Tentamen, EDAF25 Objektorienterad modellering och design - Helsingborg

2015–06–04, 14.00–19.00

Anvisningar: Denna tentamen består av 4 uppgifter. Preliminärt ger uppgifterna

1. 2 + 2 = 4 p
2. 1 + 2+ 3 + 3 = 9 p
3. 3 + 4 + 6 = 13 p
4. 2 + 2 = 4 p

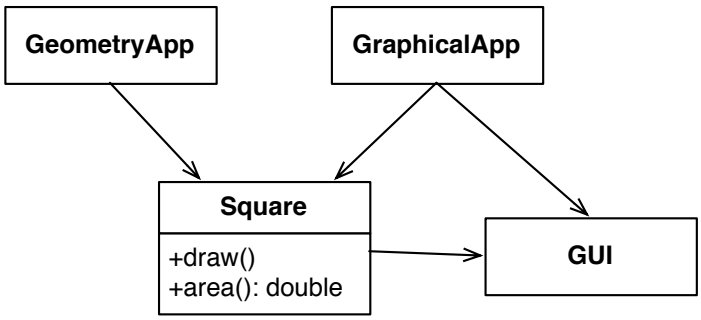
Tillåtna hjälpmedel:

- Martin: Agile Software Development
- Andersson: UML-syntax
- Föreläsningsbilderna F1- 8 (inklusive sedvanliga anteckningar)
- Java snabbreferens

När rättningen är klar meddelas detta på kursens hemsida (`cs.lth.se/kurs/edaf25`).

---

1. a) Square i UML-diagrammet, figur 1, har ansvar att rita ut sig själv och att beräkna sin area. Detta bryter mot SRP (eng: Single Responsibility Principle). Ändra designen så att den inte bryter mot SRP. Visa lösningen med ett nytt UML-klassdiagram. (2p)



Figur 1: Exempel på brott mot SRP

b) Kan brott mot SRP ge problem vid underhåll av kod? Motivera ditt svar kortfattat med hjälp av exemplet ovan och din kunskap om illaluktande design (eng: design smells) (2p)

2. Nedan visas Java-kod för tre klasser: Car, Truck och VehicleTest.

```
public class Car {
    private boolean status;
    public Car(){
        this.status = false;
    }
    public void start(){
        this.status = true;
    }
    public void drive(){
        System.out.println("I'm driving a car!");
    }
    public void stop(){
        this.status = false;
        System.out.println("Car stopped");
    }
    public void test(){
        start();
        if(this.status){
            drive();
            stop();
        }
    }
}
public class Truck {
    private boolean status;
    public Truck(){
        this.status = false;
    }
    public void start(){
        this.status = true;
    }
    public void drive(){
        System.out.println("I'm driving a truck!");
    }
    public void stop(){
        this.status = false;
        System.out.println("Truck stopped");
    }
    public void test(){
        start();
        if(this.status){
            drive();
            stop();
        }
    }
}
public class VehicleTest {
    public static void main(String args[]){
        Car car = new Car();
        testCar(car);
        Truck truck = new Truck();
        testTruck(truck);
    }
    public static void testCar(Car car){
        car.test();
    }
    public static void testTruck(Truck truck){
        truck.test();
    }
}
```

a) Bryter designen mot någon känd designprincip? Motivera ditt svar. (1p)

b) Använd Template Method mönstret för att förbättra designen. Gör de ändringar som behövs utan att förändra beteendet vid exekvering av main-metoden i VehicleTest. Lösningen ges som Java-kod. (2p)

c) Använd istället Strategy-mönstret för att få till motsvarande förbättring. Även här ges lösningen som Java-kod. (3p)

d) Även Factory-pattern skulle gå att tillämpa här. Visa med ett UML-klassdiagram hur. Beskriv implementeringen av klassen *VehicleTest* med Java-kod. De metoder som används av testklassen ska också vara synliga i klassdiagrammet. (3p)

3. Ett företag vill utveckla mjukvara för att styra en kaffeautomat som finns tillgänglig för studenter och lärare på Campus. I automaten kan man köpa tre olika sorters varma drycker: kaffe, te eller varm choklad. Kaffet kostar 10kr, choklad kostar 8kr och te kostar 5kr. Ett normalt köp går till som följer: 1) Kunden stoppar pengar i automaten. 2) Värdet av instoppade mynt visas i automatens display. 3) Kunden väljer dryck. 4) Drycken serveras. 5) Eventuell växel betalas tillbaks. Kunden kan också välja att avbryta köpet och ska då få alla instoppade pengar tillbaka.

   a) Rita ett tillståndsdiagram för systemet. Hårdvaran tar hand om igenkänning av mynt och doseringen av drycker. Tre funktioner ska styras: återlämning av mynt (release), servering av dryck (dispense) samt en display där värdet av instoppade mynt visas (display). Automaten har 3 knappar för val av de respektive dryckerna, 1 knapp för att avbryta köpet och ett myntinkast. Tillsammans ger detta fem olika insignaler att hantera. Du ska använda UML-syntax (3p)

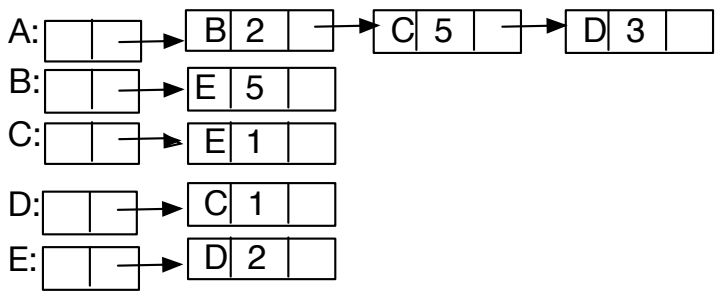
   b) Föreslå klasser och rita ett UML klassdiagram innehållande paket, klasser, attribut (med typer), metoder (med parametrar och returtyper), associationer och beroenden. I diagrammet skall även synlighet för metoderna framgå. Dela upp designen i två paket, *hardware* och control. Låt paketet *hardware* innehålla 4 klasser: *MoneyHandler, BeverageDistributor, Display* och *Beverage*. Paketet *control* ska innehålla minst två klasser. För var och en av klasserna i *control* , beskriv med ord dess tänkta ansvarsområde. (4p)

   c) Rita ett sekvensdiagram som visar exekveringen vid ett köp av en kopp choklad enligt scenariot ovan. Kunden betalar 10kr och ska således få 2 kronor tillbaka tillsammans med sin kopp. Klassdiagram och sekvensdiagram skall överensstämma. (6p)

4. Under kursen har vi pratat om olika sätt att representera en graf däribland närhetslistan (F1). Figur 2 visar ett exempel på en sådan närhetslista.

   a) I vilken ordning besöks noderna i grafen vid en djupet först traversering med start i nod A? (2p)

   b) Använd Dijkstras algoritm för att beräkna kortaste vägen mellan nod A och nod E. Hur många iterationer behövs innan kortaste vägen är funnen med denna metod? Vilken väg kommer att hittas? (2p)



Figur 2: Närhetslista