# ex4javafx

Joachim von Hacht

# Detta är Överkurs



Kommer inte på tentan. Allt som behövs för labbar är färdigt!

- Detta är för den intresserade.

# Datorgrafik och JavaFX




Datorgrafik

- Används för att rita bilder på en datorskärm
- Varje enskild bildpunkt (pixel) måste kunna adresseras (komma åt/ändra)
- Grafik innebär att (delar av) datorskärmen ritas om med en viss frekvens (ca 60 ggr/sek. i vårt fall)
- Man brukar skilja på 2D- och 3D-grafik, bilden visar en variant av 2D-grafik.

Vi kommer att använda **JavaFX** för att skriva grafiska 2D program.

- JavaFX är numera att separat bibliotek. För att det skall fungera måste vi ladda ned och lägga till JavaFX i IntelliJ.

Se upp vid import: Det finns flera äldre grafikbiblioteket, AWT och Swing. Vissa klasser i AWT heter samma som i JavaFX. Vid konstiga fel kontrollera import. Skall (troligen) aldrig stå import java.awt….eller java.swing

# Programstruktur i JavaFX

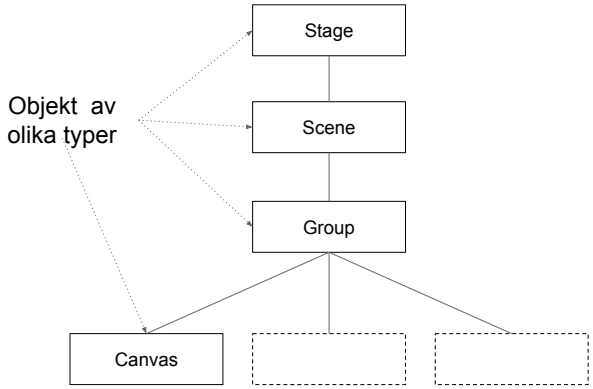
```
public class MyGraphicalProgram extends Application {

    @Override
    public void init(){
      // Initialize program data
    }

    @Override
    public void start(Stage primaryStage) throws Exception {
        // Set up graphics
    }


    public static void main(String[] args) {
        launch(args);
    }

}
```

4

Ett JavaFX program måste ange extends Application

- Innan grafiken startas anropas automatiskt methoden init()
  - Kan används för att initialisera applikationen
- Efter detta anropas metoden start(), där grafiken initialiseras och startas
  - Som parameter skickar JavaFX automatiskt ett Stage objekt, se nästa bild.
  - När metoden start är färdig kommer det att visas ett fönster på skärmen (vi måste kode lite för att det skall hända, mer strax)
- @Override betyder att vi gjort egna versioner av metoderna, mer senare.

# JavaFX Scengraf




5

JavaFX använder en Scengraf. Scengrafen är en datastruktur uppbyggd av objekt av olika typer

- Stage-objekt motsvarar ett fönster, skapas automatiskt av JavaFX och skickas som argument till start-metoden
- Scen, ett objekt som beskriver en scen i fönstret (man kan byta scener i ett fönster)
    - Skapas i programmet av oss.
    - Innehåller olika objekt (vissa synliga på skärmen, andra inte)
- Group är en grupp av olika objekt
- Canvas är ett enskilt synligt objekt. Fungerar som en rityta för grafik.
- Finns många andra objekt.
- Allt hanteras som referenser.

# Start metoden

```
public class MyGraphicalProgram extends Application {
    GraphicsContext gc;   // Accessible to render methods
    @Override
    public void start(Stage primaryStage) throws Exception {
        Pane root = new Pane();
        Canvas canvas = new Canvas(width, height);
        gc = canvas.getGraphicsContext2D();
        root.getChildren().addAll(canvas);

        Scene scene = new Scene(root);
        primaryStage.setScene(scene);
        primaryStage.setTitle("My Graphic Program");
        primaryStage.show();  // Window visible on screen
}
```

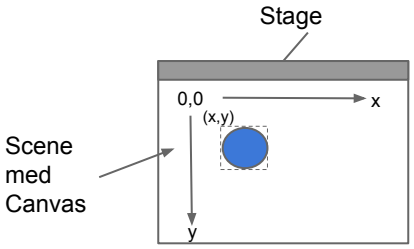


Analys av koden i start-metoden:

1. Vi skapar ett “panel-objekt”
2. ..och en rityta (Canvas)
3. Frågar ritytan efter en grafikkontext. Mer strax ...
4. Lägger till ritytan ovanpå panelen.
5. Skapar scenen.
6. Lägger till panelen till scenen
7. Lägger till scenen till fönstret.
8. Sätter titel på fönstret.
9. … och slutligen ritar ut allt på skärmen

# Lågnivå-rendering

```
// Using graphics context to draw
void renderBall(GraphicsContext gc) {
  ...
  gc.setFill(Color.BLUE);
  gc.fillOval(x, y, 10, 10);
  ...
}
```





7

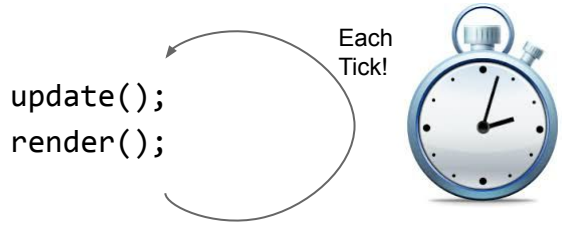
Utritning kallas **rendering**

- Rendering av objekt i vårt universum sker i ett grafiskt 2D universum (**skärmkoordinater**) med (0,0) i övre vänstra hörnet.

Allt vi behöver för att rita finns i ett GraphicsContext objekt

- Via Canvas-objektet kan vi få tillgång till ett GraphicsContext objekt
- Genom att anropa metoder på objektet kommer JavaFX att se till att något ritas (på motsvarande canvas)
- Metoderna vi anropar kallas **grafikprimitiver** (finns liknande i många språk)
    - Kan rita linjer, rektanglar, cirklar o.s.v.
    - För figurer utgår grafikprimitiverna från övre vänstra hörnet (x,y), figuren hamnar alltså snett nedanför (x,y)
    - Kan även rita ut bilder.

# Animeringsloop




8

För att animera ett förlopp gör vi följande:

1. Uppdatera världen rent logiskt, flytta saker logiskt, nya positioner, objekt kommer och går (update)
2. Rita ut en bild av världens nya tillstånd (render)
    - Som tidigare: Skilj på logik och IO (grafik i detta fall).

Animeringen drivs av en Timer, d.v.s ett objekt som kan upprepa något med en viss periodicitet (vi behöver ingen loop).

Finns mycket mer om detta i JavaFX men inget vi behöver.

# En JavaFX Timer

```
AnimationTimer timer = new AnimationTimer() {
   // Called by JavaFX approx. 60 times / sec.
    // now is the current time, supplied automagically
    public void handle(long now) {
                update(now); // Update logic
                render();   // Using GraphicsContext
   }
};

timer.start();     // Start timer
```

9

En JavaFX AnimationTimer är ett objekt med en fördefinierad metod som kommer att anropas av JavaFX med en viss periodicitet

- Lite komplicerat att skapa en AnimationTimer, vi går inte in på detaljerna
    - timer variabeln håller en referens till ett AnimationTimer-objekt
- Måste stå public framför metoden handle.
- Metoden handle() anropas automatiskt av JavaFX före varje rendering (d.v.s. ca. 60 ggr/sek)
- Metoden får som parameter aktuell systemtid (i nanosekunder).

# Uppdateringsfrekvens

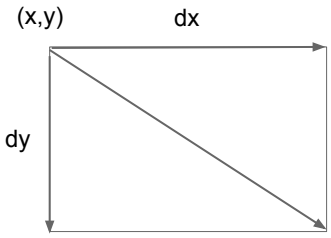
```
long lastUpdateTime;

// Called by a timer
void update(long now) {
   if (now - lastUpdateTime > 200_000_000) {
       x = x + 2;            // Do something
       lastUpdateTime = now;
   }
}
```



Parametern now kan användas för att styra hur ofta vi uppdaterar datan i programmet.

# Rörelse

```
class SpaceShip {
   double x;
   double y;
   double dx;
   double dy;


  void move() {
     this.x += dx;
     this.y += dy;
  }

}
```




Datorgrafik utgår alltid från att y-axeln pekar nedåt.



Objektet som skall röra sig i en 2d värld behöver instansvariabler för

- Position, x och y.
- Hastighet, dx, dy
  - Om dx = dy = 0 så står objektet stilla
- För att flytta objektet finns en metod move() som uppdaterar x och y utifrån hastigheten.

Kan även använda en normaliserad vektor + en hastighet.

- Ofta används en maxhastighet som inte får överskridas.