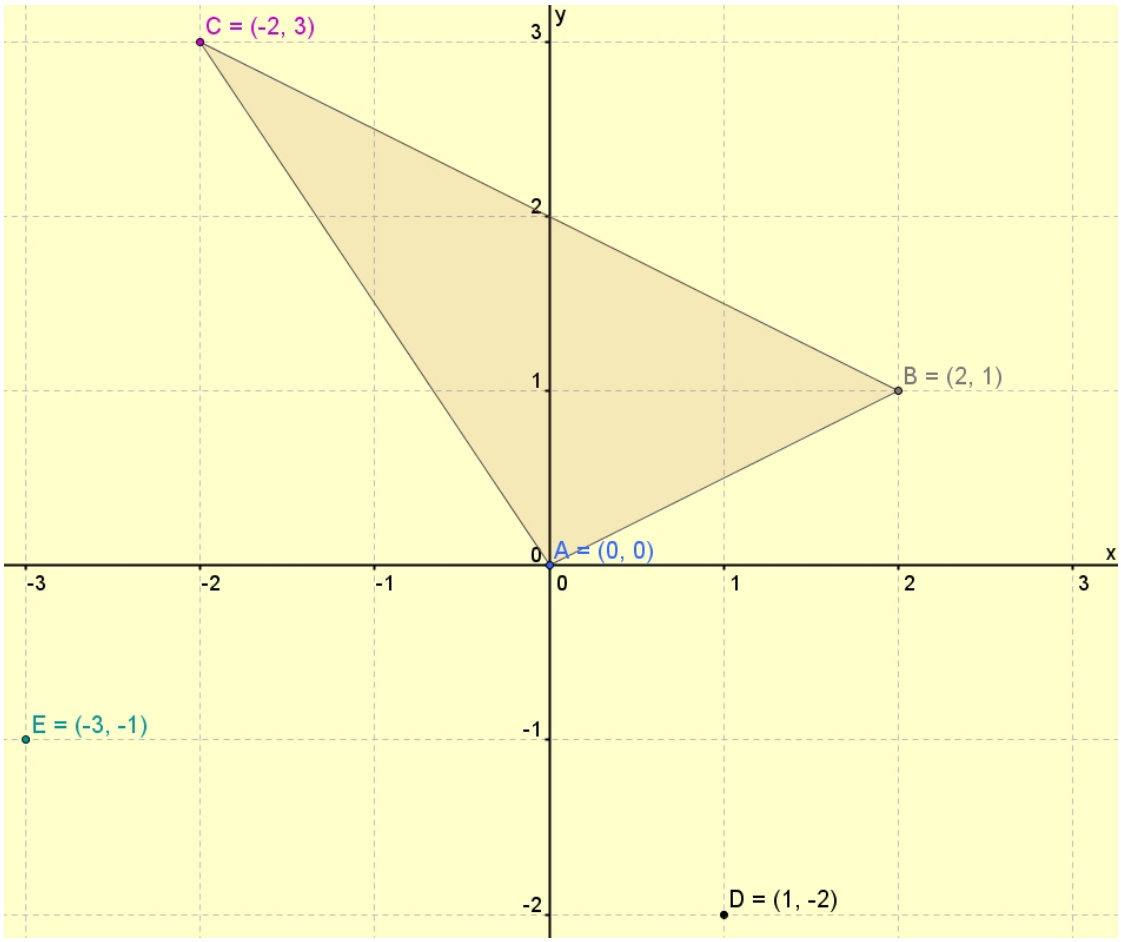
OOP1 – Oefenopdracht 3b

Punt en Veelhoek

Inleiding

Een **punt** P heeft een naam en wordt weergegeven door twee getallen, de x- en de y-coördinaat: (p_x, p_y). In de tekening zie je vijf punten met hun coördinaten. Deze zijn: A(0, 0), B(2, 1), C(–2, 3), D(1, –2) en E(–3, –1).



Een **veelhoek** heeft een naam en is gedefinieerd als een lijst van (drie of meer) objecten van het type Punt. Zo is een veelhoek die uit drie punten bestaat natuurlijk een driehoek. In de tekening zie je bijvoorbeeld de driehoek die uit de punten A, B en C bestaat.

In deze opdracht maak je de klassen **`Punt`** en **`Veelhoek`**. Deze klassen hebben een aantal methoden. De applicatie moet worden getest vanuit een **`Main`** klasse.

Om het je gemakkelijker te maken is de opdracht in vijf stappen onderverdeeld.
In opdracht A en B maak je de klasse **`Punt`** en in opdracht C t/m E maak je de klasse **`Veelhoek`**.

Opdracht A

- Maak een klasse **`Punt`** met de eigenschappen die in de beschrijving zijn genoemd (een punt heeft een naam en is gedefinieerd door zijn x- en y-coördinaat). Kies de juiste data types.
- Maak twee *constructors* voor de klasse **`Punt`**:
    - Een constructor met alleen een naam. Deze zet de naam en initialiseert het punt op (0, 0).
    - Een constructor met drie argumenten: de naam, en de x- en y-coördinaat. Deze zet de naam en initialiseert het punt op de x- en y-coördinaat die als argumenten zijn meegegeven.
- Maak in de klasse **`Punt`** een methode **`print()`** die de naam en de coördinaten van het punt afdrukt. Bijvoorbeeld: "*A(2, 4)*".
- Test de klasse **`Punt`** door in de **`main()`** methode van klasse **`Main`** het volgende te doen.
    - Maak één object van het type **`Punt`** aan door de constructor met één argument te gebruiken.
    - Vraag aan de gebruiker om de naam en de x- en y-coördinaat van het tweede punt.
    - Maak nog een object van het type **`Punt`** aan (gebruik nu de andere constructor, waarbij de input van de gebruiker wordt gebruikt).
    - Roep van beide objecten de **`print()`** methode aan en controleer of de output klopt.

Opdracht B

- Maak in de klasse **`Punt`** een methode **`verschuif(deltaX, deltaY)`**.
    - Deze methode verschuift een punt een aantal stappen (deltaX) in de x-richting (positief is naar rechts, negatief is naar links) en een aantal stappen (deltaY) in de y-richting (positief is naar boven, negatief is naar beneden).
    - Doe deze verschuiving door de eigenschappen van het punt (dat wil zeggen: de x- en y-coördinaat)aan te passen.
    - Voorbeeld: als je punt C(–2, 3) verschuift met (6, –8), dan wordt de nieuwe x-coördinaat –2 + 6 = 4 en de nieuwe y-coördinaat 3 – 8 = –5, dus het verschoven punt C wordt (4, –5).
- Test deze methode door in de **`main()`** methode van klasse **`Main`** het volgende te doen.
    - Maak een random punt aan en druk zijn coördinaten af (gebruik weer de methode **`print()`** van de klasse **`Punt`**.
    - Vraag aan de gebruiker de x- en y-verschuiving.
    - Verschuif het punt met de methode **`verschuif()`**, gebruik de verschuiving die de gebruiker heeft ingetypt.
    - Druk de nieuwe coördinaten van het verplaatste punt af en controleer of het klopt.

De klasse **`Punt`** is gereed.
We gaan nu de klasse **`Veelhoek`** maken. Deze maakt gebruik van de klasse **`Punt`**.

Opdracht C

- Maak een klasse **`Veelhoek`** met de eigenschappen die in de beschrijving zijn genoemd (een Veelhoek heeft en naam en bestaat uit een lijst van punten). Gebruik voor deze lijst een array.
- Maak twee *constructors* voor de klasse **`Veelhoek`**:
    - Een constructor met één argument: De naam. Deze zet de naam en maakt een array van drie punten die allemaal op (0, 0) staan. De namen van de punten zijn A, B, C, …
    - Een constructor met twee argumenten: de naam en het aantal punten. Dat is het aantal hoeken/punten waaruit de veelhoek bestaat.
        - Deze moet eerst testen of het aantal punten groot genoeg is (een veelhoek kan niet uit minder dan drie punten bestaan). Als het kleiner is dan drie moet er een passende foutmelding op het scherm worden afgedrukt. Vervolgens wordt gedaan alsof het aantal gekozen punten drie is wordt een driehoek gemaakt van drie punten die op (0, 0) staan. De namen van de punten zijn A, B, C, …
        - Als het wel drie of hoger is moet een array worden gemaakt. De grootte van het array wordt bepaald door het aantal punten. Alle punten in het array moeten op (0, 0) staan. De namen van de punten zijn A, B, C, …
        - De naam van de veelhoek moet ook worden geïnitialiseerd.
- Maak in de klasse **`Veelhoek`** een methode **`print()`** . Deze drukt eerst de naam af en het aantal punten van de veelhoek af. Vervolgens worden de coördinaten van ieder punt van die veelhoek afgedrukt. Bijvoorbeeld:
  "*De veelhoek Vierkant heeft 4 punten: P1(0, 0), P2(3, 0), P3(3, 3) en P4(0, 3).*"
  LET OP: Maak hierbij gebruik van de **`print()`** methode die je in opdracht A hebt gemaakt.
- Test je klasse door in de **`main()`** methode van klasse **`Main`** het volgende te doen.
    - Maak één object van het type **`Veelhoek`** aan door de constructor met één argument te gebruiken.
    - Vraag aan de gebruiker om de naam en het aantal punten van de tweede veelhoek.
    - Maak nog een object van het type **`Veelhoek`** aan (gebruik nu de andere constructor, waarbij de input van de gebruiker wordt gebruikt).
    - Roep van beide objecten de **`print()`** methode aan en controleer of de output klopt.

Je hebt nu een veelhoek gemaakt met allemaal punten die op (0, 0) staan. Dat is niet zo nuttig. Dat gaan we aanpassen in opdracht D.

Opdracht D

- Maak in de klasse **`Veelhoek`** een methode **`setPunt(int index, Punt punt)`** om een punt van de veelhoek te veranderen. Deze methode krijgt twee argumenten mee:
    - De index van het punt. Deze geeft aan welk punt van de veelhoek je wil veranderen. Bij een vijfhoek kan dat dus 0, 1, 2, 3 of 4 zijn. Kies het juiste data type.
      LET OP: Test of de index legaal is. Deze moet groter of gelijk aan 0 en kleiner dan het aantal punten van de veelhoek zijn. Zo niet, dan moet een passende foutmelding op het scherm worden afgedrukt.
    - Een object van het type **`Punt`**. Hiermee wordt het punt in de veelhoek geïnitialiseerd (als de index tenminste legaal is).
- Test deze methode door in de **`main()`** methode van klasse **`Main`** het volgende te doen.
    - Neem de veelhoek van vraag C. Vraag aan de gebruiker per punt wat de naam en de coördinaten zijn. Pas de veelhoek hiermee aan door voor ieder punt de methode **`setPunt()`** van de veelhoek aan te roepen. Roep vervolgens de **`print()`** methode van deze veelhoek aan en controleer of de output klopt.

Opdracht E

- Maak in de klasse **`Veelhoek`** een methode **`verschuif(deltaX, deltaY)`**. Deze methode verschuift alle punten van de veelhoek een aantal stappen in de x-richting (deltaX) en een aantal stappen in de y-richting (deltaY).
  LET OP: Maak daarbij gebruik van de **`verschuif(deltaX, deltaY)`** methode van de klasse **`Punt`** die je al in opdracht B hebt gemaakt!
- Test deze methode door in de **`main()`** methode van klasse **`Main`** het volgende te doen.
    - Gebruik de veelhoek die in opdracht D is gemaakt.
    - Vraag aan de gebruiker de x- en y-verschuiving.
    - Verschuif de veelhoek met de methode **`verschuif()`**, gebruik de verschuiving die de gebruiker heeft ingetypt.
    - Druk de nieuwe coördinaten van de verplaatste veelhoek af en controleer of het klopt.

Richtlijnen bij coderen (zie ook HBO-ICT code conventions [ICC])

- Zorg dat je naam en het doel van het programma bovenin staan (ICC #1).
- Gebruik de juiste inspringing (*indentation*) bij de lay-out (ICC #2) .
- Let op juist gebruik hoofdletters en kleine letters (ICC #3).
- Gebruik goede namen (ICC #4).
- Vermijd *magic number*s (ICC#5).
- Gebruik javadoc tags: **`@author`**, **`@param`** en **`@return`** (ICC #6).
- Voeg waar nodig commentaar toe die inzicht geven in je code (ICC#7).
- Denk aan *encapsulation* (ICC #9).