Fler Design Patterns

Objekt-orienterad programmering och design

Alex Gerdes och Sólrún Halla Einarsdóttir, 2018

Vad är ett design pattern?

- Ett *design pattern* (designmönster) är en (ofta namngiven) generell lösning av en vanligt återkommande situation inom (mjukvaru-)design.
  - Vi pratar mest om design patterns inom objekt-orienterad design, men de existerar (mer eller mindre uttalade och erkända) inom alla paradigmer.

- Ett design pattern har ingen färdig kod – de är abstrakta mallar för hur ett problem kan lösas.
  - ”Formaliserade” best-practices.
  - I en given situation och kontext kan vi instansiera ett design pattern med specifika klasser, metoder, etc, och få en färdig lösning.

Varför design patterns?

- Design patterns låter oss återanvända struktur och metodik som andra smarta designers redan kommit på, och som använts och prövats under lång tid.

- Design patterns sätter namn på vanligt förekommande strukturer, vilket effektiviserar kommunikation mellan designers.

- Viktigt att lära sig namnen och strukturerna (”remember”, ”understand”), samt att lära sig känna igen situationerna där de är användbara (”apply”) (och även situationerna där de **inte** bör användas).

Övning

- Börja från koden som finns att ladda ner från hemsidan.
  - Vår gamla kod från förra veckan har fått ett helt nytt sub-paket `tda551.polygon`, som tillhandahåller ytterligare en variant på implementation av polygoner. Koden utanför paketet är (modulo några få tillsnyggningar) densamma som förra veckan.
  - Sub-paketet `tda551.polygon` är inte helt färdigimplementerat ännu. Specifikt är metoderna `translate`, `scale` och `rotate` konsekvent inte implementerade i de konkreta sub-klasserna.
- Rita ett UML-diagram över det nya paketet `tda551.polygon`.
  - För mer utmaning, inkludera även resten av filerna i paketet `tda551`.
  - Ni behöver inte inkludera beroenden på filer som ligger utanför paketet, e.g. `JComponent` eller `Point`.
- Identifiera de design patterns som används (inklusive i implementations-övningen nedan). Vissa har ni sett förut, några är nya.
  - http://www.oodesign.com
  - https://en.wikipedia.org/wiki/Software_design_pattern
  - http://lmgtfy.com/?q=object-oriented+design+pattern

Övning forts.

- Implementera metoderna `translate`, `scale` och `rotate` i de klasser som ännu inte implementerar dessa.
  - Hint: Varje implementation kommer, korrekt gjord, bestå av högst två statements.
  - Specifikationen är som följer:
    - En polygon representeras av en sekvens av nestade objekt, där varje objekt i kedjan representerar en specifik aspekt av polygonen.
    - En polygon initieras till en uppsättning punkter i en `BasePolygon`.
    - Om den därefter e.g. roteras hanteras detta av `RotatePolygon`, etc.
    - En polygon ska bara ha en rotation (etc) – roteras polygonen flera gånger ska dess `RotatePolygon`-objekt uppdateras.
- För mer utmaning: Implementera en ny klass `PolygonGroup` som representerar en grupp av polygoner. Gör det möjligt att applicera rotation och övriga transformationer på hela gruppen. Vissa förändringar av existerande kod och struktur är nödvändiga.