Laboration 2 – använda färdigskrivna klasser, kvadrat

Mål: Du ska lära dig att läsa specifikationer av klasser och att utnyttja färdigskrivna klasser för att lösa enkla uppgifter. Du ska också lära dig lite mera om Eclipse.

Förberedelser

- Läs avsnittet Bakgrund.
- Tänk igenom följande frågor och se till att du kan svara på dem:
  1. Vad är en main-metod? Hur ser en klass med en main-metoden ut i stora drag?
  2. Vad är referensvariabler och vad har man dem till?
  3. Hur skapar man objekt?
  4. Hur anropar man en metod på ett objekt?
  5. Vad använder man parametrar till?

Bakgrund

En klass `Square` som beskriver kvadrater har nedanstående specifikation.

```
/** Skapar en kvadrat med övre, vänstra hörnet i x,y
    och med sidlängden side. */
Square(int x, int y, int side);

/** Ritar kvadraten i fönstret w. */
void draw(SimpleWindow w);

/** Raderar bilden av kvadraten i fönstret w. */
void erase(SimpleWindow w);

/** Flyttar kvadraten avståndet dx i x-led, dy i y-led. */
void move(int dx, int dy);

/** Tar reda på x-koordinaten för kvadratens läge. */
int getX();

/** Tar reda på y-koordinaten för kvadratens läge. */
int getY();

/** Tar reda på kvadratens area. */
int getArea();
```

Användning av klassen Square

I nedanstående program skapas först ett ritfönster. Därefter skapas en kvadrat som placeras mitt i fönstret och ritas upp.

```
1  import se.lth.cs.pt.window.SimpleWindow;
2  import se.lth.cs.pt.square.Square;
3
4  public class DrawSquare {
5      public static void main(String[] args) {
6          SimpleWindow w = new SimpleWindow(600, 600, "DrawSquare");
7          Square sq = new Square(250, 250, 100);
8          sq.draw(w);
```

```
9       }
10  }
```

- På rad 1 och rad 2 importeras de klasser som behövs, i detta fall: klassen `SimpleWindow` och klassen `Square`. De klasserna finns inte i Javas standardbibliotek utan har skrivits speciellt för kurserna i programmeringsteknik. Klasserna finns i "paket" vars namn börjar med `se.lth.cs`; det betyder att de är utvecklade vid institutionen för datavetenskap vid LTH, som använder domännamnet `cs.lth.se`.
- På rad 6 skapas ett `SimpleWindow`-objekt. Refererensvariabeln `w` refererar till detta objekt.
- Därefter skapas ett `Square`-objekt som referensvariabeln `sq` refererar till.
- Slutligen ritas kvadraten `sq`.

Notera parametrarna som man använder när man skapar objekten. I `new`-uttrycket som skapar kvadratobjektet står det till exempel (250, 250, 100). Det betyder att kvadraten ska ha läget 250, 250 och sidlängden 100. Läget och sidlängden kan man ändra senare i programmet, med `sq.move(dx,dy)` och `sq.setSide(newSide)`.

Lägg också märke till att referensvariablerna `w` och `sq` har olika typer. En referensvariabels typ avgör vilka slags objekt variabeln får referera till. `w` får referera till `SimpleWindow`-objekt och `sq` får referera till `Square`-objekt.

Användning av klassen SimpleWindow

Bläddra fram källkoden till klassen `Square`. Du hittar klassen i Eclipse-projektet *cs_pt* under katalogen *src* och i paketet `se.lth.cs.pt.square`. Titta på källkoden och försök förstå vad som händer. Några av metoderna i `SimpleWindow` (metoderna för att flytta pennan och för att rita linjer) utnyttjas i metoden `draw` i klassen `Square`.

`SimpleWindow` innehåller också metoder, som inte används i `Square`, t.ex. för att ta hand om musklick. Dessa metoder har följande beskrivning:

```
/** Väntar tills användaren har klickat på en musknapp. */
void waitForMouseClick();

/** Tar reda på x-koordinaten för musens position vid senaste musklick. */
int getMouseX();

/** Tar reda på y-koordinaten för musens position vid senaste musklick. */
int getMouseY();
```

När exekveringen av ett program kommer fram till `waitForMouseClick` så "stannar" programmet och fortsätter inte förrän man klickat med musen någonstans i programmets fönster. Ett program där man skapar ett fönster och skriver ut koordinaterna för varje punkt som användaren klickar på har följande utseende:

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

public class PrintClicks {
    public static void main(String[] args) {
        SimpleWindow w = new SimpleWindow(600, 600, "PrintClicks");
        while (true) {
            w.waitForMouseClick();
            w.moveTo(w.getMouseX(), w.getMouseY());
            w.writeText("x = " + w.getMouseX() + ", " + "y = " + w.getMouseY());
        }
    }
}
```

`while (true)` betyder att repetitionen ska fortsätta ”i oändlighet”. Man avbryter programmet genom att välja Quit i File-menyn i `SimpleWindow`-fönstret.

Också i nedanstående program ska användaren klicka på olika ställen i fönstret. Nu är det inte koordinaterna för punkten som användaren klickar på som skrivs ut, utan i stället avståndet mellan punkten och den förra punkten som användaren klickade på. Vi sparar hela tiden koordinaterna för den förra punkten (variablerna `oldX` och `oldY`).

Gå igenom ett exempel ”för hand” och övertyga dig om att du förstår hur programmet fungerar.

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

public class PrintClickDists {
    public static void main(String[] args) {
        SimpleWindow w = new SimpleWindow(600, 600, "PrintClickDists");
        int oldX = 0; // x-koordinaten för "förra punkten"
        int oldY = 0; // y-koordinaten
        while (true) {
            w.waitForMouseClick();
            int x = w.getMouseX();
            int y = w.getMouseY();
            w.moveTo(x, y);
            int xDist = x - oldX;
            int yDist = y - oldY;
            w.writeText("Avstånd: " + Math.sqrt(xDist * xDist + yDist * yDist));
            oldX = x;
            oldY = y;
        }
    }
}
```

Datorarbete

1. Logga in på datorn, starta Eclipse, öppna projektet *Lab02*. (Logga in, starta Eclipse och öppna ett projekt ska du alltid göra, så det skriver vi inte ut i fortsättningen.)

2. Öppna en webbläsare och gå till kursens hemsida, `cs.lth.se/edaa55-ht2` . Under Dokumentation finns en länk till dokumentationen av de färdigskrivna klasser som används under laborationerna, alltså klasserna i paketet `se.lth.cs.pt`. Leta upp och titta igenom specifikationen av klassen `Square`. Det är samma specifikation som finns under Bakgrund, fast i något annorlunda form.

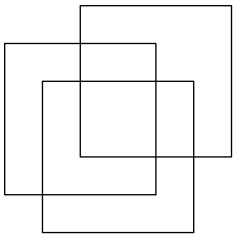
3. Klassen `DrawSquare` finns i filen *DrawSquare.java*. Öppna filen, kör programmet.

4. Kopiera filen *DrawSquare.java* till en ny fil med namnet *DrawThreeSquares.java*. Enklast är att göra så här:

   1. Markera filen i projektvyn, högerklicka, välj Copy.
   2. Högerklicka på *(default package)*, välj Paste, skriv det nya namnet på filen.

   Notera att Eclipse ändrar klassnamnet i den nya filen till `DrawThreeSquares`.

   Ändra sedan klassen så att kvadraten ritas tre gånger. Mellan uppritningarna ska kvadraten flyttas. Du ska fortfarande bara skapa ett kvadratobjekt i programmet. Resultatet ska bli en figur med ungefär följande utseende:

   Testa programmet, rätta eventuella fel.

5. Någonstans i ditt program skapas ett kvadratobjekt. Det görs i en sats som har ungefär följande utseende: `Square sq = new Square(300,300,200)`. Ta bort denna sats från programmet (eller kommentera bort den). Eclipse kommer att markera flera fel i programmet, eftersom `sq` inte är deklarerad och du senare i programmet utnyttjar den variabeln. Läs och tolka felmeddelandena.

   Lägg sedan in satsen `Square sq = null` i början av programmet. Eclipse kommer inte att hitta några fel, eftersom programmet nu följer de formella reglerna för Javaprogram. Exekvera sedan programmet och se vad som inträffar. Studera felmeddelandet så att du kan tolka det.

   Meddelanden om exekveringsfel skrivs i konsolfönstret. Det skrivs också ut en ”stack trace” för felet: var felet inträffade, vilken metod som anropade metoden där det blev fel, vilken metod som anropade den metoden, osv. Om man klickar på ett radnummer öppnas rätt fil i editorn med den raden markerad (under förutsättning att programtexten, ”källkoden”, för den filen är tillgänglig).

   Lägg sedan tillbaka den ursprungliga satsen för att skapa kvadratobjektet. Ändra argumentet `w` i det första anropet av `draw` till `null`. Kör programmet, studera det felmeddelande som du får.

   När man får exekveringsfel kan det vara svårt att hitta orsaken till felet. Här är en debugger till stor hjälp, i och med att man kan sätta brytpunkter och köra programmet stegvis.

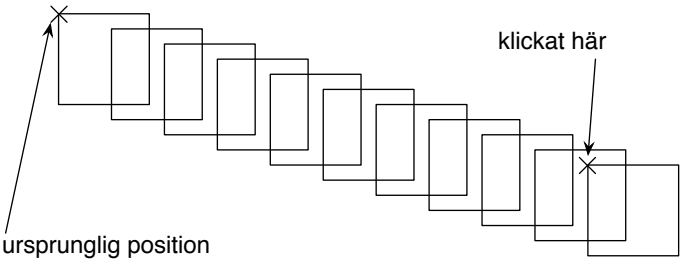
6. Kör programmen `PrintClicks` och `PrintClickDists`.

7. Skriv ett program där ett kvadratobjekt skapas och ritas upp i ett ritfönster. När användaren klickar med musen i fönstret ska bilden av kvadraten raderas, kvadraten flyttas till markörens position och ritas upp på nytt. Titta på dokumentationen av SimpleWindow och Square som finns länkad från kurshemsidan. Vilka metoder som verkar användbara för uppgiften kan du hitta där?

   Välj ett lämpligt namn på klassen. För att skapa en fil för klassen kan du antingen skapa en tom klass (klicka på New Java Class-ikonen, skriv namnet på klassen, klicka på Finish) eller kopiera någon av de filer som du redan har.

   Testa programmet.

8. Modifiera programmet i uppgift 7 så att varje flyttning går till så att kvadraten flyttas stegvis till den nya positionen. Efter varje steg ska kvadraten ritas upp (utan att den gamla bilden raderas). Exempel på kvadratbilder som ritas upp när flyttningen görs i 10 steg:



Kvadratens slutliga position behöver inte bli exakt den position som man klickat på. Om man till exempel ska flytta kvadraten 94 pixlar i 10 steg är det acceptabelt att ta 10 steg med längden 9.

Skriv in och testkör programmet.

9. Observera att programmet från uppgift 8 ritar många bilder av samma kvadrat och att alla bilderna kommer att synas i fönstret när programmet är slut. Om man raderar den ”gamla” bilden innan man ritar en ny bild så kommer man att få en ”rörlig”, ”animerad”, bild. För att man ska se vad som händer måste man då göra en paus mellan uppritning och radering — det gör man med `SimpleWindow`-metoden `delay`, som har en parameter som anger hur många millisekunder man ska vänta. Exempel:

```
while (sq.getSide() > 0) {
    sq.draw(w);
    SimpleWindow.delay(10);
    sq.erase(w);
    sq.setSide(sq.getSide() - 10);
}
```

Kodsnutten ovan animerar en ny mindre kvadrat tills dess att kvadratens sida blivit mindre än eller lika med noll.

Kopiera koden från uppgift 8 till en ny fil `AnimatedSquare.java`. Lägg in fördröjning och radering så att kvadratbilden blir ”animerad”. (Använd *inte* `SimpleWindow`-metoden `clear`.)

Anmärkning: i ett ”riktigt” program som visar rörliga bilder åstadkommer man inte animeringen på det här sättet. Denna lösning har bristen att man inte kan göra något annat under tiden som animeringen pågår, till exempel kan programmet inte reagera på att man klickar med musen.

10. I denna uppgift ska du lära dig fler kommandon i Eclipse. Det finns ett otal kommandon, mer eller mindre avancerade, och många av dem använder man sällan. Två kommandon som man ofta använder:

- Source-menyn > Format. Korrigerar programlayouten i hela filen. Ser till exempel till att varje sats skrivs på en rad, att det är blanka runt om operatorer, och så vidare. Man bör formatera sina program regelbundet, så att de blir läsbara. Observera att programmet ska vara korrekt formaterat när du visar det för labhandledaren för att få det godkänt. Notera också kortkommandot som står intill menyalternativet. Prova att använda kortkommandot också, och lär dig gärna det, då det är snabbare och mer bekvämt än att klicka.

- Refactor-menyn > Rename. Ändrar namn på den variabel eller metod som markerats (lokalt om det är en lokal variabel, i hela klassen om det är ett attribut, även i andra klasser om det är en publik metod).

11. Kontrollera med hjälp av checklistan nedan att du behärskar de olika momenten i laborationen. Diskutera med övningsledaren om någonting är oklart.

Checklista

Exempel på vad du ska kunna efter laborationen:

- Tyda specifikationer av klasser.
- Skriva program som använder färdiga klasser:
  - Deklarera referensvariabler och skapa objekt.
  - Anropa metoder på objekt.