          scan.nextLine(); //nodig om buffer volledig leeg te maken
14
15          System.out.print("Geef het karakter in : ");
16          String kar = scan.nextLine();
17
18          String res;
19          if (kar.length() == 0) {
20              res = Vierkant.maakVierkant(z);
21          } else {
22              res = Vierkant.maakVierkant(z, kar.charAt(0));
23          }
24
25          System.out.println(res);
26      }
27
28  }
```

Schrijf in de klasse Vierkant de methoden "maakVierkant" met één en twee argumenten.

HoGent

Oefening 6. Schrijf een applicatie die …

Uitvoer:

```
run:
Geef de zijde van het vierkant in : 5
Geef het karakter in : c
ccccc
ccccc
ccccc
ccccc
ccccc

BUILD SUCCESSFUL (total time: 7 seconds)
```

HoGent

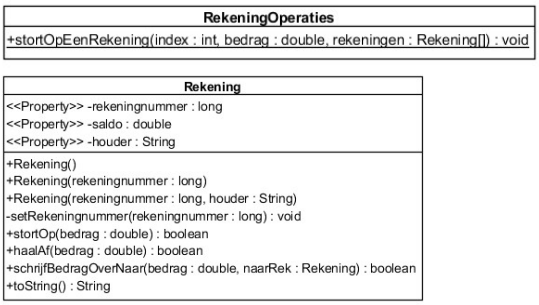
Oefening 7. Methodes in domeinklasse

- Domeinklasse Rekening
  - methode **stortOp**
    Enkel positieve bedragen kunnen gestort worden. Als het storten lukt, geef je true terug.
  - methode **haalAf**
    Enkel positieve bedragen kunnen afgehaald worden en het saldo moet toereikend zijn. Als het afhalen lukt, geef je true terug.
  - methode **schrijfBedragOverNaar**
    Haal het bedrag van af van de huidige rekening en stort die op de meegegeven rekening. Enkel als het afhalen lukt, kan je storten. Als dat storten niet lukt, moet het geld teruggeplaatst worden op de huidige rekening. Geld mag niet zomaar verdwijnen! De methode geeft true terug als het overschrijven lukt.

HoGent 19

Oefening 7

- domein:



- ui:

HoGent 20

Oefening 7

- Schrijf de static methode **stortOpEenRekening** in klasse RekeningOperaties.

  Het argument is een array van Rekeningen, een index en een bedrag.

  In deze methode wordt het bedrag gestort op de rekening met de meegegeven index. Er wordt gebruik gemaakt van de instantie-methode ***stortOp*** van de klasse **Rekening**.

  Let op: vermijd een ArrayIndexOutOfBoundsException!

HoGent

Oefening 7

Uitvoer van de applicatie:

```
run:
Geef nummer van de rekening [1 - 3]: 3
Geef het bedrag: 500
Wil je nog storten op een rekening (ja=1): 1
Geef nummer van de rekening [1 - 3]: 0
Geef nummer van de rekening [1 - 3]: 5
Geef nummer van de rekening [1 - 3]: 2
Geef het bedrag: 300
Wil je nog storten op een rekening (ja=1): 0
Beginsituatie
De rekening met rekeningnummer 123123456712 behoort toe aan Sam en heeft als saldo €0,00
De rekening met rekeningnummer 123456789012 behoort toe aan Arno en heeft als saldo €300,00
De rekening met rekeningnummer 101010101010 behoort toe aan Stef en heeft als saldo €500,00
Eindsituatie
De rekening met rekeningnummer 123123456712 behoort toe aan Sam en heeft als saldo €20,00
De rekening met rekeningnummer 123456789012 behoort toe aan Arno en heeft als saldo €350,00
De rekening met rekeningnummer 101010101010 behoort toe aan Stef en heeft als saldo €480,00
BUILD SUCCESSFUL (total time: 22 seconds)
```

HoGent

Oefening 7

- Methode **stortOpRekeningen**

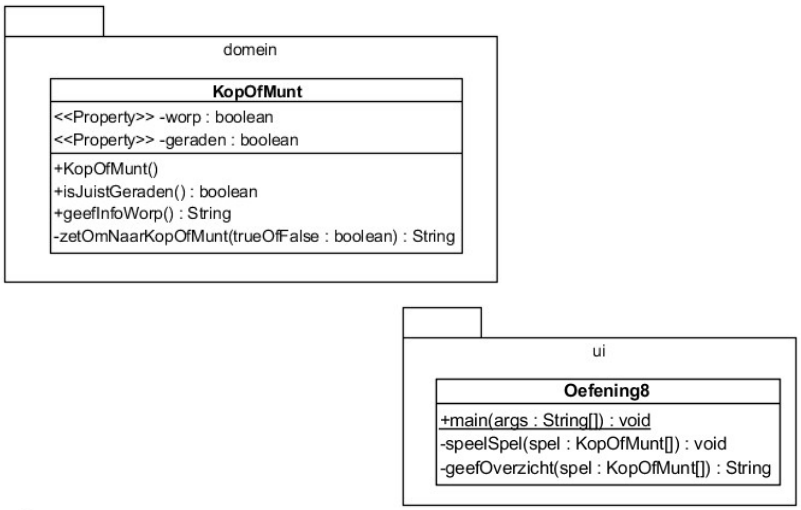
  Vraag de gebruiker op welke rekening gestort moet worden en welk bedrag. Roep dan de methode aan die effectief de storting uitvoert (stortOpEenRekening uit klasse RekeningOperaties), zie uitvoer
- Methode **toonRekeningen**

  Roept per rekening de methode toon1Rekening op.
- Methode **toon1Rekening**

  Drukt de tekstweergave van deze rekening af.

HoGent

Oefening 8



HoGent 24

Oefening 8

- Schrijf de domeinklasse KopOfMunt die voldoet aan de gegeven testklasse KopOfMuntTest en die volgende elementen bevat (zie ook UML):
  - 2 attributen: worp en geraden, beide van type boolean (true betekent kop, false betekent munt)
  - Een constructor die de worp willekeurig instelt
  - Een getter voor de worp
  - Een setter (zonder voorwaarde) voor het attribuut geraden
  - Methode isJuistGeraden: geeft true terug indien worp en geraden dezelfde waarde hebben, anders false
  - Methode geefInfoWorp: geeft alle gegevens van het huidige object (worp, geraden en "juist"/"fout"). Maak voor het tonen van worp en geraden gebruik van de methode zetOmNaarKopOfMunt en gebruik 3 kolommen van telkens 10 karakters om de output te formatteren. Zie ook voorbeeldoutput op volgende slide.
  - Methode zetOmNaarKopOfMunt: zet de meegegeven boolean waarde om naar "kop" (voor true) of "munt" (voor false).

HoGent 25

Oefening 8

- Schrijf de applicatieklasse KopOfMuntApplicatie die volgende 3 methodes bevat (zie ook UML):
  - main-methode:
    - ✓ Bepaalt het aantal worpen (hier: 10)
    - ✓ Maakt een array aan met het gepaste aantal KopOfMunt-objecten
    - ✓ Roept de (private) methodes speelSpel en geefOverzicht aan, telkens met als parameter de array
  - methode speelSpel:
    - ✓ Maakt per array-element een KopOfMunt-object aan
    - ✓ Laat de speler per array-element raden of de worp kop of munt is en stel deze waarde in in het attribuut geraden
  - methode geefOverzicht:
    - ✓ Bereidt de output voor (zie voorbeeld op volgende slide)
    - ✓ Berekent de score

HoGent 26

Oefening 8

Voorbeeld uitvoer:

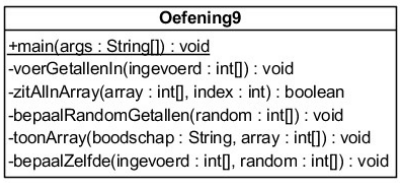
```
run:
Welkom bij het spel "Kop of munt"
Raad 10 keer of een worp kop of munt zal opleveren.
Uw score wordt bepaald door het aantal juiste voorspellingen
Worp 1: wordt het kop (1) of munt (2)? 1
Worp 2: wordt het kop (1) of munt (2)? 2
Worp 3: wordt het kop (1) of munt (2)? 1
Worp 4: wordt het kop (1) of munt (2)? 2
Worp 5: wordt het kop (1) of munt (2)? 1
Worp 6: wordt het kop (1) of munt (2)? 2
Worp 7: wordt het kop (1) of munt (2)? 2
Worp 8: wordt het kop (1) of munt (2)? 1
Worp 9: wordt het kop (1) of munt (2)? 2
Worp 10: wordt het kop (1) of munt (2)? 1

      WORP   GERADEN EVALUATIE
       kop       kop     juist
       kop      munt      fout
       kop       kop     juist
       kop      munt      fout
       kop       kop     juist
      munt      munt     juist
       kop      munt      fout
       kop       kop     juist
      munt      munt     juist
       kop       kop     juist

Je haalt 7 op 10

BUILD SUCCESSFUL (total time: 1 minute 51 seconds)
```


Oefening 9



Schrijf een applicatie met volgende methodes (zie ook UML):

- **main**: maak 2 arrays aan die elk 5 getallen kunnen bevatten en roep daarna achtereenvolgens de methodes **voerGetallenIn, bepaalRandomGetallen, toonArray** (1x met de ingevoerde en 1x met de random getallen als parameter) en **bepaalZelfde** aan, telkens met de juiste parameter(s).

Oefening 9

- **voerGetallenIn**: laat de gebruiker 5 unieke getallen uit het interval [0,10] inlezen, waarbij een passende melding wordt gegeven per soort fout en er direct een herkansing wordt aangeboden (zie ook voorbeelduitvoer)
- **zitAlInArray**: controleer of het element op een gegeven index al op een vorige index in de gegeven array te vinden is
- **bepaalRandomGetallen**: laat het systeem een gegeven array opvullen met unieke random getallen uit het interval [0,10], waarbij gebruik gemaakt wordt van de methode zitAlInArray om te checken of het getal uniek is
- **toonArray**: toon eerst de opgegeven boodschap en vervolgens de rij getallen uit de array in kolommen van 3 karakters breed
- **bepaalZelfde**: vergelijk de 2 gegeven arrays, waarbij de getallen die in beide arrays voorkomen, gescheiden door een spatie, worden bijgehouden in een String en deze String of een passende foutmelding (zie voorbeelduitvoer) wordt geprint

HoGent 29

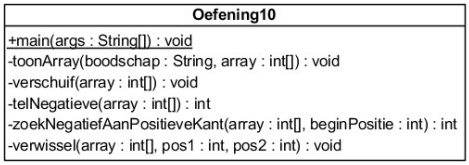
Oefening 9

```
run:
Geef getal 1: 12
Getal mag niet groter zijn dan 10!
Geef getal 1: -1
Getal mag niet kleiner zijn dan 0!
Geef getal 1: 0
Geef getal 2: 4
Geef getal 3: 0
Dit getal heb je al gekozen! Probeer opnieuw!
Geef getal 3: 7
Geef getal 4: 2
Geef getal 5: 5
Door jou gekozen getallen
  0  4  7  2  5
Door het systeem gekozen getallen
 10  7  1  5  6
De getallen die in beide arrays voorkomen zijn: 7 5
BUILD SUCCESSFUL (total time: 30 seconds)
```

```
run:
Geef getal 1: 3
Geef getal 2: 6
Geef getal 3: 5
Geef getal 4: 0
Geef getal 5: 10
Door jou gekozen getallen
  3  6  5  0 10
Door het systeem gekozen getallen
  8  2  4  1  9
In de ingevoerde array zitten geen getallen die ook in de random array voorkomen
BUILD SUCCESSFUL (total time: 27 seconds)
```

HoGent 30

Oefening 10



Schrijf een applicatie met volgende methodes (zie ook UML):

- **main**: definieer een array met een willekeurig aantal elementen die gehele getallen voorstellen en maak gebruik van de methodes **toonArray** om de oorspronkelijke en de aangepaste array te tonen en de methode **verschuif** om de elementen in de array anders te rangschikken. In het voorbeeld wordt als array int[] a = {-5, 2, 7, -4, 3, 9, -1}; gebruikt.
- **toonArray**: toon eerst de opgegeven boodschap en vervolgens de rij getallen uit de array in kolommen van 3 karakters breed (zie ook oefening 9)

HoGent 31

Oefening 10

- **verschuif**: maak gebruik van de methode **telNegatieve** om het aantal negatieve elementen (stel: x) in de array te kennen; doorloop vervolgens de x eerste elementen van de array en indien je daar op een index y een positief getal vindt, zoek dan een index z waarop zicht een positief getal bevindt dat niet op zijn plaats staat (methode **zoekPositiefAanNegatieveKant**) en wissel de elementen op index y en z om (methode **verwissel**)
- **telNegatieve:** doorloop de opgegeven array en verhoog de teller telkens je een negatief getal tegenkomt; return de teller
- **zoekNegatiefAanPositieveKant**: gegeven een beginpositie in een opgegeven array, doorloop de array vanaf deze positief en check of het getal op deze positie/index negatief is – zo ja, return de index
- **verwissel**: verwissel binnen de gegeven array de elementen op de 2 opgegeven indexen van plaats

HoGent 32

Oefening 10

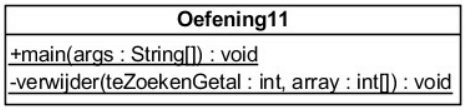
```
run:
oorspronkelijke array
 -5  2  7 -4  3  9 -1
na verschuiving
 -5 -4 -1  2  3  9  7
BUILD SUCCESSFUL (total time: 0 seconds)
```

Of met een array met alleen maar positieve (of negatieve) elementen…

```
run:
oorspronkelijke array
 77 67 71 74 45 44 23  0
na verschuiving
 77 67 71 74 45 44 23  0
BUILD SUCCESSFUL (total time: 0 seconds)
```

HoGent 33

Oefening 11



Initialiseer een array met 10 willekeurige gehele getallen. (De array is een lokale variabele.)

Vraag vervolgens om een willekeurig geheel getal in te voeren.

Schrijf een **methode**, die het ingevoerde getal zoekt en verwijdert in de array. Voeg op het einde van de array aan met nullen.

Voorbeeld:

a={4,8,2,**3**,5,17,7,99,**3**,12} en getal = 3

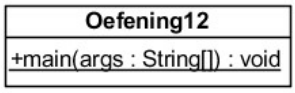
a wordt

a = { 4,8,2,5,17,7,99,12,**0**,**0**}

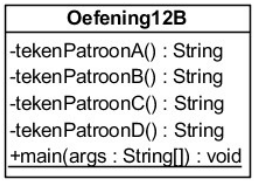
HoGent 34

Oefening 12. Patroonapplicatie

a) Open en run Oefening12.java

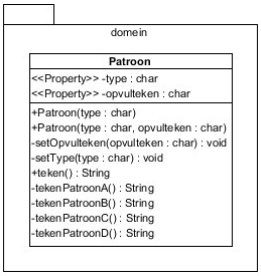


b) Herschrijf deze applicatie met 4 non-static methodes (tekenPatroonA, tekenPatroonB, tekenPatroonC en tekenPatroonD (zie oef H5)).



HoGent

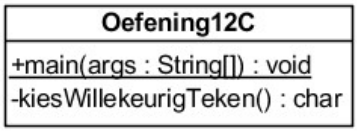
c) We werken nu met een domeinklasse en een applicatieklasse.



Maak de **domeinklasse** Patroon.

- De 4 methodes om de patronen uit te tekenen kan je (gedeeltelijk) overnemen uit opgave b).
- De methode teken() kiest naargelang de waarde van het attribuut type voor de juiste methode tekenPatroon**X**().
- Zorg voor de attributen, constructors, getters en setters.
  - → Type kan enkel 'A', 'B', 'C' of 'D' zijn, opvulteken bevat '*', '-', '+', '!' of '?'.
  - → Gooi een IllegalArgumentException bij foute waarden.

HoGent 36



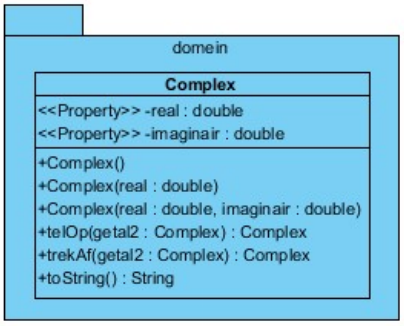
Maak de **applicatieklasse** Oefening12C.

- Methode **main:**
  Je vraagt aan de gebruiker het type (‘A’, ‘B’, ‘C’, of ‘D’ – ‘Z’ om te stoppen).
  Je maakt een object van Patroon aan en je geeft een type door.
  Daarna genereer je een willekeurig karakter met de methode **kiesWillekeurigTeken** en roep je de methode teken() aan zodat het gevraagde patroon, opgevuld met het gegenereerde karaker, getoond kan worden. Plaats boven het patroon ook telkens een titel.

- Methode **kiesWillekeurigTeken:**
  genereert een willekeurig karakter om het patroon mee op te vullen, te kiezen uit de geldige opvultekens, namelijk ‘*’, ‘+’, ‘-’, ‘!’ en ‘?’.

HoGent 37

Oefening 13



Maak een klasse Complex

Attributen: real en imaginair
(double) (real + imaginair * i)

Methoden:

- default-constructor (real = imaginair = 0.0)
- constructor met één parameter (imaginair = 0.0)
- constructor met 2 parameters
- de methode telOp() om de optelling van 2 complexe getallen te bepalen
- de methode trekAf() om de aftrekking van 2 complexe getallen te bepalen
- de methode toString() definiëren voor deze klasse; de string moet het complex getal weergeven in de vorm (real;imaginair)

HoGent 38

Oefening 13

- Maak gebruik van de 3 constructoren om objecten te maken van Complex en stop de 3 objecten in een dynamische array.
- Druk de tekstversie van de objecten af op het scherm.
- Neem het tweede object en derde object, maak daarvan de som. Deze som voeg je dan ook toe aan de dynamische array.
- Druk de tekstversie van de objecten nogmaals af op het scherm.

```
run:
(0,0;0,0)
(5,0;0,0)
(6,2;7,8)
Na de optelling:
(0,0;0,0)
(5,0;0,0)
(6,2;7,8)
(11,2;7,8)
BUILD SUCCESSFUL (total time: 0 seconds)
```

HoGent 39

Oefening 14: Wat doet de volgende methode?

```
// parameter b moet een positief integer zijn
// om oneindige recursie te vermijden
// werk uit met a = 4, b = 3

public int mysterie (int a, int b)
{
    if (b == 1)
        return a;
    else
        return a + mysterie ( a, b – 1);
}
```

HoGent 40

Oefening 15. Bepaal uitvoer + schema

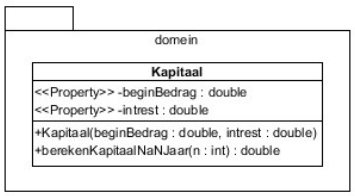
Evalueer de volgende recursieve methode aan de hand van een schema en bepaal de uitvoer voor n = 3:

```
void f (int n)
{
   if (n > 0)
   {
         f(n-2);
         System.out.printf("%3d",n);
         f(n-1);
   }
}
```

HoGent 41

Oefening 16.

Maak de domeinklasse Kapitaal volgens volgende UML:



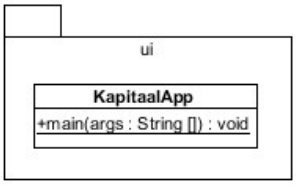
Het beginBedrag moet minimaal 0 euro zijn.
De intrest is een percentage en ligt dus in het interval [0,100].
De methode berekenKapitaalNaNJaar is recursief.

Voorbeeld:
Jan belegt 1000 euro aan een samengestelde intrest van 8%.
Zijn kapitaal na n jaar = K(n)
formule : K(n) = 1.08 * K(n-1) met K(0) = 1000

HoGent 42

Oefening 16.

Maak daarna ook de applicatie volgens de volgende UML:



In de main-methode wordt het startkapitaal, de intrest en het aantal jaar gevraagd.
Na invoer wordt een object van Kapitaal gemaakt waarmee het kapitaal na het gevraagde aantal jaar berekend wordt.
Zie ook de voorbeelden op de volgende slides!

HoGent 43

Oefening 16.

Voorbeelden bij foute invoer:

```
run:
Geef het startkapitaal (0 om te stoppen): -1000
Geef het aantal jaar: 5
Geef het intrestpercentage: 1,23
Exception in thread "main" java.lang.IllegalArgumentException: Beginbedrag moet minimaal 0 zijn
        at domein.Kapitaal.controleerBeginBedrag(Kapitaal.java:24)
        at domein.Kapitaal.<init>(Kapitaal.java:9)
        at ui.KapitaalApplicatie.main(KapitaalApplicatie.java:23)
Java Result: 1
BUILD SUCCESSFUL (total time: 18 seconds)

run:
Geef het startkapitaal (0 om te stoppen): 1000
Geef het aantal jaar: 2
Geef het intrestpercentage: 123
Exception in thread "main" java.lang.IllegalArgumentException: Intrest moet in het interval [0,100] liggen
        at domein.Kapitaal.controleerIntrest(Kapitaal.java:36)
        at domein.Kapitaal.<init>(Kapitaal.java:11)
        at ui.KapitaalApplicatie.main(KapitaalApplicatie.java:23)
Java Result: 1
BUILD SUCCESSFUL (total time: 18 seconds)
```

HoGent 44

Oefening 16.

Voorbeelden bij juiste invoer:

```
run:
Geef het startkapitaal (0 om te stoppen): 1000
Geef het aantal jaar: -1
Geef het aantal jaar: 2
Geef het intrestpercentage: 2
Het kapitaal van €1000,00 groeit bij een interest van 2,00 na 2 jaar aan tot €1040,40.
Geef het startkapitaal (0 om te stoppen): 2000
Geef het aantal jaar: 10
Geef het intrestpercentage: 1,23
Het kapitaal van €2000,00 groeit bij een interest van 1,23 na 10 jaar aan tot €2260,07.
Geef het startkapitaal (0 om te stoppen): 2000
Geef het aantal jaar: 25
Geef het intrestpercentage: 1,35
Het kapitaal van €2000,00 groeit bij een interest van 1,35 na 25 jaar aan tot €2796,56.
Geef het startkapitaal (0 om te stoppen): 12345,67
Geef het aantal jaar: 15
Geef het intrestpercentage: 1,89
Het kapitaal van €12345,67 groeit bij een interest van 1,89 na 15 jaar aan tot €16348,88.
Geef het startkapitaal (0 om te stoppen): 0
BUILD SUCCESSFUL (total time: 1 minute 38 seconds)
```


Oefening 17

Gegeven:

```
public class OefTheorie
 {
   public static void main(String arg[])
   {
     double gem;
     int table[][] = new int [3][2];
     voerGetalIn(table);
     gem = berekenGemiddelde(table);
     String uitvoer = String.format("gemiddelde is %.1f“, gem);
     System.out.print(uitvoer);
   } //einde methode main
```

Oef5
+main(arg : String []) : void
-voerGetalIn(table : int [][]) : void
-berekenGemiddelde(table : int [][]) : double

Schrijf de methodes invoer en gemiddelde.

Methode voerGetalIn**:** de gebruiker geeft alle elementen van de array in.

De methode **“**berekenGemiddelde**”** berekent het gemiddelde van alle elementen van de array.

Oefening 18

Creëer de twee dimensionale array “rekening” van type Rekening.

De array bestaat uit 3 rijen.

Het aantal kolommen per rij wordt meegegeven bij het uitvoeren van de applicatie.

Per element van de array creëer je een object van Rekening en vul je het saldo als volgt op:

```
run:
10,00 20,00 30,00 40,00
11,00 21,00 31,00
12,00 22,00
BUILD SUCCESSFUL (total time: 0 seconds)
```

HoGent 47