

# JavaFX och Händelsestyrda program

DIT012/Joachim von Hacht

# Datorgrafik



2

## Datorgrafik

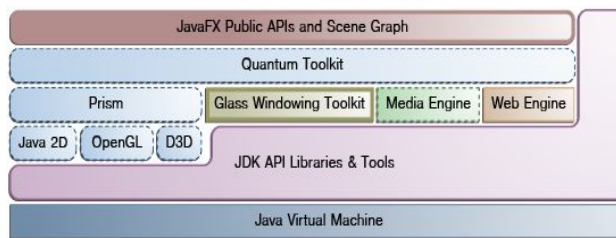
- Används för att rita bilder på en datorskärm
- Varje enskild bildpunkt (pixel) måste kunna adresseras (komma åt/ändra)
- Grafik innebär att (delar av) datorskärmen ritas om med en viss frekvens (ca 60 ggr/sek. i vårt fall)
- Man brukar skilja på 2D- och 3D-grafik, bilden visar en variant av 2D-grafik.

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

- JavaFX ingår i JDK/JRE.

# JavaFX

"[JavaFX](#) is a set of **graphics and media** packages that enables developers to design, create, test, debug, and deploy rich [grafiska] client applications that operate consistently across diverse platforms." // Oracle



Citatet: Lite "branding", får tas med en nypa salt.

JavaFX behövs bara som en viss kännedom inför labbarna. Kommer inte på tenta!!!

# Programstruktur i JavaFX

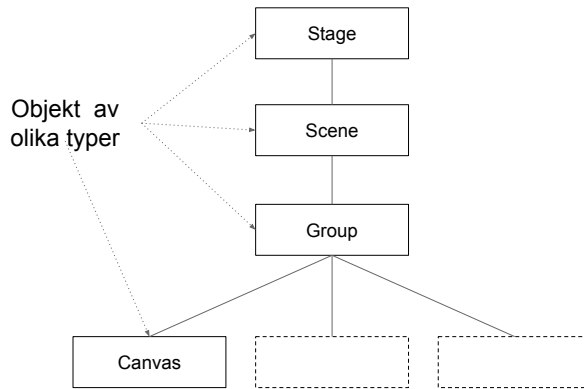
```
public class MyGraphicalProgram extends Application {  
  
    @Override  
    public void init(){  
        // Initialize (create objects, object model)  
    }  
  
    @Override  
    public void start(Stage primaryStage) throws Exception {  
        // Set up graphics  
    }  
  
    public static void main(String[] args) {  
        Launch(args);  
    }  
}
```

4

Ett JavaFX program måste ange extends Application

- Innebär att vi ärver en massa metoder m.m. (jämför klassen Object)
  - Mer i senare kurser
- Innan grafiken startas anropas automatiskt metoden init()
  - Använd för att initialisera applikationen
  - Speciellt skapa objektmodellen för programmet
- 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 visas ett fönster på skärmen (vi måste kode lite för att det skall hända, mer strax)

# 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.

# Start metoden

```
public class MyGraphicalProgram extends Application {  
    GraphicsContext gc; // Accessible to render methods  
    @Override  
    public void start(Stage primaryStage) throws Exception {  
        Group root = new Group();  
        Canvas canvas = new Canvas(width, height);  
        gc = canvas.getGraphicsContext2D();  
        root.getChildren().addAll(canvas);  
  
        Scene scene = new Scene(root);  
        primaryStage.setScene(scene);  
        primaryStage.setTitle("MyGraphicProgram");  
        primaryStage.show(); // Window visible on screen  
    }  
}
```

6

Analys av koden i start-metoden:

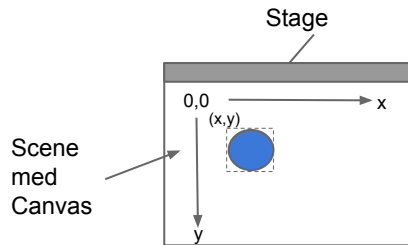
1. Vi skapar ett gruppobjekt
2. ..och en rityta (Canvas)
3. Frågar ritytan efter en grafikkontext. Mer strax ...
4. Lägger till ritytan till gruppen
5. Skapar scenen.
6. Lägger till gruppen till scenen
7. Lägger till scenen till fönstret.
8. Sätter titel på fönstret.
9. ... och slutligen ritar ut på skärmen

Man får se upp: Det finns t.ex. flera olika Canvas!

- Det skall stå: import javafx.... Canvas, alltså inledas med javafx.

# Lågnivå Rendering

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



Grafikkontext



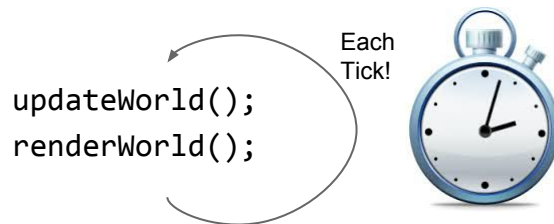
Utritning kallas **rendering**

- 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)

# Animeringsloop



8

Det vi gör är att:

- Uppdatera världen rent logiskt, flytta saker logiskt, nya positioner, objekt kommer och går (`updateWorld`)
- Rita ut en bild av världens nya tillstånd (`renderWorld`)
  - 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.



# En JavaFX Timer

```
long previousTime = nanoTime();
AnimationTimer timer = new AnimationTimer() {
    // Called by JavaFX approx. 60 times / sec.
    public void handle(long currentNanoTime) {
        long elapsedNanos = currentNanoTime - previousTime;
        if (elapsedNanos > 250000000) {
            updateWorld(); // Update Logic
            renderWorld(gc); // gc is GraphicsContext
            previousTime = currentNanoTime;
        }
    }
};
timer.start(); // Start timer
```

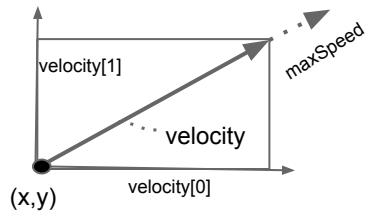
9

En JavaFX AnimationTimer är ett objekt som anropar en fördefinierad metod, handle, 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 anropas automatiskt av JavaFX före varje rendering (d.v.s. ca. 60 ggr/sek)
- Metoden får som parameter aktuell systemtid (currentNanoTime).
  - För att minska hastigheten på renderingen använder vi systemtiden (i koden ovan)

# Rörelse

```
class Spaceship {  
    final double maxSpeed;  
    final double[] velocity;  
    double x;  
    double y;  
  
    void move() {  
        this.x += velocity[0];  
        this.y += velocity[1];  
    }  
}
```

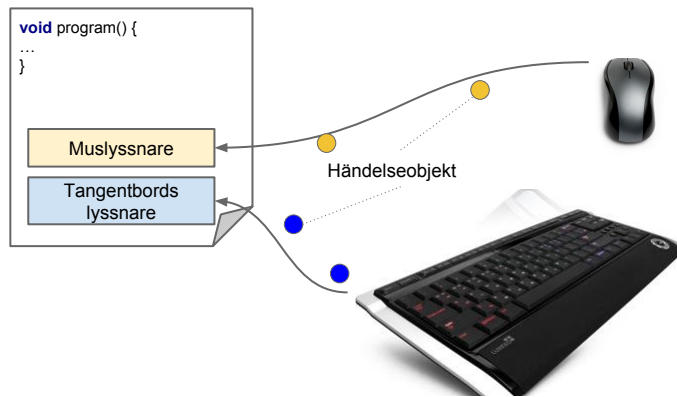


10

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

- Position, x och y.
- Hastighet (velocity), d.v.s. en vektor som beskriver hastighet (= längden av vektorn) och riktningen
  - Om velocity är 0 så står objektet stilla
  - Kan även använda en normaliserad vektor + en hastighet (att skala med).
- Ofta används en maxhastighet som inte får överskridas.
- För att flytta objektet finns en metod move() som uppdaterar x och y utifrån hastigheten.

# Händelsestyrda Program



11

Inmatningen till våra program har hittills använt en kommandorad.

- De flesta program fungerar inte på det sättet ...
- .. de är istället **händelsestyrda**

Ett händelsestyrt program kommer att ta emot indata då olika typer av yttre händelser inträffar

- Man klickar på musen, trycker på en tangent
  - Exakt hur det går till kan vi inte gå in på, vi säger att ett **händelsesystem** sköter det hela
  - Vi konstaterar att en "händelse" i form av ett objekt kommer att skickas som ett argument till en **"lyssnar"-metod (event handler)** då vi t.ex. trycket på en tangent
    - Objektet innehåller information om vilken typ av händelse som inträffat och data om händelsen
      - T.ex. En moshändelse och muspekarens position, en tangenthändelse och vilken tangent, etc
  - Det är vi själva som skapar lyssnarmetoderna.

# Metodreferenser

```
// Use in iteration
private void renderCountry(Country c) { // The method
    ...
}

diceWars.getCountries().forEach(this::renderCountry);
```

Metodreferenser

```
// Use in JavaFX
private void animationFinished(Event e) { // The method
    from.setSelected(false);
    country.setSelected(false);
    ...
}

private ParallelTransition dicesAnimation = ... ;
dicesAnimation
    .setOnFinished(this::animationFinished);
```

12

Man kan ha referenser till metoder i Java

- Vi går inte in på detaljerna här (man kan spara referenser i variabler d.v.s. det finns typer för olika sorters metoder m.m.)
- En metodreferens skrivs: objektet :: metoden (dubbla kolon)
  - Objektet vi använder är this.

Användning

- För intern iteration av samlingar (forEach)
  - Man anger vilken metod som skall anropas för varje element i samlingen (elementet skickas som argument till metoden, d.v.s. metoden måste ha en parameter av elementtypen)
- I JavaFX då vi skapar händelsestyrda program

# Lyssnarmetoder

```
void keyPressed(KeyEvent event) { // Event handler for keyboard
    switch (event.getCode()) {
        case LEFT:
            catchTR.bucketLeft();
            break;
        case RIGHT:
            catchTR.bucketRight();
            break;
        default:
    }
}

@Override
public void start(Stage primaryStage) throws Exception {
    ...
    Scene scene = new Scene(root);
    // Register event handlers (metod references!)
    scene.setOnKeyPressed(this::keyPressed);
    scene.setOnKeyReleased(this::keyReleased);
    ...
}
```

13

Lyssnarmetoder är metoder som har en Event-parameter (finns flera olika t.ex. KeyEvent i bilden)

Vi måste koppla lyssnarmetoderna till händelsesystemet

- Så att systemet vet vart händelseobjekten skall skickas.
- Görs genom att skicka en metodreferens som argument till en "registreringsmetod" (setOnKeyReleased i bilden)
  - Finns flera olika registreringsmetoder (dessutom olika för olika objekt)

# Grafiska Användargränssnitt

```
private FlowPane createPlayersPanel() {  
    FlowPane fp = new FlowPane();  
    fp.setStyle(flowPane);  
    for (Player p : diceWars.getPlayers()) {  
        Label l = new Label(p.getName());  
        l.setStyle("-fx-padding: 5px;");  
        + colorToWeb(p.getColor());  
        fp.getChildren().add(l);  
        playerLabel.put(p, l);  
    }  
    Button next = new Button("Next");  
    next.setStyle("-fx-padding: 5px;");  
    -fx-background-color: gray;");  
    next.setOnMouseClicked(this::next);  
    fp.getChildren().add(next);  
    return fp;  
}
```



Avancerat FX GUI

14

Förutom "lågnivå"-rendering m.h.a. GraphicsContext kan man i JavaFX skapa [grafiska användargränssnitt](#) (Graphical User Interface (GUI))

- Det finns färdiga klasser för paneler, knappar, menyer, texttrutor, layouter, m.m.
  - Alltså objekt som automatiskt renderas (med visst utseende, går dessutom att "styla" med CSS t.ex.)
  - Det går att koppla lyssnarmetoder till objekten
  - Inget vi ger oss in på att koda (dyker upp färdigt i någon laboration).
- I Bilden: Kod och GUI stämmer inte, bara exempel ...