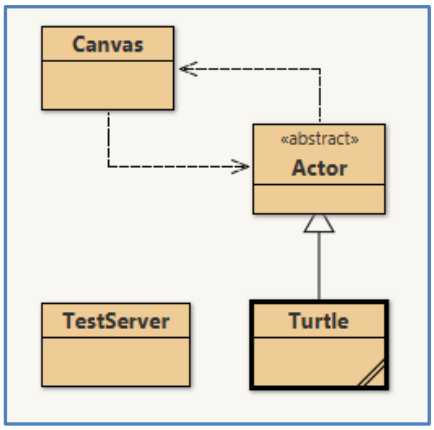
SKILDPADDE 1 (TURTLE 1)

I denne opgave skal I lave nogle metoder, som kan tegne de geometriske figurer, som I kan se længere nede i dette dokument.

Til formålet skal I bruge BlueJ-projektet ***Turtle*** (zip). Start med at downloade projektet, og husk at pakke det ud, før I går i gang.

Projektet indeholder tre klasser (samt den "sædvanlige" ***TestServer*** klasse):



Canvas klassen modellerer det "lærred", som I tegner på. Den er forholdsvis kompleks, og I skal blot opfatte den som en klasse, I kan bruge, uden at forstå implementationen. I må ikke ændre koden i denne klasse.

Actor klassen stiller en række metoder til rådighed:

- Accessor metoder der returnerer koordinaterne for skildpaddens øjeblikke position, samt den vinkel den har. Herudover kan man få at vide, om den har pennen nede (dvs. trækker en streg efter sig) samt hvilken farve pennen har.
- Mutator metoder der ændrer skildpaddens koordinater, vinkel, farve og penposition.
- En metode, ***pause***, der får skildpadden til at vente 100 msek, hvilket gør det meget lettere at følge dens bevægelser. Metoden kan også kaldes med et heltal som parameter (så venter den det angive antal msek).

For de fleste af ***Actor*** klassens metoder er det rimelig oplagt, hvordan de fungerer. Hvis I er i tvivl, kan I:

- kigge i ***Actor*** klassens dokumentation (ved at åbne editoren og vælge Documentation i øverste højre hjørne).
- kigge i ***Actor*** klassens implementaton (ved at åbne editoren og vælge Source Code i øverste højre hjørne).
- teste dem på lærredet (f.eks. ved at højreklikke et ***Turtle*** objekt og kalde metoden interaktivt – under "inherited from Actor").

Actor klassen er en såkaldt abstrakt kasse. Det betyder, at man ikke kan lave ***Actor*** objekter. I må ikke ændre koden i denne klasse.

Turtle klassen er den klasse, som I skal arbejde med, og hvor I skal skrive kode. I ovenstående klassediagram er der en pil med trekantet hoved fra ***Turtle*** klassen til ***Actor*** klassen. Pilen angiver, at ***Turtle*** kassen er en subklasse af ***Actor*** klassen, hvilket betyder, at alle ***public*** metoder i ***Actor*** klassen kan anvendes i ***Turtle*** klassen (fuldstændig som om de var defineret i ***Turtle*** klassen). Derimod har ***Turtle*** klassen ikke adgang til ***Actor*** klassens feltvariabler, idet disse er defineret til at være ***private***.

Turtle klassen har to konstruktører. I den ene kan I selv specificere, hvor på lærredet skildpadden placeres, mens den anden placerer skildpadden i punktet (100,100), hvilket er pasende for de fleste af de tegninger, som I skal lave.

Når I kalder en af konstruktørerne, dukker der et 600 x 600 lærred op på skærmen med en lille skildpadde placeret på det specificerede sted på lærredet, hvor punktet (0,0) er i øverste venstre hjørne.. Den sorte prik i midten af skildpadden indikerer, at pennen er nede og at farven er sort. Under lærredet er der to knapper. ***Clear*** knappen visker de ting ud, der er tegnet (uden at røre skildpadden), mens ***Grid*** knappen ændrer afstanden mellem de grå hjælpelinjer, der gør det lettere at aflæse koordinaterne.

Turtle klasen indeholder en række forskellige metoder:

- Demometoderne ***triangle*** og ***square*** illustrerer, hvordan man kan tegne trekanter og kvadrater ved hjælp af metoder fra ***Actor*** klassen. Start med at studere deres implementation.
- Fem test metoder, der gør det let for instruktorerne (og jer selv) at tjekke, om de metoder, som I har implementeret, fungerer korrekt.
- Alle øvrige metoder består af et metodehoved (som bl.a. angiver metodens returtype, navn og parametre) samt en tom krop. Det er jeres opgave at implementere disse metoder, dvs. udfylde kroppene på passende vis. Parametrenes navne samt den kommentar, der er knyttet til hver parameter, fortæller, hvad den pågældende parameter bruges til.

I dette projekt angives alle afstande og vinkler ved hjælp af typen ***double***, dvs. reelle tal. På den måde undgår vi for store afrundingsfejl. Alle de steder, hvor I skal bruge en ***double***, kan I i stedet bruge en ***int*** (dvs. et heltal). Hvis I vil præcisere, at tallet er en ***double***, skriver I et punktum og nogle decimaler bagefter, f.eks. 360.0 (som er af typen ***double***) i stedet for 360 (som er af typen ***int***).

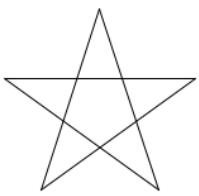
Opgave 1

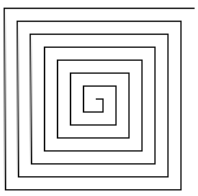
Implementer metoderne ***polygon*** og ***circle*** (som vist i en af forelæsningerne). Test at metoderne virker efter hensigten – også når skildpadden starter i en vinkel, der er forskellig fra 0. Når skildpadden er færdig med at tegne figuren, skal den være tilbage i udgangspositionen og have samme vinkel, som den have før start. Det er op til brugeren at sørge for, at pennen er nede før metoderne kaldes. Hvis dette ikke er tilfældet, gennemløber skildpadden stadig figuren, men efterlader sig intet spor.

Modificer demometoderne ***triangle*** og ***square***, således at de bruger ***polygon*** metoden i stedet for selv at have en for løkke.

Opgave 2

Implementer metoderne ***star*** og ***spiral***, således at de tegner nedenstående figurer. For sidstnævnte metode, kræves det ikke at skildpadden slutter i udgangspositionen, og det er nemmest at tegne figuren indefra og ud (altså starte i midten). Metoderne (og alle efterfølgende metoder) skal også fungere, når skildpadden starter i en vinkel der er forskellig fra 0.



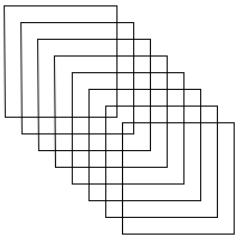


Opgave 3

Når I om lidt skal tegne mere komplicerede figurer, får I meget ofte brug for at skildpadden tager pennen op, positionerer sig et nyt sted, for derefter atter at sætte pennen ned på lærredet. I skal derfor nu implementere to hjælpemetoder, ***jumpTo*** og ***jump***, som kan gøre dette for jer. Den første metode hopper til den angivne position på lærredet. Den anden metode hopper relativt til skildpaddens nuværende position og vinkel. Den første parameter angiver, hvor meget skildpadden skal hoppe frem af, mens den anden angiver, hvor meget den skal hoppe til siden. For begge metoder gælder, at skildpaddens vinkel ikke ændres, og at der sluttes med et kald af ***penDown*** metoden, således at skildpadden er klar til at tegne videre.

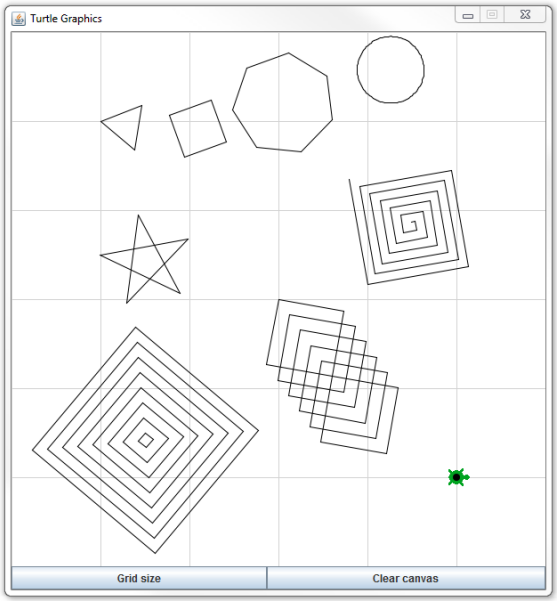
Opgave 4

Implementer metoden ***squares***, således at den tegner nedenstående figur. Det kræves ikke, at skildpadden slutter i udgangspositionen. Husk at metoden også skal fungere, når skildpadden starter i en vinkel der er forskellig fra 0. Metoden kalder ***penUp*** og ***penDown*** flere gange. Det er op til brugeren at sørge for, at pennen er nede før metoden kaldes. Hvis dette ikke er tilfældet, vil første del af figuren ikke blive vist.



Opgave 5

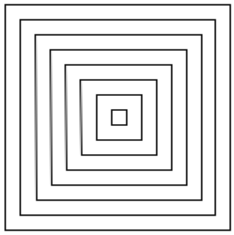
Skab en skildpadde og kald metoden ***testTurtle1*** (eller kald klassemetoden ***testFirst***). Metoden skal producere nedenstående lærred (hvor den nederste figur først bliver tegnet, når I har løst opgave 6). Hvis jeres lærred ser forkert ud, er der noget galt med jeres metoder, og I skal derfor rette dem til. Det er selvfølgelig forbudt at modificere testmetoden.



Dernæst skal I afprøve jeres program ved hjælp af testserveren. Kald klassemetoden ***test*** i ***TestServer*** klassen med parameteren ”Turtle1”. Hvis testserveren finder fejl, skal l gennemgå jeres kode og forsøge at rette dem.

Opgave 6 (til dem, der har mod på mere)

Implementer metoden ***squares2***, således at den tegner nedenstående figur. Det er nemmest at begynde indefra og så arbejde sig udad. Det kræves ikke, at skildpadden slutter i udgangs-positionen. Husk at metoden også skal fungere, når skildpadden starter i en vinkel der er forskellig fra 0. Metoden kalder ***penUp*** og ***penDown*** flere gange. Det er op til brugeren at sørge for, at pennen er nede før metoden kaldes. Hvis dette ikke er tilfældet, vil første del af figuren ikke blive vist.



Afprøv den skrevne metode ved hjælp af testserveren (som beskrevet i opgave 5).