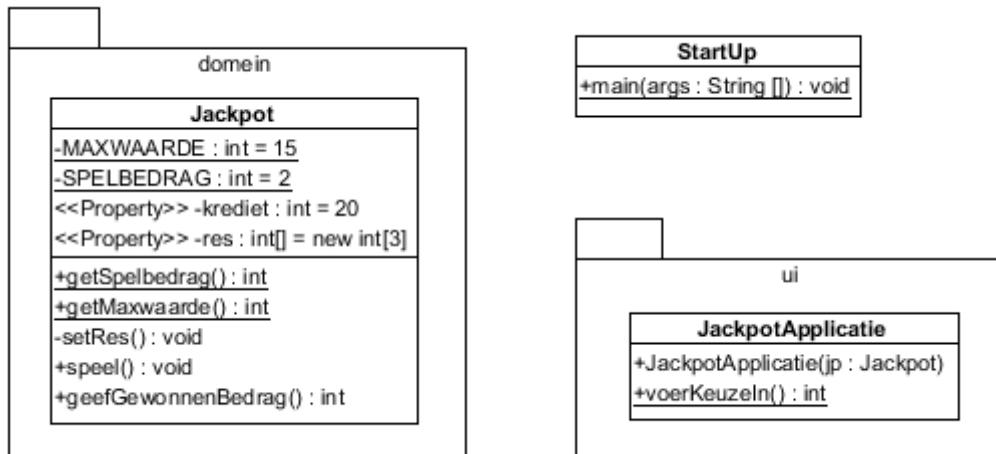


## 1. Jackpot

UML :



De toepassing simuleert een jackpot. Bij het opstarten krijgt de gebruiker een startkrediet toegewezen (van 20 munten) waarmee kan gespeeld worden. Elk spel kost een vast bedrag (2 munten) dat wordt afgetrokken van het beschikbaar krediet. Eventuele winsten worden opgeteld bij het krediet dat nog beschikbaar is. De speler kan spelen tot het kredietbedrag is opgebruikt!

De jackpot bestaat uit drie cijfers, die voor elk spel bepaald worden door een random-generator (getallen tussen 0 en 15 = maximumwaarde).

Het win-schema ziet er als volgt uit:

<b>3 nullen</b>	<b>= 4 maal de inzet</b>
<b>2 nullen</b>	<b>= 2 maal de inzet</b>
<b>1 nul</b>	<b>= 1 maal de inzet</b>
<b>drie gelijke cijfers</b>	<b>= 2 maal de inzet</b>

Maak de domeinklasse **Jackpot** met:

- Attributen : maximumwaarde (=15), krediet (=20), spelbedrag (=2), res[]
- Methoden :
  - getKrediet()
  - getSpelbedrag()
  - getMaximumwaarde()
  - getRes() en setRes()
  - speel(): past het kredietbedrag aan; via setRes() worden 3 nieuwe waarden bepaald voor de random getallen en via de methode geefGewonnenBedrag() wordt het kredietbedrag eventueel aangepast
  - geefGewonnenBedrag(): bepaalt het (eventueel) gewonnen bedrag

Werk een applicatie uit, die via een menu de volgende keuzes laat aan de speler:

- 1) nieuw spel**
- 2) speel → toont het behaalde resultaat**
- 3) stop het spel**
- 4) toon krediet**
- 5) toon spelbedrag**
- 6) stop programma**

De applicatie wordt gestart via de StartUp-klasse waarbij in de main-methode een leeg Jackpot-object wordt gemaakt dat wordt meegegeven aan de JackpotApplicatie-constructor.

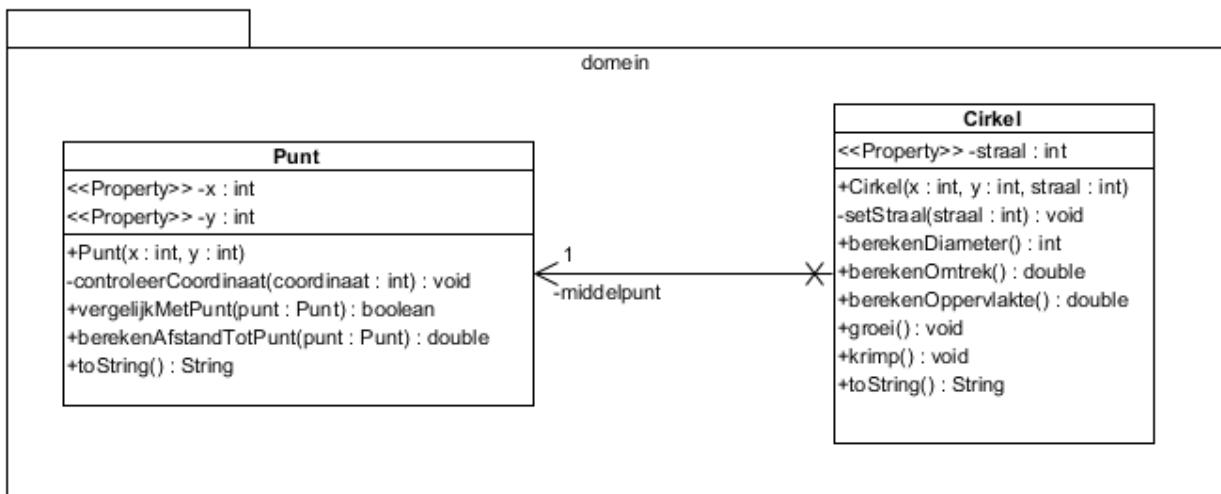
Indien men het spel probeert te spelen of het huidige krediet probeert te tonen van een spel dat nog niet aangemaakt werd, wordt een foutmelding getoond. Je moet dus altijd eerst een nieuw spel starten (optie 1) voor je optie 2 of 4 uit het menu kan kiezen. Optie 3 zorgt ervoor dat het huidig spel vernietigd wordt. Optie 5 kan te allen tijde gekozen worden, aangezien de gebruiker moet kunnen weten hoeveel munten hij moet hebben om het spel te kunnen spelen.

```
Geeft uw keuze :  
1) nieuw spel  
2) speel  
3) stop het spel  
4) toon krediet  
5) toon spelbedrag  
6) stop programma  
Keuze: 2  
Er is geen spel actief of je hebt niet voldoende krediet!  
Geeft uw keuze :  
1) nieuw spel  
2) speel  
3) stop het spel  
4) toon krediet  
5) toon spelbedrag  
6) stop programma  
Keuze: 5  
Het spelbedrag = 2
```

```
Geef uw keuze :  
1) nieuw spel  
2) speel  
3) stop het spel  
4) toon krediet  
5) toon spelbedrag  
6) stop programma  
Keuze: 1  
Geef uw keuze :  
1) nieuw spel  
2) speel  
3) stop het spel  
4) toon krediet  
5) toon spelbedrag  
6) stop programma  
Keuze: 2  
      5  2   7  
Krediet = 18  
Geef uw keuze :  
1) nieuw spel  
2) speel  
3) stop het spel  
4) toon krediet  
5) toon spelbedrag  
6) stop programma  
Keuze: 2  
      3 14   1  
Krediet = 16  
Geef uw keuze :  
1) nieuw spel  
2) speel  
3) stop het spel  
4) toon krediet  
5) toon spelbedrag  
6) stop programma  
Keuze: 4  
Het krediet = 16  
Geef uw keuze :  
1) nieuw spel  
2) speel  
3) stop het spel  
4) toon krediet  
5) toon spelbedrag  
6) stop programma  
Keuze: 6  
BUILD SUCCESSFUL (total time: 3 minutes 3 seconds)
```

## 2. Cirkel

UML domein:

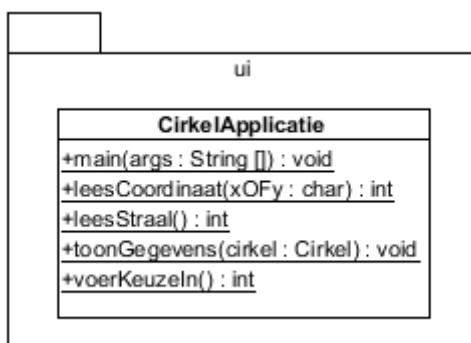


Hergebruik de domeinklasse **Punt** uit hoofdstuk 6 en vul aan met een `toString()`-methode die het punt afbeeldt als (x , y).

Maak een tweede domeinklasse **Cirkel** met :

- Attributen:
  - middelpunt (een Punt-object)
  - straal (integer)
- Methodes:
  - constructor
  - `getMiddelpunt()`: getter voor het middelpunt
  - `getStraal()`: getter voor de straal
  - `setStraal()`: straal moet altijd tussen 1 en 500 (grenzen inbegrepen) liggen
  - `berekenDiameter()`: berekent en geeft de diameter terug
  - `berekenOmtrek()`: berekent en geeft de omtrek terug
  - `berekenOppervlakte()`: berekent en geeft de oppervlakte terug
  - `groei()`: laat de straal 5 eenheden groeien (blijf controleren op de voorwaarde)
  - `krimp()`: idem als `groei()`, maar verklein nu de straal met 5 eenheden

UML applicatie:



Maak een applicatieklasse die de functionaliteit van bovenstaande domeinklassen gebruikt

- In de **main**-methode laat je de gebruiker eerst het middelpunt en de straal van een cirkel ingeven via de methode **leesCoordinaat** (2x oproepen) en **leesStraal**. Hiermee maak je een Cirkel-object dat je alvast een eerste keer laat zien met de methode **toonGegevens**.
- Gebruik hierna de methode **voerKeuzeIn** om een menu te tonen met de volgende opties:
  1. Krimpen van de bestaande cirkel
  2. Groeien van de bestaande cirkel
  3. Een nieuw middelpunt definiëren
  4. Het programma stoppen

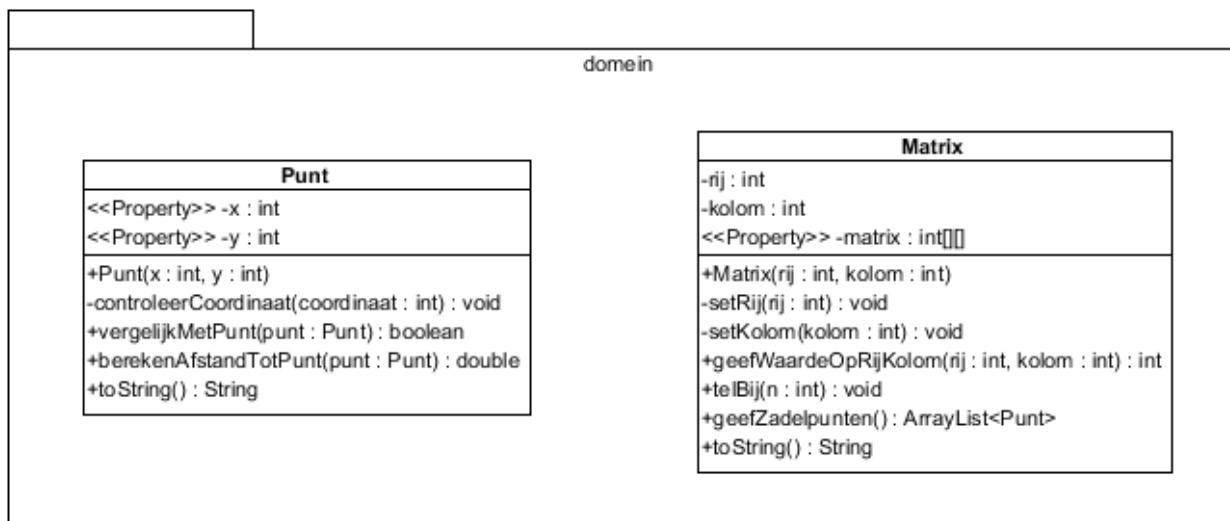
Deze methode vraagt de gebruiker een keuze te maken en retourneert deze. Zolang de keuze niet geldig is wordt deze opnieuw gevraagd.

- Naargelang de gemaakte keuze wordt de bijpassende methode uit de domeinklasse aangeroepen. Voor optie 3 worden eerst opnieuw coördinaten voor het middelpunt ingelezen via **leesCoordinaat**. Wanneer optie 1, 2 of 3 gekozen wordt, zullen de volgende resultaten (bijvoorbeeld) onmiddellijk te zien zijn op het scherm:

```
De cirkel heeft:  
zijn middelpunt op: (20 , 30)  
straal van: 55  
diameter van: 110  
een omtrek van: 345,58  
een oppervlakte van: 9503,32
```

### 3. Matrix

UML domein:



Gebruik opnieuw de domeinklasse **Punt** uit hoofdstuk 6, aangevuld met de **toString**-methode uit oefening 2 hierboven.

Maak een domeinklasse **Matrix** met

- Attributen:
  - Aantal rijen ( rij )
  - Aantal kolommen ( kolom )
  - 2-dim array ( matrix )
- Methoden:
  - Constructor met 2 parameters (aantal rijen en kolommen): maakt een matrix met het gewenste aantal rijen en kolommen
  - setRij en setKolom: controleert of er minstens 1 rij en kolom is
  - setMatrix: controleert of de opgegeven matrix het opgegeven aantal rijen en kolommen heeft
  - geefWaardeOpRijKolom(int rij, int kolom): controleert of het opgegeven rij- en kolomnummer bestaan en geeft zo mogelijk het element op deze positie in de matrix terug
  - telBij(int n): telt bij alle elementen van de matrix de waarde n op
  - geefZadelpunten: in de speltheorie is een zadelpunt van een matrix A het koppel (i,j), zodanig dat het element in de matrix op rij i en kolom j het kleinst is in haar rij en het grootst in haar kolom. Een matrix kan meerder zadelpunten hebben. Gebruik een ArrayList van de klasse Punt om de resultaten te retourneren.

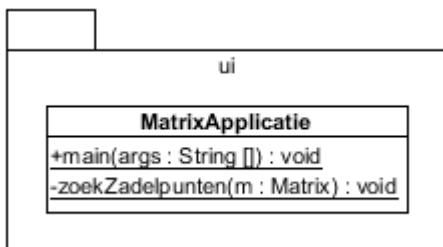
Voorbeeld:

1	2	6
7	13	16
4	8	7

Het element 7 op rij 2 en kolom 1 is een zadelpunt want 7 is het kleinste getal van de tweede rij en het grootste van de eerste kolom. Het punt (2,1) wordt dus in de lijst van zadelpunten toegevoegd.

- `toString()`: toont de matrix rij per rij, waarbij de kolommen telkens 5 posities breed zijn

UML applicatie:



Maak een tweede klasse met een main-methode, die de functionaliteit van de bovenstaande klasse gebruikt. Maak hiervoor een aantal Matrices in de main-methode en geef deze telkens door aan de methode zoekZadelpunten, die achtereenvolgens de oorspronkelijke Matrix toont, er een willekeurig getal tussen 1 en 10 bij optelt, de nieuwe Matrix toont en de lijst met zadelpunten bepaalt en toont.

Voorbeelduitvoer:

oorspronkelijke matrix:

1	2	6
7	13	16
4	8	7

4 erbij opgeteld

5	6	10
11	17	20
8	12	11

De zadelpunt(en):

11 op positie (2 , 1)

oorspronkelijke matrix:

1	6	6
1	13	16
1	8	7

10 erbij opgeteld

11	16	16
11	23	26
11	18	17

De zadelpunt(en):

11 op positie (1 , 1)

11 op positie (2 , 1)

11 op positie (3 , 1)