



# Hello world

## Introduksjon

Formålet til denne leksjonen er å lære hvordan man får satt opp et Java-prosjekt. I tillegg skal du lære litt om sammenhengen mellom JavaFX-koden og JavaFX-appen.

## Steg 1: Sette opp Java-prosjektmappe og app-klasse

Eclipse strukturerer koden i såkalt prosjekter. Vanligvis har en ett Java-prosjekt som meste lager små app-er, så er det greit å samle dem i ett Java-prosjekt.

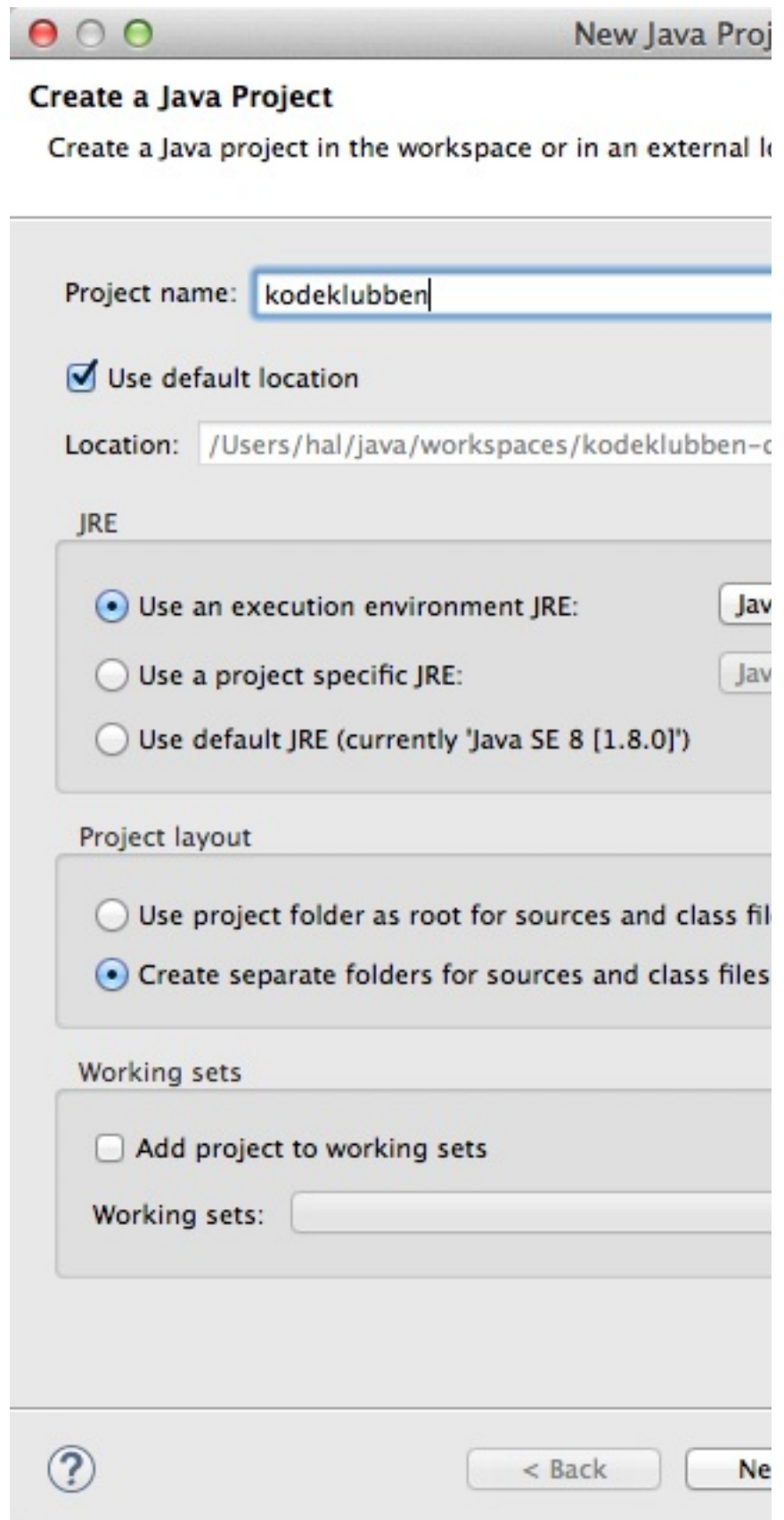
Et prosjekt er enkelt sagt en mappe med innhold/oppsett tilpasset typen av programmeringsspråk, så når du skal lage en ny app med JavaFX må du lage en mappe med flere under-mapper, og en av disse heter `src` og vil inneholde koden. Uoversiktlig, spesielt hvis du har flere app-er i samme prosjekt, så bør du gjøre så kan lage Java-filen for app-en din!



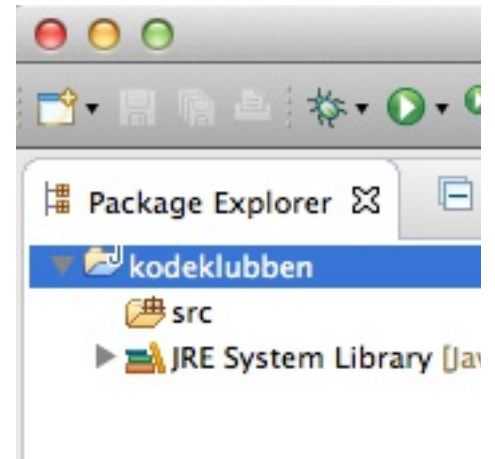
## Sjekkliste



Lag et nytt Java-prosjekt ved å velge `File > New > Java Project`. Du vil da få opp et skjema hvor du bl.a. kan fylle inn navnet på prosjektet og pakke- og klasse-navn. Merk at du bør holde deg til de engelske bokstavene a-z og underekstremer i innstillingene lar du være.

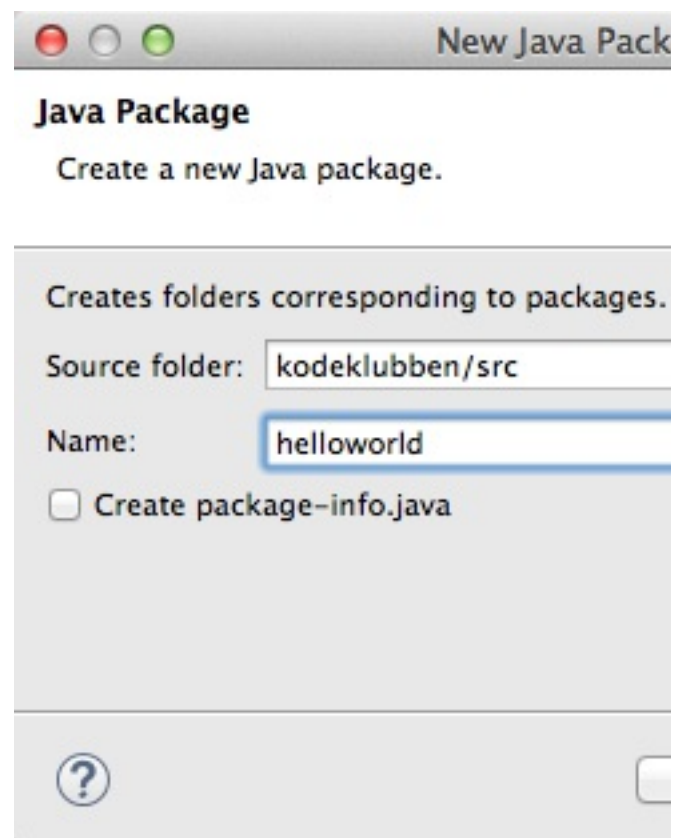


I **Package Explorer**-panelet vil du se at det dukker opp en mappe **src** og en mappe som heter **JRE System Library [JavaSE-1.8]**. Mens **JRE System Library [JavaSE-1.8]** viser at prosjektet er JavaFX. Skjermtutklippet under viser omtrent hvordan det vil bli:

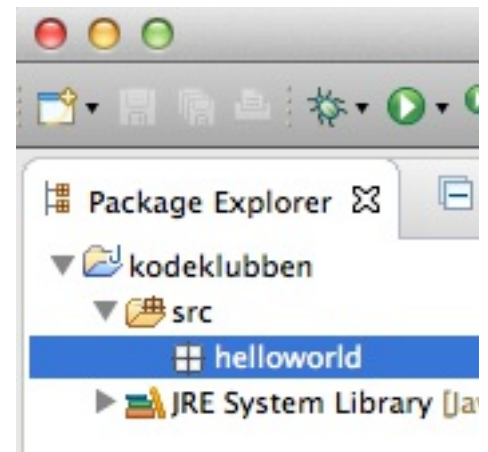


- Lag en ny Java-*mappe* for app-en i denne leksjonen. Java kaller som mapper. Pass først på at du har valgt (klikket på) riktig Java **New > Package** eller ikonet som ser ut som en pakke med et pluss-ikon. Klikk på **src**-mappa og velg **New > Package**.

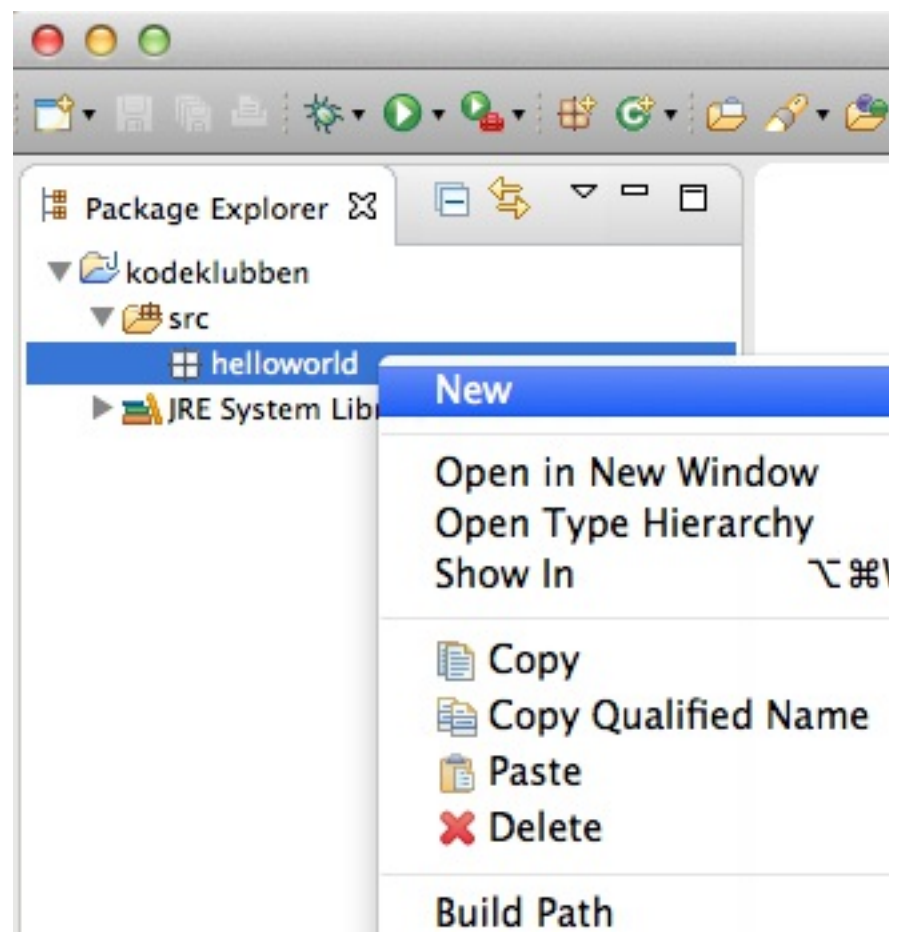
Du vil da få opp et skjema hvor du kan skrive inn hvilken kode-*pakke*-navnet. Kode-mappen skal være **kodeklubben/src** (eller **/src**). Pakkenavn inneholder som regel bare små bokstaver, alt **helloworld**.



Skjermutklippet under viser omtrent hvordan det vil bli seende i



- ☐ Lag en ny Java-klasse (Java-filer kalles *klasser*) ved å høyre-klikke på **helloworld** og velge **New > Class**.



Du vil da få opp et skjema hvor kode-mappa og pakken allerede er opprettet. Klassenavn starter alltid med stor forbokstav, og hvert delord starter med stor forbokstav. Så når vi nå skal lage en app vi kaller Hello World-app

New Java Class

### Java Class

Create a new Java class.

Source folder:

Package:

☐ Enclosing type:

---

Name:

Modifiers: ☒ public ☐ package ☐ private  
☐ abstract ☐ final ☐ static

Superclass:

Interfaces:

Which method stubs would you like to create?


☐ public static void main(String[] args)

☐ Constructors from superclass

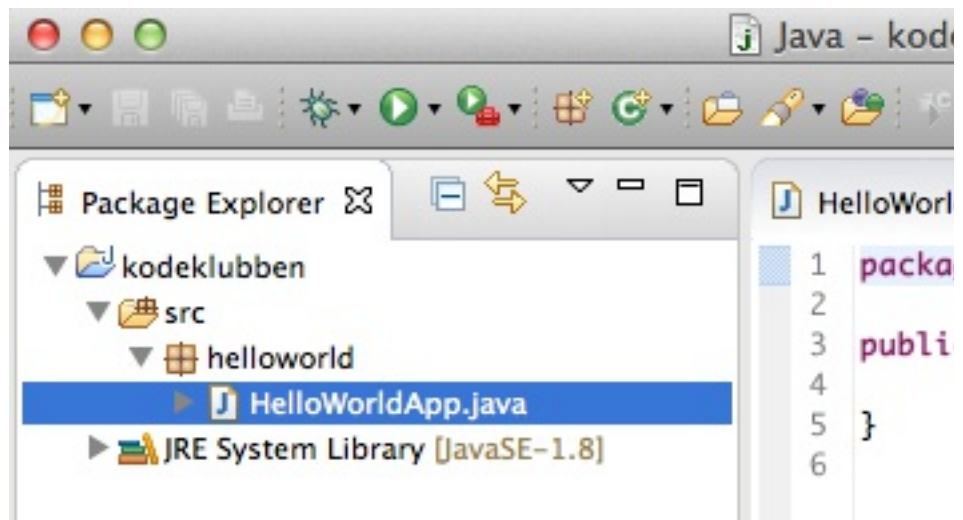
☒ Inherited abstract methods

Do you want to add comments? (Configure templates...)

☐ Generate comments



Skjermutklippet under viser omtrent hvordan det vil bli seende i



## Steg 2: Skrive og kjøre Hello World

En Hello World-app er ment å være den enkleste app-en en kan tenke dette:



# Hello world

En slik app må inneholde følgende elementer:

- ☐ Den må bygge på `Application`-klassen (i pakken `javafx.application`) og implementere `Application`-klassedefinisjonen. Uten det, er klassen rett og slett ikke en app.
- ☐ Den **må** inneholde en metode (Java-funksjoner kalles *metoder*) som kalles `start` og som tar en `Application` som argument. Denne metoden er den første metoden som blir kalt når appen startes.
- ☐ Den *kan* inneholde en `init`-metode som typisk brukes for å initialisere appen.

slik metode.

- ☐ Oppstartsfunksjonen `main`, som kjøres når klassen din startes s klassen din som argument. Når du kjører koden vil følgende skje
  - ☐ App-en din vil bli laget. App-en vil være et **HelloWorldAp** klassen.
  - ☐ `init`-metoden vil bli kalt (hvis du har en, og det har ikke
  - ☐ App-vinduet vil bli laget (automatisk av JavaFX, ikke av vå
  - ☐ start-metoden blir kalt med app-vinduet som argument (d navnet)

Her er den nødvendige koden, med kommentarer:

```
// klassen ligger i pakken helloworld, og
// det må stemme med package-deklarasjonen
package helloworld;

// med en import-setning, så slipper en å skrive hele navnet
// vi har like godt en import-setning for hver av klassene vi
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.layout.Pane;
import javafx.scene.text.Font;
import javafx.scene.text.Text;
import javafx.stage.Stage;

// bygg på Application-klassen med extends
// det er det som gjør at vår klasse kan fungere som en JavaF
public class HelloWorldApp extends Application {

    // her er metoden som kalles med app-vinduet
    // den tar inn app-vinduet som argument
```



```

public void start(Stage primaryStage) {
    // vi lager oss det grafiske tekst-objektet, av typen Text
    Text helloWorldText = new Text("Hello world");
    // så sier vi hvor i vindet den skal plasseres, x- og y-koordinater
    helloWorldText.setLayoutX(10);
    helloWorldText.setLayoutY(50);
    // vi lager oss et font-objekt, av typen Font
    Font font = Font.font("Arial", 36);
    // og setter den som teksten font
    helloWorldText.setFont(font);

    // vi må også lage et panel, av typen Pane
    Pane root = new Pane();
    // vi setter ønsket størrelse, bredde og høyde
    root.setPrefWidth(300);
    root.setPrefHeight(200);
    // og putter teksten inni
    root.getChildren().add(helloWorldText);

    // til slutt legges panelet inn i app-vinduet
    primaryStage.setScene(new Scene(root));
    // og vises frem
    primaryStage.show();
}

// dette er den egentlig oppstartsmetoden
public static void main(String[] args) {
    // kall den innebygde funksjonen launch, med app-klassen
    launch(HelloWorldApp.class, args);
}
}

```

## Sjekkliste

- ☐ Skriv inn koden over, ved å kopiere og lime inn linjene i din egen kode. Java-nøkkelord som `package`, `class`, `public`, `void`, `new`, `Text`, `Font`, `Scene`, `Stage`, `Pane`, `launch` osv. får ulik farge. Innebygde Java-nøkkelord som `package`, `class`, `public`, `void`, `new`, `Text`, `Font`, `Scene`, `Stage`, `Pane`, `launch` osv. verdier er blå osv. Dette hjelper oss å skjønne hvordan Eclipse f



så har vi kanskje glemt en " som avslutter en tekst-verdi.

- ☐ Kjør koden ved å høyreklikke på fila eller i editoren og velge `Run` vindu:



# Hello world

- ☐ Lek litt med koden over. Prøv f.eks. å endre verdiene som styrer (`setLayoutX(...)` og `setLayoutY(...)`), skriftstypen (`Font.family` og `setPrefHeight(...)`):

- ☐ Skriv inn en annen tekst, f.eks. navnet ditt.
- ☐ Endre posisjonen slik at teksten kommer lenger ned og til høyre.
- ☐ Finn en annen skriftstype du liker og se hva som skjer når du øker fonten. Unngå at toppen av teksten kuttet.
- ☐ Velg en kjempestor font og skriv en laaaaang tekst. Øk vinduets høyde.
- ☐ Deklarer en variabel **windowHeight** med `int windowHeight` og en formel som beregner y-posisjonen du setter med `setLayoutY(...)` i vinduet.

- ☐ Legg merke til at det er ulike måter å sette ulike verdier på:
  - ☐ Teksten settes direkte når en lager **Text**-objektet med `new Text(...)`
  - ☐ De fleste verdier settes med egen metoder som begynner med `set`

- ☐ Skriftstypen settes også når den lages, men den lages med `Font(...)`.

## Steg 3: Bruke kode-komplett Eclipse

Eclipse inneholder mange nyttige funksjoner for å gjøre koding mer effektiv og kan både *foreslå* kode du kan skrive og *rette* enkle feil i koden.

- ☐ Eclipse holder rede på hvilke navn (på variabler, klasser og pakker) på hva du har skrevet inn. Lag en ny linje under der skriftstypen variabelen for Text-objektet f.eks. `hello`. Så holder du nede ctrl-f og får en liste over alle navn som begynner med **hello**:

```
public void start(Stage primaryStage) {
    Text helloWorldText = new Text("Hello World");
    helloWorldText.setLayoutX(100);
    helloWorldText.setLayoutY(100);
    Font font = Font.font("Arial", FontWeight.NORMAL, 14);
    helloWorldText.setFont(font);
    helloWorldText.setText("hello");
}
```

A screenshot of the Eclipse IDE showing a Java code file. The code is for a simple application window. The variable 'helloWorldText' is declared and initialized. The text 'hello' is being assigned to the 'text' property of 'helloWorldText'. A code completion popup is visible, showing a list of suggestions for the variable 'hello'. The suggestions are: 'helloWorldText : Text', 'helloworld', and 'HelloWorldApp - helloworld'.

Her ser du at Eclipse foreslår et variabelnavn (**helloWorldText**), et pakke- eller klassenavn (**HelloWorldApp**). Hvis du velger **helloWorldText**, så legges denne til i koden. Dette kalles *komplettering* (eng: *code completion*) og gjør det bl.a. greit å bruke lange navn.

- ☐ Eclipse vet hvilke verdier du kan sette for ulike typer grafiske objekter. Hvis du skriver inn et punktum ( `.` ) etter **helloWorldText**-navnet. Eclipse vil automatisk vise en liste over alle metodene til **Text**-objektet. Hvis du skriver inn

begynner med nettopp **setF**:

```
public void start(Stage primaryStage) {
    Text helloWorldText = new Text("Hello world");
    helloWorldText.setLayoutX(10);
    helloWorldText.setLayoutY(50);
    helloWorldText.setFill(Color.BLUE);
    Font font = Font.getDefaultFont();
    helloWorldText.setFont(font);

    Pane root = new Pane();
    root.setPrefWidth(300);
    root.setPrefHeight(300);
    root.getChildren().add(helloWorldText);
    primaryStage.setScene(new Scene(root, 300, 300));
    primaryStage.show();
}
```

Der finner du bl.a. **setFill** og **setFont**. **fill** og **font** kalles *egenskaper* med metoder som har **set** foran egenskapsnavnet.

Argument-typen viser hva slags verdi du må gi inn. F.eks. tar **setFill** en **javafx.scene.paint.Paint**). Velg setFill fra lista og skriv inn **Color.BLUE** det du har skrevet.

☐ Kjør app-en din igjen, så ser du effekten av kallet til setFill-metode.

## Hva har du lært?

- lage nye Java-prosjekter med **New > Java Project**
- lage nye Java-pakker med **New > Package**
- lage nye Java-klasser med **New > Class**
- hva en klasse må ha av kode for å bli en app-klasse
- kjøre app-klassen som en Java-applikasjon
- hvordan plassere en tekst i et vindu
- hvordan endre verdier for plassering, skriftstype, farge og vindustørrelse
- hvordan bruke kode-kompletteringsfunksjonen

I leksjonen **FXML-logo** vil du lære hvordan lage skjerminnhold med FXML.

