Namn: Leo Blumenberg
Java version:”1.8.0_191"
IDE: Eclipse
E-mail: leo.blumenberg@gmail.com

För min lösning av uppgift 3 så hade jag 3 klasser: Card, CardPanel, CardFrame.

Klassen Card hanterar positionen och bilderna som ritas upp och manipuleras i CardPanel. Konstruktionen av Card kräver position där kortet ska ritas upp samt två ImageIcons på bilder som ska ritas upp på framsidan och baksidan. Card har två viktiga variabler box och side. Box är rektangeln som lagrar inom vilket område som man kan manipulera klassen med musen och dess position. Side är en boolean som avgör vilken sida av Card som ska ritas upp. Card har två viktiga metoder paint() och action(). Paint() kallas i paintComponent i klassen CardPanel och avgör vilken sida av Card som ska ritas upp och ritar sedan upp den. Metoden action() ändrar storleken på box beroende på storleken på den aktiva bilden som ritas upp. Sedan finns ett par get och set metoder för att kunna sätta position och den aktiva sidan som ska ritas upp.

Klassen CardPanel har en superklass JPanel och implementerar klasserna ActionListener, MouseListener och MouseMotionListener. CardPanel ritar upp kort och låter användaren manipulera dem med musen. CardPanel har tre viktiga variabler width, height, timer och en lista cardList. Width och height lagrar storleken på panelen i integer. Timer kallar på metoden actionPerformed() var 50 millisekund. CardList lagrar alla kort som läggs till i CardPanel. CardPanel har fyra viktiga metoder paintComponent(), actionPerformed(), mouseClicked() och mouseDragged(). ActionPerformed() kallas varje gång den får en signal från Timer, då kollar den om storleken på CardPanel har ändrats och sätter width och height till den aktuella storleken. Metoden kallar även på repaint() metoden som i sin tur kallar på metoden paintComponent() och till sist går den igenom cardList och kallar på action() metoden hos alla kort. PaintComponenet() målar först hela CardPanel svart och går sedan igenom cardList och kallar på paint() metoden på alla kort i listan så att de ritas upp. MouseClicked() kallas varje gång användaren klickar med musen och kollar igenom cardList för att se om användaren klickat på något av korten. Om användaren klickar en gång på ett kort så målas kortet längst fram i fönstret, om det dubbel klickas på så byter kortet sida och den andra sidan på kortet målas upp. MouseDragged() kallas när en musknapp är nedtryckt och dras över fönstret. Metoden kollar igenom cardList för att se om musen är över ett kort när metoden kallas, om den är det så målas kortet upp längst fram och kortet centreras på musen och följer den tills att den vänstra musknappen släpps.

Klassen CardFrame har en superklass JFrame och har som syfte att skapa ett fönster som användaren kan interagera med. I CardFrame lägger jag CardPanel och i CardPanel lägger jag sedan in 7 st Card objekt och gör att allt synligt sedan skapar jag en ny CardFrame i en main metod.

När jag kom fram till min lösning så började jag med att göra en ett program som ritar upp en blå rektangel med en svart bakgrund. Jag tog inspiration från uppgift 1 där vi fick kod som ritade upp bollar så jag gjorde tre liknande klasser (Card, CardPanel, CardFrame) som ritade

upp rektanglar. Därpå modifierade jag CardPanel klassen att implementera MouseListener så att jag kunde ta emot signaler från musen och börja manipulera mina rektanglar. Därefter så behövde jag att programmet reagera på att man klickade på rektanglarna så att de till en början bytte färg till röd och sedan tillbaks till blå. Detta gjorde jag genom att implementera MosueListener i klassen CardPanel som lyssnar på mus händelser. Nästa problem var att kunna dra rektanglarna med musen och det löste jag genom att implementera MouseMotionListener i klassen CardPanel och sedan ändra positionen på Card så att den centreras på musen och positionen följer musen tills man släpper vänster musknapp. Nästa steg var sätta en bild på rektangeln och det gjorde jag genom att vid konstruktion av Card klassen ta emot två ImageIcon och sedan rita upp dem i paint() metoden i Card och sedan kalla på metoden i paintComponenet() i CardPanel(). Det sista jag gjorde var att skapa en lista i CardPanel där jag kunde lagra alla Card som skulle ritas upp och sedan göra så att det Card som användaren senast integrera med alltid målas upp längst fram i fönstret.

I min lösning så använde jag mig av java.awt.*, javax.swing.*, java.awt.event.* och java.util.ArrayList från standardbiblioteket. Jag använde java.awt.* för att kunna rita upp rektanglar och för att kunna använda javax.swing.* och java.awt.event.*. Javax.Swing.* använde jag mig av för att kunna använda ImageIcon för att rita upp bilder, JPanel för att kunna lägga in mina kort i, och komma åt paintComponent. och JFrame för att kunna skapa ett fönster som användaren kan interagera med. Jag använde mig av java.awt.event.* så att jag kunde använda mig av ActionListener, MouseListener och MouseMotionListener så att användaren kunde manipulera objekt i fönstret. Java.util.ArrayList använde jag mig för att skapa en lista där jag kunde lagra Cards.

De jag hade svårt med när jag jobbade med uppgiften var att få MouseListener att fungera som jag ville och att rita upp .gif filer. Med MouseListener så tog det mig ett tag att förstå hur den fungera och det tog mig ett tag hitta ett sätt för användaren att interagera med bilderna som ritades upp och det tog mig ett tag att inse att Rectangle har en metod contains(Point) där jag kunde lägga in musens koordinater för att se om användaren klickat på bilden. I efterhand tror jag att det kunde gjorts på ett bättre sätt att identifiera vilket kort som användaren interagerar med. Mitt andra problem var hur jag skulle rita upp .gif filer på ett bra sätt och det löste jag sedan genom att ta emot en ImageIcon vid konstruktion av Card och ritar upp dem med paint() metoden som sedan kallas i paintComponent. Jag tycker att det här var bästa sättet att göra det på efter hur mina Klasser var utformade innan jag började med problemet.