# 18.0 Appletter og grafik

I foregående lektion blev du introduceret til appletter. Små tegninger, der enten var fuldstændig fastlagt fra begyndelsen eller blev tegnet vha. loops og betingelser. Et par af dem indeholdt en tilfældighedsgenerator (i form af Math.random()) så appletten så anderledes ud for hver gang, den blev kørt.  
 Som du sikkert har bemærket, så fylder sådan grafik faktisk rigtig mange programlinier. Det er grunden til at vi hidtil har lavet alle vores programmer som *applikationer* (selvstændigt program med en *main*-metode) og har kørt dem i Eclipses ”Message”-vindue. Det, et program faktisk udfører, og de forskellige valgstrukturer, drukner ofte i grafik, hvis man skriver det hele sammen i et langt program.

Ekstra… det med småt…:  
Det problem der påpeges ovenfor, er et meget generelt problem inden for programmering.  
Løsningen på dette problem er, at man lave sine ”hardcore” programmer, - altså dem der laver de interessante kalkulationer – i nogle klasser for sig selv. Al grafik, og dermed brugerflade, laves i helt andre klasser. Sådanne grafiske brugerflader hedder Graphic User Interface, forkortet GUI. Fordelen ved at holde brugerfladen ude fra selve programmet er blandt andet at GUI’en lettere kan moderniseres og se smarte ud, uden at det grundlæggende behøver at ændres. Der findes mange editorer til at hjælpe med at lave GUI. Borlands (nu Codegear) Enterprise versioner har en, hvor rammer, knapper, skydere o.m.a. ”blot” kan trækkes ind på brugerfladen, frem for at skrive alle programlinier selv.  
Det ligger udenfor dette fags (Programmering C ) pensum og tidsramme at gå i dybden med førnævnte designproblem som at adskille beregningslaget og præsentationslaget (GUI’erne). Bl.a. på Ingeniørhøjskolen Odense Teknikum er der særskilte fag der tager sig af den slags: Objektorienteret Analyse og Design.   
Her fortsætter lektionen med at gøre grafikken interaktiv, der er ingen store komplicerede kalkulationer.

## 18.1 ”Interaktive” appletter

Hidtil har vore appletter været temmelig døde, når de først var tegnet. Hvis brugeren skal deltage med mus eller tastatur vil det kræve at appletten kan få besked fra mus og tastaturet, når brugeren foretager sig et og andet på appletten Til den opgave bruger vi *Listeners* (”Lyttere”) en gruppe klasser, der populært sagt lytter til styresystemet og opfanger aktiviteter fra hhv. mus og tastatur.  
 *Listeners* er en særlig slags klasse nemlig *Interfaces.*  De fleste interfaces har *abstrakte metoder* hvilket er en metode uden indhold:

public void mouseClicked(MouseEvent event) { /\* tom! \*/ }

Tricket ved en abstrakt metoder er at denne metode opsnapper, når nogen klikker med musen. Hvad der så derefter skal ske, er helt op til dig som programmør. Hvis du blot lader den forblive tom, ja så sker der intet ved at klikke med musen. Men andre har lavet dette interface og klaret al kom­mu­ni­ka­tion med operativ systemet. Vi kan frit bruge løs af disse *interface-*klasser, når blot vi overholder at skrive alle interfacet’s metoder op.

***Muse-aktiviteter :***   
I Java opdeles aktiviteter med musen i 2 kategorier: ”mouse events” (hændelse) og ”mouse motion” (bevægelse).  
1. Musehændelserne er:  
*mouse pressed* – når muse-knappen trykkes ned.  
*mouse released* – når den slippes  
*mouse clicked* – nedtryk og slip uden at flytte musen der imellem.  
*mouse entered* – når muse-pointeren bevæges over en bestemt komponent.  
*mouse exited* – når muse-pointeren bevæges væk fra/forlader en bestemt komponent.

2. Musebevægelserne er:  
*mouse moved* – når musen bevæges.  
*mouse dragged* – når musen bevæges mens knappen er trykket ned.  
  
Lad os se et eksempel der er bygget af to klasser; selve appletten *Dots* og en listener *DotsMouseListener.* Appletten tegner en plet, der hvor musen klikkes[[1]](#footnote-1) på den sorte flade.

Først Dots-appletten:

1 import java.applet.Applet;

2 import java.awt.\*;

3

4 public class Dots extends Applet

5 {

6 private final int APPLET\_WIDTH = 200;

7 private final int APPLET\_HEIGHT = 100;

8 private final int RADIUS = 6;

9

10 private Point clickPoint = null;

11 //------------------------------------------------------------------

12 public void init() /\*skaber en Listener for muse-hændelser

13 til denne applet \*/

14 {

15 DotsMouseListener listener = new DotsMouseListener(this);

16 addMouseListener(listener);

17

18 setBackground (Color.black);

19 setSize (APPLET\_WIDTH, APPLET\_HEIGHT);

20 }

21

22 //-------------------------------------------------------------------

23 public void paint (Graphics page) /\* tegner en plet på

24 et bestemt sted \*/

25 {

26 page.setColor (Color.green);

27 if (clickPoint != null)

28 page.fillOval (clickPoint.x - RADIUS, clickPoint.y - RADIUS,

29 RADIUS \* 2, RADIUS \* 2);

30 }

31

32 //-----------------------------------------------------------------

33 public void setPoint (Point point) /\* Ændrer punkt til der

34 hvor næste plet skal tegnes \*/

35 {

36 clickPoint = point;

37 }

38 }

Her kommer Listeneren:

1 import java.applet.Applet;

2 import java.awt.\*;

3 import java.awt.event.\*;

4

5

6 class DotsMouseListener implements MouseListener

7 {

8 private Dots applet;

9 //-----------------------------------------------------------------

10

11 public DotsMouseListener (Dots applet) /\*gemmer en reference til 12 appletten\*/

13 {

14 this.applet = applet;

15 }

16 //-----------------------------------------------------------------

17

18 public void mouseClicked (MouseEvent event) /\* opfanger punkt hvor

19 musen blev klikket, sætter punktet i appletten og lader den gentegne \*/

20 {

21 Point clickPoint = event.getPoint();

22 applet.setPoint (clickPoint);

23 applet.repaint();

24 }

25

26 //-----------------------------------------------------------------

27 // Herunder tomme metoder som ikke benyttes. Skal med aht interface!

28 //-----------------------------------------------------------------

29 public void mousePressed (MouseEvent event) {}

30 public void mouseReleased (MouseEvent event) {}

31 public void mouseEntered (MouseEvent event) {}

32 public void mouseExited (MouseEvent event) {}

33 }

### Opgave 18.1

1. Kør appletten ved enten at trykke ”kør” i Eclipe (grøn afspilningsknap) eller start Dots.html fra stifinder eller browser. Klik med musen på den sorte applet. Hold musen helt stille imens.
2. Udskriv denne lektion på printer, så du kan følge nedenstående forklaringer til klasserne.  
   Gennemgå selv klasserne linie for linie.

**Forklaring til Dots:**   
Bemærk at appletten har to metoder: *init()* og *paint(…).*  
*init()* - linie 12 - kaldes automatisk, når appletten aktiveres første gang. *paint()* køres umiddelbart efter *init(). paint()* køres også når *repaint()* kaldes.

Appletten, *Dots,* starter med at skabe et objekt af typen *DotsMouseListener* og sender sig selv (*this*) med. Dvs. den sender adressen til hvor den selv(*Dots*) ligger i hukommelsesstakken. Det er for at *DotsMouselisteneren* skal kunne sende besked tilbage til vores *Dots*-applet.

15 DotsMouseListener listener = new DotsMouseListener(this);

Derefter kaldes en applet-metode *addMouseListener(…):*

16 addMouseListener(listener);

Den metode står overhovedet ikke i *Dots*! Men husk at alle vores appletter indtil nu har arvet fra den oprindelige urmoder Applet ved , - sammen med klassenavnet - , at skrive *extends Applet*.

4 public class Dots extends Applet

Det giver os nedarvet ret til at bruge metoder som står i den oprindelige Applet-klasse. Det metoden *addMouseListener(…)* gør, er, at den sørger for at vores Muse-lytter bliver registreret i operativsystemet. Først nu er vi i stand til at opsnappe, når nogen piller ved musen.  
Herefter er det almindeligt layout og grafik der foregår i resten af appletten *Dots*. Men bemærk lige linie 10 hvor vi tager et objekt af typen *Point*.   
Ekstra opgave: Gæt hvad Point mon er og undersøg den. Du kan få vist metoder i denne klasse ved inde i Eclipse editor ( og inde i Dots-klassen) at skrive clickpoint.\_ (blanktegn efter punktum).

**Forklaring til DotsMouseListener:**

I linie 6 lover vi at implementere metoderne fra Interfacet *MouseListener*.

Linie 11-15 er en simpel kontruktør, hvor adressen til den medsendte applet gemmes i en privat applet.  
Linie 18-24 er een af de 5 obligatoriske muse-hændelses-metoder. Bemærk i linie 18 at metoden får et argument MouseEvent med!

18 public void mouseClicked (MouseEvent event)

Igen har der været nogle geniale programmører før os, der har sørget for det grove arbejde: Når operativsystemet registrerer musehændelser, får vores Listener besked og får tilmed et helt *Event*-objekt med i købet. *MouseEvent* har egne metoder, såsom *getPoint()*, *getX(),* *getY(),* *getClickCount()* og andre.  
Herefter sættes punktet til det leverede fra operativsystemet og appletten gentegnes i linie 23. Bemærk at vi ikke kan finde metoden *repaint()* i Dots’ kildetekst: Det er en nedarvet metode fra ur-moderen Applet, som vi frit kan bruge.

Linie 29-32 er de øvrige 4 obligatoriske musehændelses-metoder. Dem benytter vi ikke i dette program, så de står blot tomme { }. Men vi lovede jo at have disse metoder i linie 6, så her er de.

## Elastik-applet: RubberLines

Lad os kigge på endnu en applet. Den er på sin vis simplere og elegantere hvis man først er med på ideen med interfaces og listeners: I én og samme klasse har vi både appletten og ikke mindre end to listeners nemlig både en muse-hændelses-lytter og en muse-bevægelses-lytter.

import java.applet.Applet;

import java.awt.\*;

import java.awt.event.\*;

public class RubberLines extends Applet implements MouseListener,

MouseMotionListener

{

private final int APPLET\_WIDTH = 200;

private final int APPLET\_HEIGHT = 200;

private Point point1 = null;

private Point point2 = null;

//-----------------------------------------------------------------

// Adds this class as a listener for all mouse related events.

//-----------------------------------------------------------------

public void init()

{

addMouseListener (this);

addMouseMotionListener (this);

setBackground (Color.black);

setSize (APPLET\_WIDTH, APPLET\_HEIGHT);

}

//-----------------------------------------------------------------

// Draws the current line from the initial mouse down point to

// the current position of the mouse.

//-----------------------------------------------------------------

public void paint (Graphics page)

{

page.setColor (Color.green);

if (point1 != null && point2 != null)

page.drawLine (point1.x, point1.y, point2.x, point2.y);

}

//-----------------------------------------------------------------

// Captures the position at which the mouse is initially pushed.

//-----------------------------------------------------------------

public void mousePressed (MouseEvent event)

{

point1 = event.getPoint();

}

//-----------------------------------------------------------------

// Gets the current position of the mouse as it is dragged and

// draws the line to create the rubberband effect.

//-----------------------------------------------------------------

public void mouseDragged (MouseEvent event)

{

point2 = event.getPoint();

repaint();

}

//-----------------------------------------------------------------

// Provide empty definitions for unused event methods.

//-----------------------------------------------------------------

public void mouseClicked (MouseEvent event) {}

public void mouseReleased (MouseEvent event) {}

public void mouseEntered (MouseEvent event) {}

public void mouseExited (MouseEvent event) {}

public void mouseMoved (MouseEvent event) {}

}

Se selv programmet igennem. Hvis der er spørgsmål, besvares de i lektio­ner­ne.

I programpakken (zip-fil) er der endnu to appletter: Directions og Rebound. Directions reagerer på tastaturet, prøv pil-tasterne og prøv også de andre (tænd for evt.højtalere!). Rebound starter og stopper en animation hvis der klikkes i appletfeltet.  
Bemærk endnu en variation: At listenerne i disse to appletter ligger i en privat klasse inde i Applet-klassen.

En listener kan altså   
1) ligge som en separat klasse, - som vi så med Dots og DotsMouseListener.  
2) Listenerne kan være helt integreret inde i appletklassen, som vi så med RubberLines.   
3) Listenerne kan ligge i indre klasser som i Directions og Rebound.

### Opgave 18.2

Prøv appletterne og at ændre lidt på farver og plet-størrelse. Kun din fantasi og tid sætter grænser.

**Opgave 18.3** (Ekstra):  
Lav en applet der tæller det antal gange museknappen er blevet klikket. Vis antallet midt i appletten.

**Opgave 18.4** (Ekstra):  
Lav en applet der tegner en cirkel med elastik effekt. Cirklen størrelse bestemmes ved at beregne hvor langt musen blev trukket. Afstandformel √((x2-x1)2+(y2-y1)2)

1. Pas på at holde musen helt stille, der hvor der klikkes. [↑](#footnote-ref-1)