Laboration 5 – implementera klasser, Turtle

Mål: Du ska fortsätta att träna på att implementera och använda klasser. Du ska också få mer träning i att använda Eclipse debugger.

Förberedelseuppgifter

- Tänk igenom följande frågor och se till att du kan svara på dem:
  1. Vad är en specifikation respektive en implementering av en klass?
  2. Vad är ett attribut? Var deklareras attributen? När reserveras det plats för dem i datorns minne och hur länge finns de kvar?
  3. När exekveras satserna i konstruktorn? Vad brukar utföras i konstruktorn?
  4. Vilka likheter/skillnader finns det mellan attribut, lokala variabler och parametrar.
  5. Vad menas med public och private?
- Läs avsnittet Bakgrund.

Bakgrund

Turtle graphics är en teknik för att rita linjer. Linjerna ritas av en (tänkt) sköldpadda som går omkring i ett ritfönster. Sköldpaddan har en penna som antingen kan vara sänkt (då ritas en linje i sköldpaddans spår) eller lyft (då ritas ingen linje). Sköldpaddan kan bara gå rakt framåt, i den riktning som huvudet pekar, men när den står stilla kan den vända sig i en ny riktning.

En klass `Turtle` som beskriver en sköldpadda av detta slag har följande specifikation:

```
/** Skapar en sköldpadda som ritar i ritfönstret w. Från början
    befinner sig sköldpaddan i punkten x,y med pennan lyft och
    huvudet pekande rakt uppåt i fönstret (i negativ y-riktning). */
Turtle(SimpleWindow w, int x, int y);

/** Sänker pennan. */
void penDown();

/** Lyfter pennan. */
void penUp();

/** Går rakt framåt n pixlar i den riktning som huvudet pekar. */
void forward(int n);

/** Vrider beta grader åt vänster runt pennan. */
void left(int beta);

/** Går till punkten newX,newY utan att rita. Pennans läge (sänkt
    eller lyft) och huvudets riktning påverkas inte. */
void jumpTo(int newX, int newY);

/** Återställer huvudriktningen till den ursprungliga. */
void turnNorth();

/** Tar reda på x-koordinaten för sköldpaddans aktuella position. */
int getX();

/** Tar reda på y-koordinaten för sköldpaddans aktuella position. */
int getY();

/** Tar reda på sköldpaddans riktning, i grader från positiv x-led. */
int getDirection();
```

Observera att vi här bestämmer vilket fönster som sköldpaddan ska rita i när vi skapar ett sköldpaddsobjekt. Det kan väl sägas motsvara verkligheten: en sköldpadda ”befinner” sig ju alltid någonstans.

I följande program ritar en sköldpadda en kvadrat med sidorna parallella med axlarna:

```
import se.lth.cs.pt.window.SimpleWindow;

public class TurtleDrawSquare {
    public static void main(String[] args) {
        SimpleWindow w = new SimpleWindow(600, 600, "TurtleDrawSquare");
        Turtle t = new Turtle(w, 300, 300);
        t.penDown();
        for (int i = 0; i < 4; i++) {
            t.forward(100);
            t.left(90);
        }
    }
}
```

I nedanstående variant av programmet har vi gjort två ändringar: 1) längden på sköldpaddans steg väljs slumpmässigt mellan 0 och 99 pixlar, 2) efter varje steg görs en paus på 100 millisekunder. Den första ändringen medför att figuren som ritas inte blir en kvadrat, den andra ändringen medför att man ser hur varje linje ritas.

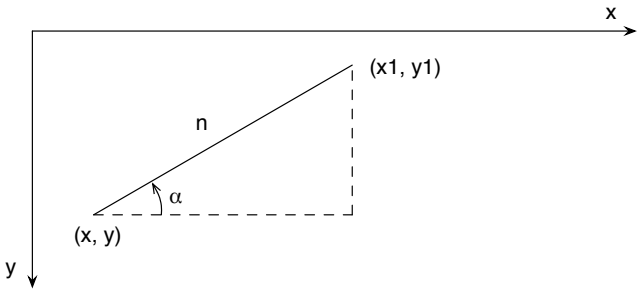
```
import se.lth.cs.pt.window.SimpleWindow;
import java.util.Random;

public class TurtleDrawRandomFigure {
    public static void main(String[] args) {
        Random rand = new Random();
        SimpleWindow w = new SimpleWindow(600, 600, "TurtleDrawRandomFigure");
        Turtle t = new Turtle(w, 300, 300);
        t.penDown();
        for (int i = 0; i < 4; i++) {
            t.forward(rand.nextInt(100));
            SimpleWindow.delay(100);
            t.left(90);
        }
    }
}
```

För att dra slumptal måste man skapa ett `Random`-objekt och att man får ett nytt slumpmässigt heltal i intervallet [0, n) med funktionen `nextInt(n)`. Skrivsättet [0, n) betyder att 0 ingår i intervallet, n ingår inte i intervallet. `rand.nextInt(100)` ger alltså ett slumpmässigt heltal mellan 0 och 99.

När klassen `Turtle` ska implementeras måste man bestämma vilka attribut som klassen ska ha. En sköldpadda måste hålla reda på:

- fönstret som den ska rita i: ett attribut `SimpleWindow w`,
- var i fönstret den befinner sig: en x-koordinat och en y-koordinat. För att minska inverkan av avrundningsfel i beräkningarna ska `x` och `y` vara av typ `double`,
- i vilken riktning huvudet pekar. Riktningen kommer alltid att ändras i hela grader. Man kan själv välja om attributet som anger riktningen ska vara i grader eller i radianer,
- om pennan är lyft eller sänkt. Ett sådant attribut bör ha typen `boolean`. booleanvariabler kan bara anta två värden: `true` eller `false`. Man testar om en booleanvariabel `isPenDown` har värdet `true` med `if (isPenDown) ...`.

När sköldpaddan ska gå rakt fram i den aktuella riktningen ska en linje ritas om pennan är sänkt. Den aktuella positionen ska också uppdateras. Antag att sköldpaddan i ett visst ögonblick är vriden vinkeln α i förhållande till positiv x-led. När metoden `forward(n)` utförs ska pennan flyttas från punkten (x, y) till en ny position, som vi kallar $(x1, y1)$:



Av figuren framgår att $(x1, y1)$ ska beräknas enligt:

$$\begin{aligned} x1 &= x + n\cos\alpha \\ y1 &= y - n\sin\alpha \end{aligned}$$

I Java utnyttjas standardfunktionerna `Math.cos(alpha)` och `Math.sin(alpha)` för att beräkna cosinus och sinus. Vinkeln `alpha` ska ges i radianer.

`x` och `y` är av typ `double`. När koordinaterna utnyttjas som parametrar till `SimpleWindow`-metoderna `moveTo` och `lineTo` och när de ska returneras som funktionsresultat i `getX` och `getY` måste de avrundas till heltalsvärden. Det kan till exempel se ut så här:

```
w.moveTo((int) Math.round(x), (int) Math.round(y));
```

Datorarbete

1. I filen *Turtle.java* i projektet *Lab05* finns ett ”skelett” (en klass utan attribut och med tomma metoder) till klassen `Turtle`.

   Implementera klassen `Turtle`: Skriv in attributen i klassen. Skriv gärna en kommentar till varje attribut som förklarar vad attributet betyder. Skriv också konstruktorn och se till att alla attribut får rätt startvärden. Implementera de övriga metoderna.

2. Klasserna `TurtleDrawSquare` och `TurtleDrawRandomFigure` finns i filerna *TurtleDrawSquare.java* och *TurtleDrawRandomFigure.java*. Kör programmen och kontrollera att din `Turtle`-implementation är korrekt.

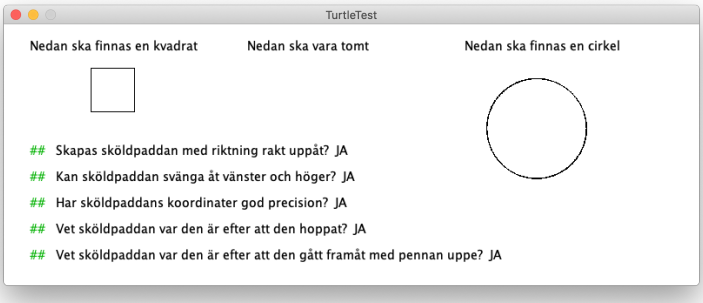
   Använd gärna debuggern för att hitta eventuella svårfunna fel. Här kan det vara bra att utnyttja kommandot Step Into. Det fungerar som Step Over med skillnaden att man följer exekveringen in i metoder som anropas.

3. Du ska nu träna mer på att använda debuggern i Eclipse. I fortsättningen förutsätter vi att du utnyttjar debuggern för att hitta fel i dina program.

   Välj ett av programmen från uppgift 2. Sätt en brytpunkt på en rad där en metod i Turtle anropas, t. ex. `t.penDown()` eller något liknande. Kör programmet i debuggern. Använd Step Into och följ hur man från `main`-metoden ”gör utflykter” in i metoderna i klassen `Turtle`.

   Lägg märke till vilka storheter (variabler, attribut, parametrar) som är tillgängliga när man ”är i” `main`-metoden resp. inuti någon metod i klassen `Turtle`. **Tips!** Om du klickar på trekanten framför this i variabelvyn visas `Turtle`-objektets attribut.

4. I projektet finns också programmet `TurtleTest`, som testar din `Turtle`-klass i olika avseenden. Ett antal misstag kan upptäckas på detta sätt. Resultatet ska se ut så här:



   Kör programmet `TurtleTest`. Om de ritade figurerna är felaktiga, eller något av testerna besvaras med ”NEJ”, gå tillbaka till din `Turtle`-klass och åtgärda felet.

5. Skriv ett program där en sköldpadda tar 1000 steg i ett fönster. Sköldpaddan ska börja sin vandring mitt i fönstret. I varje steg ska steglängden väljas slumpmässigt i intervallet [1, 10]. Efter varje steg ska sköldpaddan vridas ett slumpmässigt antal grader i intervallet $[-180, 180]$.

6. Skriv ett program där *två* sköldpaddor vandrar över ritfönstret. Steglängden och vridningsvinkeln ska väljas slumpmässigt i samma intervall som i föregående uppgift. Vandringen ska avslutas när avståndet mellan de båda sköldpaddorna är mindre än 50 pixlar. Sköldpaddorna ska turas om att ta steg på följande sätt:

```
while ("avståndet mellan sköldpaddorna" >= 50) {
    "låt den ena sköldpaddan ta ett slumpmässigt steg och göra
     en slumpmässig vridning"
    "låt den andra sköldpaddan ta ett slumpmässigt steg och göra
     en slumpmässig vridning"
    SimpleWindow.delay(10);
}
```

   Låt den ena sköldpaddan börja sin vandring i punkten (250,250) och den andra i (350,350).

7. Betrakta klassen `Turtle` och de program du skrev i uppgift 5–6 ovan. Ge exempel på ett attribut, en parameter och en lokal variabel. Ange också var i klassen/programmet respektive storhet är tillgänglig och får användas.

Exempel på	*Namn*	*Får användas var*
Attribut		
Parameter		
Lokal variabel		

Checklista

I den här laborationen har du övat på

- skillnaden mellan attribut, parametrar och lokala variabler,
- att använda konstruktorer för att sätta attributens startvärden,
- att använda klassen Math för beräkningar, och
- att skapa och använda en egen klass.