ex6eventdriven

Joachim von Hacht

Metodreferenser

```
doIt(this::sayIt);  // Method as parameter

void sayIt(String msg) {
   out.println("Got a callback ...");
   out.println(msg);
}

// Param type Consumer is type for void method
// with single String param (as sayIt above)
void doIt(Consumer<String> callBack) {
   ...
   callBack.accept("Message from do it");
}
```

2

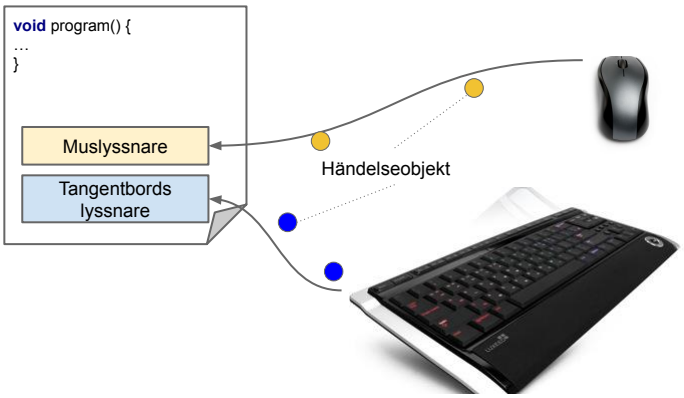
Man kan ha referenser till metoder i Java

- 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 kan skrivas: objektet :: metoden (dubbla kolon)
    - Objektet vi använder är this.

Användning

- Istället för lambda-uttryck (t.ex. i samband med traversering, filtrering av samlingar)
- Callback metoder, metoder som skall anropas tillbaka, senare.
    - JavaFX använder callback-metoder då vi skapar händelsestyrda program, s.k. lyssnar metoder

Händelsestyrda Program



3

Inmatningen till våra program har ofta 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 (eventhandler)** då vi t.ex. trycker på en tangent
    - Objektet innehåller information om vilken typ av händelse som inträffat och data om händelsen
      - T.ex. En mushändelse och muspekarens position, en tangenthändelse och vilken tangent, etc
  - Det är vi själva som skapar lyssnarmetoderna.

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);
     ...
}
```

4

Lyssnarmetoder är (callback)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-paddin
+ colorToWeb(p.getColor()));
      fp.getChildren().add(l
      playerLabel.put(p, l);
   }
   Button next = new Button("
   next.setStyle("-fx-padding
-fx-background-color: gray;")
   next.setOnMouseClicked(thi
   fp.getChildren().add(next)
   return fp;
}
```

Avancerat FX GUI

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, textrutor, 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 (finns färdiga menyer i någon lab)

Anm: I Bilden: Kod och GUI stämmer inte, bara exempel ...