HÖGSKOLAN
I GÄVLE

AKADEMIN FÖR TEKNIK OCH MILJÖ

Avdelningen för industriell utveckling, IT och samhällsbyggnad

Objektorienterad Design och

Programmering, 7.5Hp, HT2015

Laboration 3

av

Miran Batti

Jonas Öster

Innehåll

Inledning

Detta är en laboration inom kursen Objektorienterad Design och Programmering. Detta är del 3 av labben. I tidigare laborationer skapades datatyper som behövs för att representera figurerna till applikationen. Sedan skapades ett logik system för styrningen av datatyperna i programmet.

Denna del av laborationen bygger ett GUI där användaren kan skapa, flytta, skala, och rotera figurer. Figurerna ska ritas ut på en panel som användaren kan se. Denna labb ska färdigställa programmet.

Krav och förutsättningar

Till labbens förfogande fanns det en uppsättning av färdiga interface och ett exempel på GUI program. Interfacen ska implementeras till programmet på ett sätt som passar och fungerar, och GUI:et ska utvidgas för att fungera med logiken och datatyperna.

Programmet ska kunna rita ut figurer på en panel som användaren ska kunna se och manipulera.

Laborationen följer labbinstruktioner [1] som innehåller frågor och uppgifter som besvaras i denna rapport.

Resultat

Del 1

a. Till GUI:et användes det färdiga exemplet som tilldelades med uppgiften.

b. För testandet användes en `Builder` instans i `ControlPanel` klassen för att anropa metoder från `FigureHandler`. Från `FigureHandler` utförs metoder som `rotate()` och `scale()` och sedan görs en utskrift med `printAll()` för att kontrollera om GUI interagerar med logiken korrekt. Programmet skriver ut positionerna till figurerna och allt fungerar som det ska.

Del 2

a. En ny package bildades för att göra det lättare att se vart alla olika klasser tillhör. Den nya package heter interface och den innehåller alla interface. Som uppgiften föreslår så skapas en `Drawable` interface. Alla figurklasser implementerar sedan `Drawable` med metoded `draw()`.

b. Interfacet `FigurePainter` med en klass `FigurePainterImpl` implementerades på liknande sätt som klasser från föregående laboration. Den skapas av `Builder` som skickar med argumenten en lista med figurer som är `Drawable`, och ett objekt av typen `PrimitivesPainter`.

c. Det första som behövs till `PrimitivesPainterImpl` är en instans av `Graphics` klassen. Klassen instansieras först i GUI:et och skickas vidare till `PrimitivesPainterImpl`(den går genom en del andra klasser först). Det viktiga här är att det är alltid samma instans av Graphics som används för att rita ut figurer.

Till ritandet av linje, punkt, cirkel, rektangel, och triangel används `paint()` metoderna i `Graphics`. Linjen använder sig av `drawLine()`; cirkeln och punkten använder `drawOval()`; triangeln och rektangeln behöver använda `drawPolygon()`.

d. För testandet av ritandet skapades ett Main program som kör GUI:et. I GUI:et kan vi skapa olika figurer och manipulera dem. Testandet visar att programmet fungerar som det ska.

Del 3

a. Strukturen har varit bra. Programmet börjar byggas med den lägsta nivå typerna- figurklasserna, man bygger sedan på figurerna och skapar en logik för att manipulera dem. Sist byggs ett GUI som ritar ut figurerna. Det har varit tydligt med vilka klasser tillhör vart i programmet.

b. Inga större problem har uppstått från själva programmets struktur.

c. MVC[2] är ett designmönster som används för att göra det lättare att ändra och byta de olika delarna i ett program. Man gör det genom att separera data ifrån resten av logiken och användargränssnittet.
Fördelen är att det är lättare att ändra och byta på programmets delar. Men nackdelen är att det är svårare och det tar mycket tid att göra ett sådant program.

d. Denna programs struktur har en del likheter med MVC mönstret. Programmet har delats upp så att det finns en data-del, en controller-del, och en view-del; dock så är datatyperna inte helt skilt från de andra delarna och datatyperna är nästan helt beroende av controllertyperna.

Del 4

En ny klass, `ExportImage`, hanterar programmets bild skapande funktioner. Programmet kan skapa bildfiler JPEG- eller PNG-format. Bilderna sparas i projektmappen och överskrids när en ny bild sparas.

`ExportImage` nås genom menyraden på GUI. Då användaren vill exportera en JPEG eller PNG så kallas `ExportImage` i `ApplicationFrame`. ExportImage använder sig av `BufferedImage` och `ImageIO` för att skapa en bildfil.

Referenser

[1] ”Objektorienterad design och programmering: Instruktionerna till laborationerna”. Jenke, Peter. [Online]. Tillgänglig: https://lms.hig.se/bbcswebdav/pid-343196-dt-content-rid-1408191_1/courses/HT15_18402/oodp_lab_instruktioner_ht15v4.pdf [Senast använd: 2015-09-30]

[2] ”whatis.techtarget,com”. *model-view-controller (MVC)*. Rouse, Margaret. [Online] Tillgänglig: http://whatis.techtarget.com/definition/model-view-controller-MVC [Senast använd: 2015-10-14]