Tentamen på kurs: **DA339A, Objektorienterad programmering**
Provkod: 2002 Tentamen 2, 2,5 hp
240103 kl. 14.15 – 19.15

Om hjälpmedel:

- **Inga tillåtna hjälpmedel får användas utöver det som finns via tentamen i Inspera och eventuell medtagen engelsk-svensk ordbok eller annan språkkombination (anteckningar får inte förekomma i medtagna ordböcker).**
- **Anteckningar för stöd under tentan får endast göras på papper tillhandahållna av tentamensorganisationen.**
- **Lapp med datorid och lösenord samt skåpnummer och kod är tillåtet att ha med in till tentamen.**
- **Resurs för att rita diagram digitalt med verktyget Visual Paradigm Online på en webbsida finns i tentamen.**
- **För uppgift 8 och 9 är det tillåtet att lämna in svar på papper med det ritade diagrammet istället för att använda Visual Paradigm Online.**

Betygsgränser

- Del med uppgifter för G: 40 poäng
- Del med uppgifter för VG: 1 uppgift (med deluppgifter), godkänd uppgift ger 100p (den något konstiga poängsättningen är för att hantera hur Inspera fungerar). Uppgiften för VG bedöms endast om man uppnår minst 30p på del med uppgifter för G.

För betyg G Godkänt krävs 24 poäng på del med uppgifter för betyg G.

För betyg VG Väl godkänd krävs minst 30p på del med uppgifter för betyg G och att uppgift för VG bedöms som godkänd. Detta ger då en poängsumma i Inspera på 130 poäng eller mer.

Övriga tentamensanvisningar

- Besvara alla frågor med en beskrivning som visar din förståelse och kunskap i frågan. Använd fullständiga meningar som gör det tydligt vad du svarar på och vad ditt svar är.
- För svar som ges i form av kod eller pseudokod krävs att denna är lämpligt formaterad så att man tydligt kan se vilka block av kod som hänger samman. Indentera väl!
- För svar som lämnas på papper (endast möjligt för frågor som innebär att rita något diagram) gäller följande:
    - Följ instruktionerna som anges på tentamensomslaget (det som signeras av tentavakten och innehåller antal inlämnade papper med mera) - det är huvudförutsättningarna för att uppgifterna ska kunna rättas.
    - För text som förekommer i diagrammen måste du skriva läsligt. Svar som inte kan läsas på grund av otydlig handstil erhåller inte poäng.
    - Röd penna får ej användas för svar (reserverad för rättning).
- Anteckningspapper/kladdpapper får tas med från tentamenssalen när tentan lämnats in.

Lärare besöker tentamen för frågor i samband med start samt vid ungefär 16.00 och 17.30 (med många som tentar kan det ta tid att gå runt till alla så tiderna är inte exakta).

För frågor som rör att få fler anteckningspapper, hur du fyller i tentamensomslag, rutiner för toa-besök, rent tekniska frågor om Inspera (inloggning, navigering med mera) eller liknande vänder du dig till tentavakterna.

Rör din fråga instruktionerna för en uppgift vänder du dig till lärare vid något av de tillfällen lärare besöker salen. När lärare besöker salen kommer de att börja i ena änden av salen och sedan gå förbi alla platser en gång. Du kan ställa din fråga när lärare passerar din plats genom att då räcka upp handen för att indikera att du har en fråga. Lärare kommer inte att gå kors och tvärs i salen för frågor ”on demand”. De passerar dig en gång vid varje tillfälle och du behöver ställa dina frågor då.

Tänk på att läsa igenom alla uppgifter på förhand så du vet om du behöver fråga något vid de tillfällen lärare besöker salen.

Uppgifter för betyg G 40 p

Uppgift 1 10p 0,5/påstående

Vilka påståenden är korrekta/sanna och vilka är felaktiga/falska?

Rätt svar i en deluppgift ger 0,5p (totalt 10p på hela uppgiften)
Fel svar på en deluppgift ger 0p, dvs inga minuspoäng.
För kod som förekommer i frågorna eller påståenden som relaterar till hur kod fungerar kan programspråket förutsättas vara Java.

Delfråga	Påstående

1	Antag att metoderna methodA och methodB finns i en klass (se nedan):
	
	`public Person[] methodA( ) {`
	`  /*kod*/`
	`}`
	`public void methodB (Person person) {`
	`  /*kod*/`
	`}`
	
	Metoderna kan användas enligt metodanropet nedan:
	
	`methodB(methodA( ));`
	Svar: FALSKT

2	En abstrakt metod behöver inte implementeras av alla subklasser till superklassen som innehåller den abstrakta metoden. Svar: FALSKT

3	Antag att klassen *ClassB* är en subklass till klassen *ClassA*. Klassen *ClassB* har en metod *foo* som inte finns i klassen *ClassA*.
	
	Då är följande syntax korrekt:
	
	`ClassA ca = new ClassB();`
	`((ClassB)ca).foo();`
	Svar: SANT

4	En klass kan ärva från flera klasser och kan endast implementera ett interface. Svar: FLASKT
5	Om en klass är abstrakt kan man inte skapa objekt av klassen. Svar: SANT
6	En enum är en klasstyp och kan instansieras, dvs. det går att skapa ett objekt av en enum med nyckelordet new. Svar: FALSKT
7	Nyckelordet throws används tillsammans en metods signatur och tvingar metoden att implementera try-catch inne i metod-kroppen. Svar: FALSKT
8	Vid hantering av undantag, kan det finnas endast ett try block och ett finally block men flera catch block. Svar: SANT
9	Ett interface kan ärva en eller flera interfacer. Svar: SANT

10	Nyckelordet final kan användas för att förhindra arv av en klass. Svar: SANT
11	Det kan finnas flera associationer av samma typ mellan två klasser. Svar: SANT
12	Vid överskuggning/overriding har man samma metod som i superklassen men med olika implementation i subklasser i en klasshierarki. Svar: SANT
13	Ett sekvensdiagram visar vilka objekt som existerar under exekvering. Svar: SANT
14	När typen av objekt är bestämt vid kompilering, sker en dynamisk bindning. Svar: FALSKT
15	Polymorfism innebär att ett objekt kan bete sig som om det tillhör två olika klasser. Svar: SANT
16	Klassvariabler och klassmetoder finns endast i en uppsättning som delas mellan samtliga instanser av klassen. Modifieraren `static` används för att ange att en variabel eller en metod är en klassvariabel respektive klassmetod. Svar: SANT
17	Vid generalisering ärvs även associationer som superklassen har till andra klasser. Svar: SANT
18	Ett klassdiagram är ett dynamiskt diagram. Svar: FALSKT
19	Om man tillämpar konceptet Boundary-Control-Entity strikt för klasser i ett system ska där inte finnas någon association mellan klasser av typen Boundary och Entity. Svar: SANT
20	En klass av typen controller i MVC-mönstret hanterar interaktionen med användaren via ett grafiskt gränssnitt. Svar: FALSKT

Uppgift 2 2p

Förklara hur du kan resonera för att avgöra om en association mellan två klasser lämpligtvis kan vara en aggregation eller inte (med aggregation avses här hur detta definieras i kurslitteratur av Bennet).

Riktlinjer för svar: Se steg för att avgöra om något är en komposition eller aggregation i slides till föreläsning F14.
Är förhållandet något som kan beskrivas semantiskt som en helhet som byggs upp av delar där helheten behöver delarna för att ”bli något meningsfullt” kan det vara relevant med en aggregation eller komposition. Om delarna kan ingå i mer än en helhet är det aktuellt med en aggregation (on inte är det en komposition)

Det handlar inte om generalisering, super-klasser, sub-klasser eller hierarkier. Om man blandat in detta i att motivera en aggregation ges 0p.

Man kan inte svara med hur man utläser ur ett diagram eller kod om något **redan är** en aggregation. Svar av denna typ ger 0p.

Det ger inte full poäng om man svarar med konsekvensen av en aggregation (eller komposition), exempelvis att man eventuellt tar bort delarna om helheten raderas. Detta är inte en orsak till att något ska vara en aggregation – det är en konsekvens av att det redan är detta.

Alla associationer innebär beroenden. Man kan inte få full poäng om man förväxlar beroende mellan två klasser i sig för att vara en aggregation. Alla former av associationer innebär något beroende mellan klasser.

Uppgift 3 2p

Förklara kortfattat begreppet abstrakt/abstract klass och hur detta används i förhållande till begreppet generalisering. Motivera nyttan med att använda abstrakta klasser och när detta lämpar sig. Frågan bör kunna besvaras med 6-8 meningar.

Riktlinjer för svar:

- Inte kunna instansiera en superklass, endast göra instanser av sub-klasser
- Samma metod i subklasser men med olika beteende, metodhuvud i superklass men subklasser av gör implementation – möjliggör polymorfism

För poäng krävs att man lyfter fram att en abstrakt klass inte kan instansifieras i sig själv utan måste ärvas eller att man lyfter fram att abstrakta metoder inte har någon metodkropp och måste realiseras i subklasser.

Att inte kunna skapa objekt av en klass kan vara lämpligt om klassen är så generell att objekt som endast är den generella klassen inte är rimligt. Rimliga objekt måste då vara någon sub-klass till den abstrakta klassen.

En klass kan vara abstrakt för att den innehåller abstrakta metoder. Detta är metoder utan metodkropp som en subklass då måste implementera. Detta används för att vis polymorfism kunna se något som den abstrakta superklassen och säkerställa att vissa metodanrop går att göra men hur den metoden är implementerad behöver skilja sig åt mellan subklasser.

För full poäng krävs att man beskriver konceptet och även nyttan med detta. Observera att nytta inte innebär vad kan göra utan varför det är lämpligt/bra att göra något.

Att endast tala om generalisering ger inte poäng. Det ger inte heller poäng att tala om saker som gäller generalisering i stort som om det är något som gäller specifikt för abstrakta klasser.

En abstrakt klass är inte exakt samma sak som en superklass/basklass. En klass kan vara en superklass/basklass utan att vara en abstrakt klass.

Interface och abstrakt klass är inte exakt samma sak.

En abstrakt klass kan ha implementerade metoder med metodkropp, den måste inte endast ha abstrakta metoder. Alla metoder i en superklass ärvs av subklassen men abstrakta metoder måste också implementeras i subklassen (eller någon subklass).

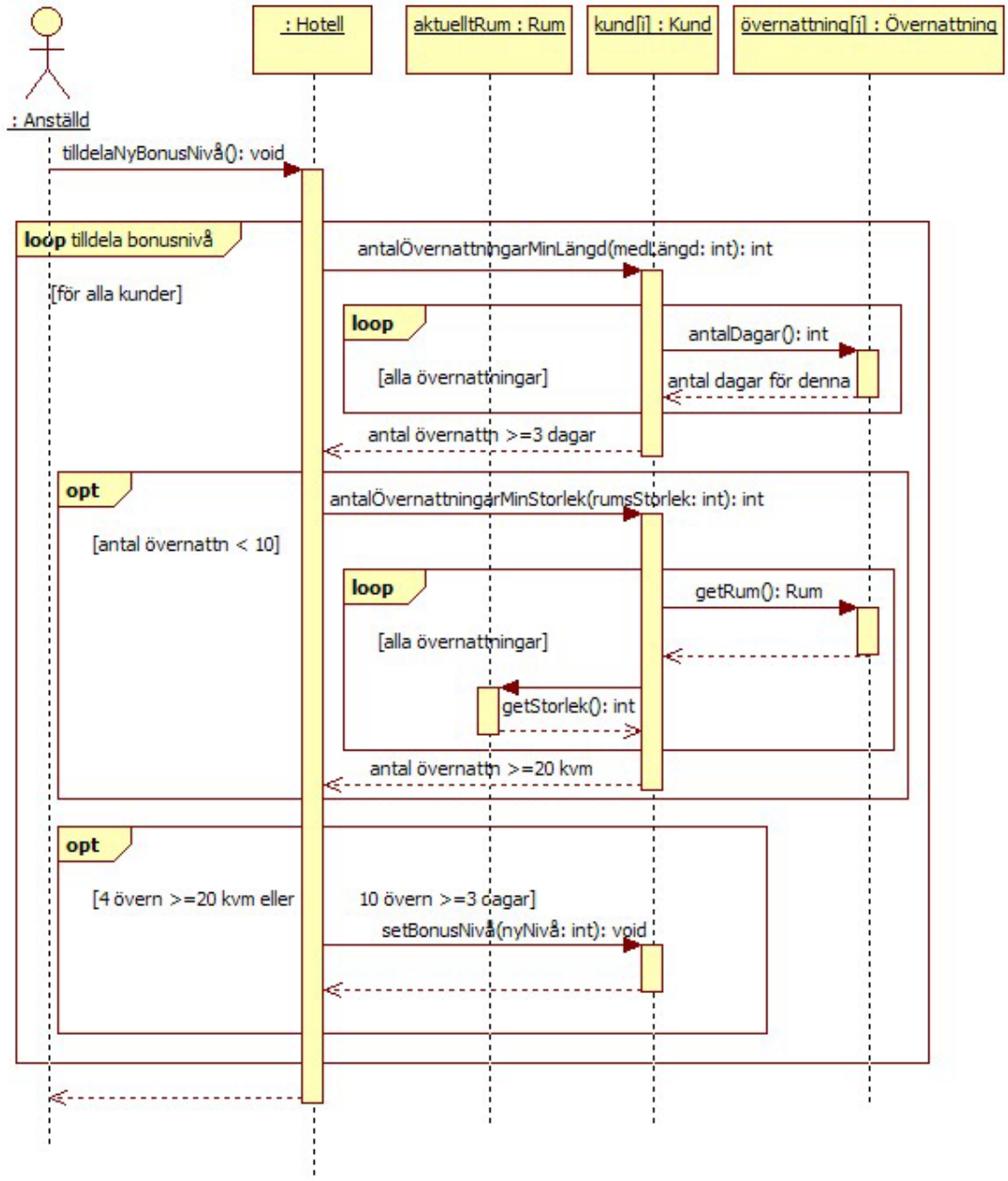
Uppgift 4 4p

Sekvensdiagrammet nedan visar vad som sker när en anställd vill köra den årliga översikten av bonusnivå för kunderna i ett system för hotell-övernattningar. Alla kunder som har gjort något av följande

- övernattat minst 10 gånger och dessa övernattningar varat minst 3 dagar
- övernattat minst fyra gånger i ett rum som är minst 20 kvm

ska sättas till bonusnivå 3. Skriv kod för de två klasserna **Hotell** och **Kund** som finns i diagrammet utifrån information om klassnamn och metoder i sekvensdiagrammet och informationen ovan.

Koden skall visa vilka metoder de två klasserna har enligt diagrammet samt den kod i metoderna som går att utläsa från diagrammet. Vissa antaganden kan göras om namn på lokala variabler, villkor i iterationer och selektioner samt villkor i dessa.



Lösningsförslag

```
class Hotell {
  public void tilldelaNyBonusNivå() {
    for(i=0; i<kund.length; i++){
      antalMin = kund[i].antalÖvernattnignarMinLängd(3);
      if(antalMin<10){
        antalKvm = kund[i].antalÖvernattnignarMinStorlek(20);
      }
      if(antalKvm>=4 || antalMin>=10){
        kund[i].setBonusNivå(3);
      }
    }
  }
}
class Kund {
  public int antalÖvernattningarMinLängd(int medLängd) {
    for(j=0; j<övernattning.length; j++){
      … = övernattning[j].antalDagar();
    }
  }
  public int antalÖvernattningarMinStorlek(int rumsStorlek) {
    for(j=0; j<övernattning.length; j++){
     Rum aktuelltRum = övernattning[j].getRum();
     … = aktuelltRum.getStorlek();
    }
  }
  public void setBonusNivå(int nyNivå) {…}
}
```

Riktlinjer för bedömning: 2p för klass Hotell om allt ok där, 4p totalt: 2p för klass Kund om allt ok där. Slutpoäng beror på samlad bedömning av svar inklusive mindre fel som kan finnas i förhållande till de generella riktlinjerna.

Klass Kund

- Missat metod setbonusNivå(int nynivå) i klass Kund ger avdrag 0,5p
- Metod antalÖvernattningarMinLängd(int medLängd) ok 1p
  - Avdrag 0,5p om loop i metoden felaktig eller om anrop i loopen är felaktigt
- Metod antalÖvernattningarMinStorlek(int rumsStorlek) ok 1p
  - Avdrag 0,5p om loop i metoden felaktig eller om anrop i loopen är felaktigt
- Om alla metoderna i Kund är där men innehållet i metoderna är helt felaktigt eller har stora fel i alla metoder ges max 0,5p av klassens totala 2p

Klass Hotell

- 1p för att få rätt på loopen med innehåll av anrop
- 0,5p för rätt på respektive if-sats med innehåll av anrop
- Avdrag 1p om if-satser ej ligger inuti loopen
- Avdrag 0,5p per felaktigt anrop i loop eller if-sats

Uppgift 5 2p

Metoderna foo och boo finns i någon klass:

```
public String foo(int a, int b){
    String text = ("#1 text = " + (a + b) + "c");
    return text;
}

public String foo(double a, double b){
    String text = ("#2 text = " + (a - b) + "c");
    return text;
}

public String foo(String a, String b){
    String text = ("#3 text = " + a + b + "c");
    return text;
}

public void boo(){
    int a = 2;
    int b = 2;
    System.out.println(foo(a, b));
}
```

Vad blir utskriften när metoden boo exekveras? (2p)

Svar/Lösningsförslag
#1 text = 4c

Uppgift 6 2p

Koden nedan kommer att kasta ett ArrayIndexOutOfBoundsException undantag (exception). Skriv om metoden och visa hur du ska hantera undantaget för att undvika att programmet går ner (kraschar) under körning, samt att programmet ska skriva ut om det gick att returnera text eller inte. (2p)

```
public String returnStringElement(){
    String[] numberArray = new String[1];
    String text = numberArray[1];
    return text;
}
```

Svar/Lösningsförslag:

```
public String returnStringElementLSG(){
    String[] numberArray = new String[1];
    //numberArray[0] = "hej";
    String text = "";
    String answer = "success";
    try{
        text = numberArray[0];
    }
    catch (ArrayIndexOutOfBoundsException e){
        answer = "failure";
    }
    finally{
        System.out.println(answer);
    }
    return text;
}
```

2p: Om utskrift att lyckas efter parsing som inte exekveras om exception uppstår och catch med felmeddelande
1p: Om löst med if-satser, kan vara mindre beroende på lösning
1p: Om try-delen ok men andra problem finns (exempelvis catch utan finally)
1p: Om lagt till finally och att den skriver ett meddelande om det gick att konvertera stringTest eller inte
-0.5p: Om double ej deklarerad utanför try-catch eller om return är fel eller om metodhuvudet saknas.

Uppgift 7 4p

Givna klasser Event och klassen Adress.

Event:

```
public abstract class Event {
    private String name;
    private int date;
    private Adress adress;

    public Event(String name, int date, String city, String street){
        this.name = name;
        this.date = date;
        this.adress = new Adress(city, street);
    }

    public String toString() {
        String textOut = String.format("Name: %s | Date: %s | Adress: %s",
name, date, adress.toString());
        return textOut;
    }
}
```

Adress:

```
public class Adress {
    private String city;
    private String street;

    public Adress(String city, String street){
        this.city = city;
        this.street = street;
    }

    public String toString(){
        String textOut = city + ", " + street;
        return textOut;
    }
}
```

Testkörning av metoden i Main ger resultatet:

```
public class Main {
    public static void main(String[] args) {
        Disco disco = new Disco("Tentamen Disco", 240103, "Malmö",
                                "Kranen street", "Tenta", "DA339A");
        System.out.println(disco);
    }
}
```

Output:

Name: Tentamen Disco | Created: 240103 | Adress: Malmö, Kranen street | Theme: Tenta | DJ playing: DA339A

I följande uppgift ska du använda dig av den givna abstrakta klassen Event och klassen Adress (se till vänster). Inga ändringar eller tillägg är tillåtna i klasserna Event eller Adress. Alla instansvariabler ska vara deklarerade med modifieraren private. (4p totalt)

Skapa en klass Disco. Denna klass ska ha två instansvariabler theme med datatyp String och playingDJ med datatyp String. Klassen Disco skall ärva klassen Event (se ovan). Alla instansvariabler ska vara deklarerade med modifieraren private. (1p)

Skapa en enda konstruktor för klassen Disco med nödvändiga parametrar för att initiera alla instansvariabler i klassen Disco och dess superklass. (2p)

Skriv en toString-metod i klassen Disco så att resultatet nedan erhålls. (1p)

En testkörning av metoden i Main samt output kan du se ovan.

Du ska svara på deluppgifterna i samma svar – skriv allt som efterfrågas för klassen Disco i ett svar.

Lösningsförslag:

```
public class Disco extends Event{
    private String theme;
    private String playingDJ;
    public Disco(String name, int date, String city, String street, String
theme, String playingDJ)
    {
        super(name, date, city, street);
        this.theme = theme;
        this.playingDJ = playingDJ;
    }
    //@Override
    public String toString() {
        String textOut = String.format("%s | Theme: %s | DJ playing: %s",
super.toString(), theme, playingDJ);
        return textOut;
    }
}
```

Uppgift 8 7p

Uppgiftstext

Konstruera ett lämpligt klassdiagram utifrån systembeskrivningen nedan. Identifiera lämpliga **entity-klasser** för systemet. Använd dig av så lämpliga associationstyper som möjligt för att beskriva hur klasserna relaterar till varandra. Använd generalisering, aggregation och/eller komposition där detta är lämpligt. Skriv ut namn på alla associationer för att förtydliga dessa. Om det går att ange en multiplicitet för en association så skall detta finnas – gör lämpliga antaganden där detta kan vara rimligt. Gör inga tillägg med information som överhuvudtaget inte finns i systembeskrivningen.

Det är inte nödvändigt att visa attribut eller operationer i klassdiagrammet. Det räcker att skriva ut klassnamn och namn på associationer och multiplicitet. Korrekt UML-notation ska användas (se notationshjälp nedan).

Systembeskrivning

Man vill skapa ett program som stödjer skrivandet av böcker, vilket är en stor, mer eller mindre seriös, hobbyverksamhet för många. Detta är den inledande beskrivningen man har efter några första intervjuer med presumtiva användare av systemet.

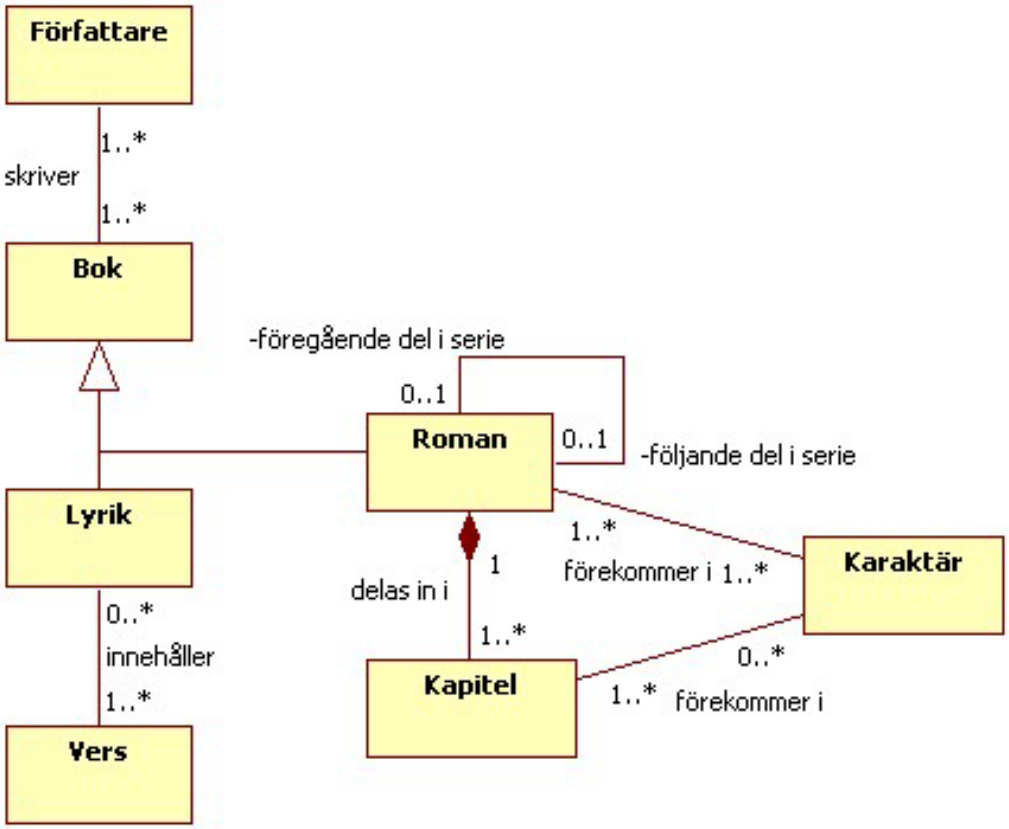
Man hittade två tydliga typer av böcker som skrevs - romaner och lyrik. Dessa är böcker men som skiljer sig åt väsentligt i hur detaljerna ser ut för deras uppbyggnad. Flera av författarna man intervjuat arbetar med andra författare på samma bok. Så en bok kan ha flera författare till den innehållande texten.

Lyrik innehåller verser. De som författade lyrik ville kunna hantera verser för sig och kunna sätta ihop verser till böcker med lyrik. En vers kan kanske komma att ingå i flera olika böcker i slutänden.

De författare som skrev romaner vill kunna hantera flera romaner som följer på varandra i en serie. Ingen har dock gett uttryck för att det behövs en trädstruktur utan det räcker att kunna hantera en kedja av romaner. Så man behöver veta vilken roman som utspelar sig före en annan och vilken som kommer efter. Givetvis så förekommer böcker som inte ingår i någon serie.

Det finns en mängd karaktärer som behöver hanteras gällande romanerna. Man behöver hålla reda på vilka karaktärer som förekommer i vilka romaner men också på en mer detaljerad nivå i vissa fall. I de mer detaljerade fallen behöver man veta i vilka kapitel som en karaktär förekommer. Romaner utan direkt kapitelindelning beräknas ha endast ett kapitel. Man ansåg också att det kunde finnas kapitel som inte innehåll någon väsentlig karaktär.

Lösningsförslag



Lösning med separat klass för Serie är också om men full poäng förutsätter att endast böcker i en serie ingår. Det vill säga en serie innehåller minst två böcker och ett Bok-objekt måste inte ha någon association till något Serie-objekt.

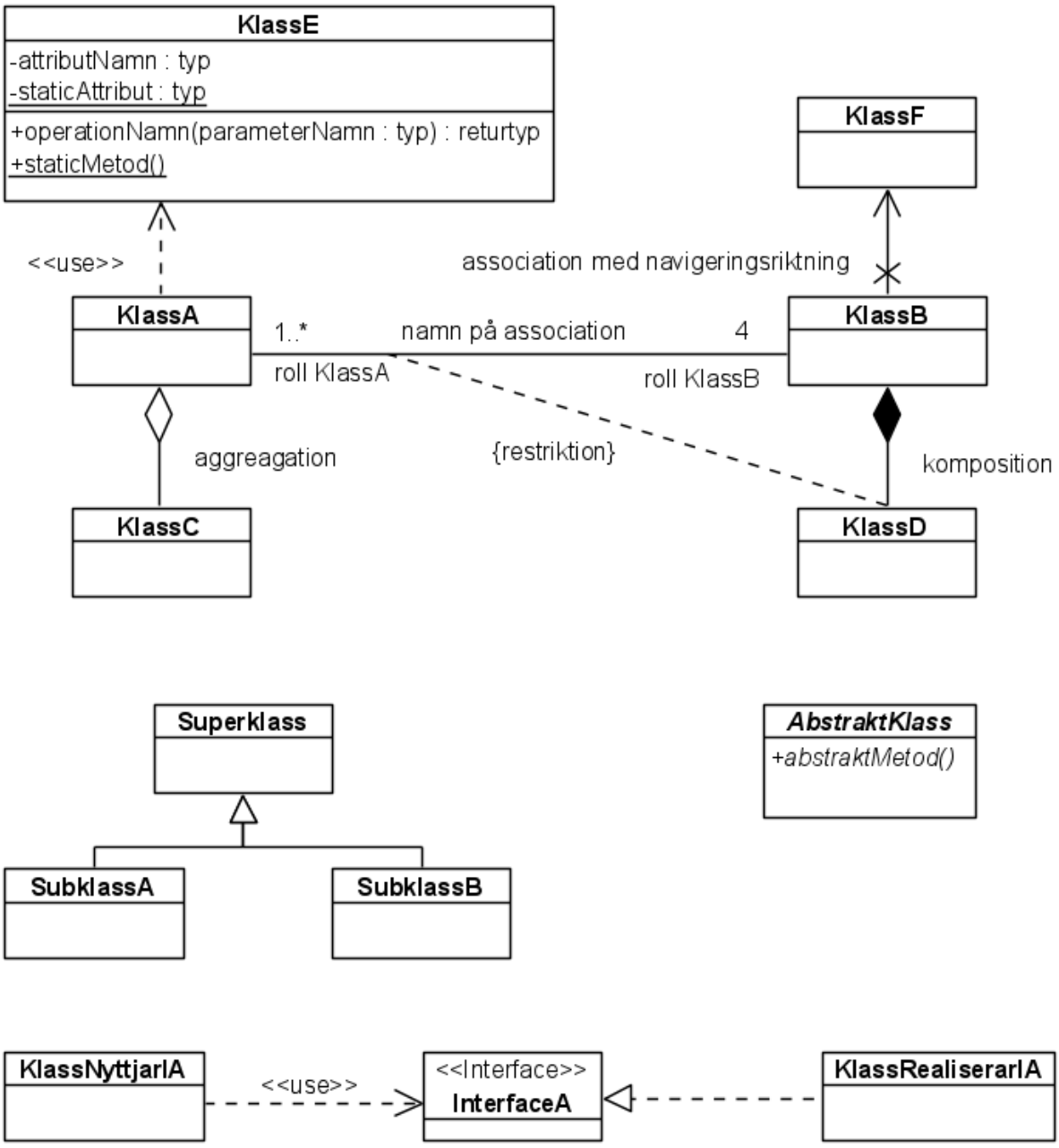
För full poäng krävs minst en rimlig generalisering och en rimlig komposition eller aggregation samt att alla relevanta multipliciteter är utsatta.

Avdrag från full poäng görs om man exempelvis inte har någon rimlig generalisering, komposition eller aggregation samt om multipliciteter saknas. 4-6 poäng ges för lösningar som har rimliga klasser men där något saknas eller någon association inte är helt rimlig.

Avdrag sker i större omfattning och 0-3 poäng ges om det finns direkt felanvändning av någon form av association, att en generalisering är utritad men sedan inte nyttjas rimligt för andra associationer, delar av det som är relevant i lösningen saknas helt eller det finns direkt olämpliga eller inte efterfrågade klasser.

Bedömningen justeras för helheten av den lösning som gets. Det innebär att vissa delar av lösningen kan vara rimliga men om något större problem finns som tyder på inte full förståelse av centrala begrepp kan poängen sättas under 4 poäng.

Notationshjälp klassdiagram

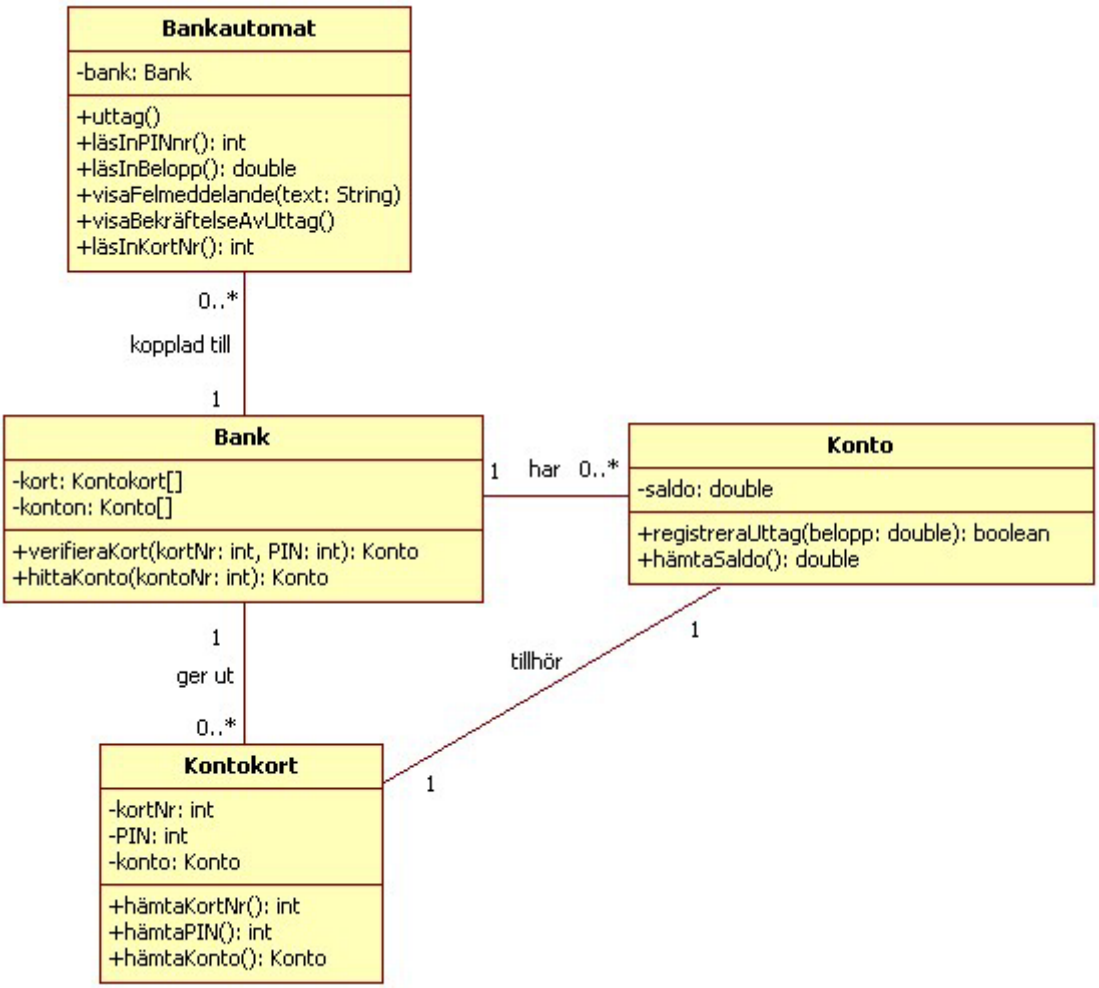


Uppgift 9 7p

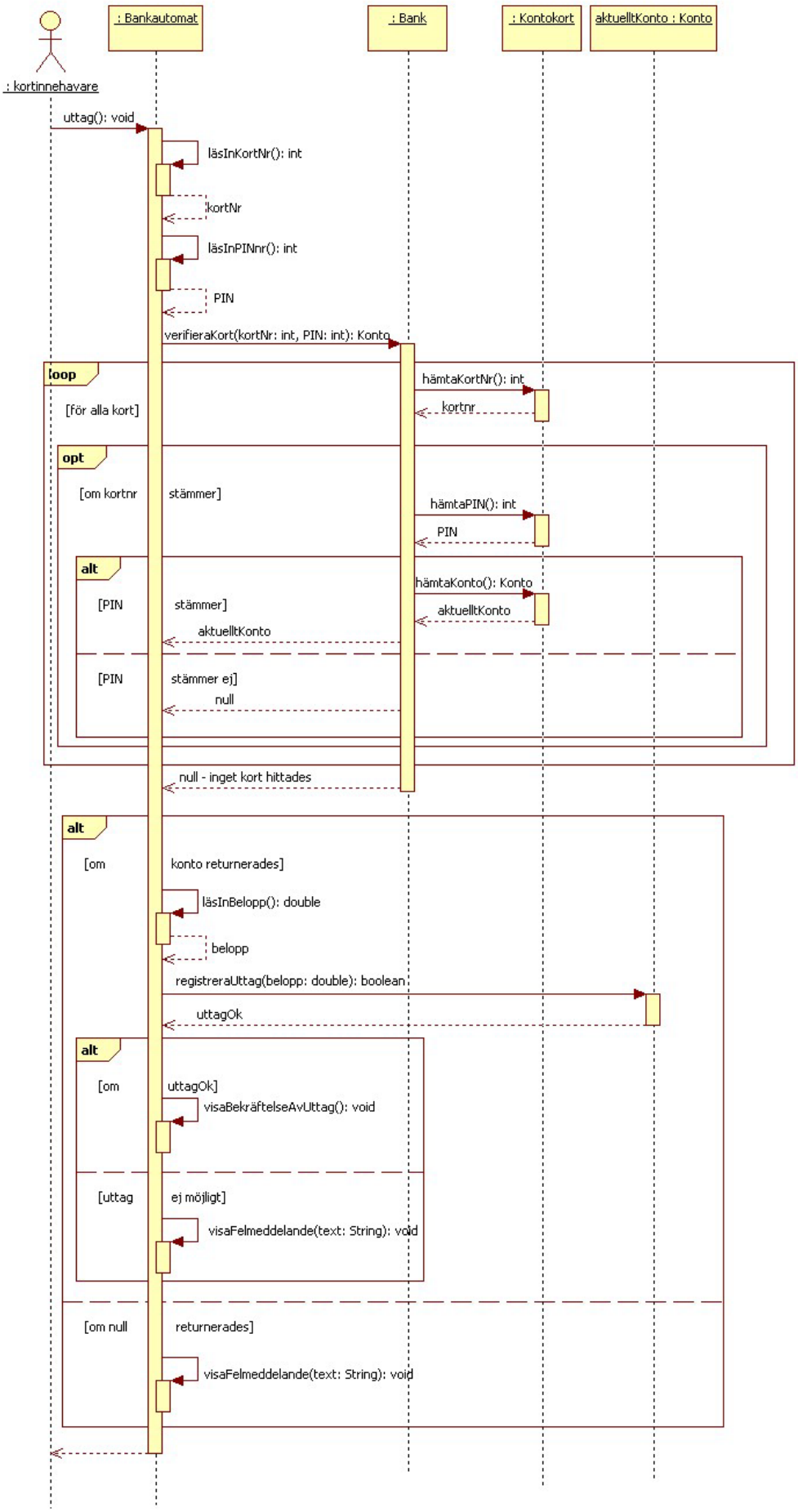
Klassdiagrammet nedan beskriver ett system som hanterar en uttagsautomat där man kan ta ut pengar från ett konto kopplat till ett bankkort.

Givet klassdiagrammet nedan rita ett sekvensdiagram som visar vad som sker när ett uttag sker enligt följande modell: Bankautomaten som är representerad av ett Bankautomat-objekt känner endast till ett Bank-objekt och måste hämta all sin information och övriga objekt via detta. Bankomaten läser in PIN-kod från kortinnehavaren och läser av kortnumret från plastkortet. Plastkortet kan dock inte i sig bekräfta något om PIN-nummer utan detta måste bekräftas via banken. Lösningen ska innebära att bankomaten visar felmeddelanden för om kontot kopplat till kortet inte kan hittas i banken, om PIN inte stämmer eller saldot är för litet för uttaget (du behöver inte visa upprepade försöka att slå in rätt PIN – att visa att Pin anges en gång räcker). Lösningen ska visa vad som sker för att hitta ett konto och verifiera PIN samt uttag om så kan göras.

Korrekt UML-notation ska användas (se notationshjälp nedan). Du skall inte skapa några nya klasser, lägga till operationer eller attribut i befintliga klasser utan skall använda dig av det som visas i klassdiagrammet. Du får däremot skapa en aktör som representerar kortinnehavaren som interagerar med Bankautomat-objektet och initierar sekvensen med anrop till metoden *uttag()*.



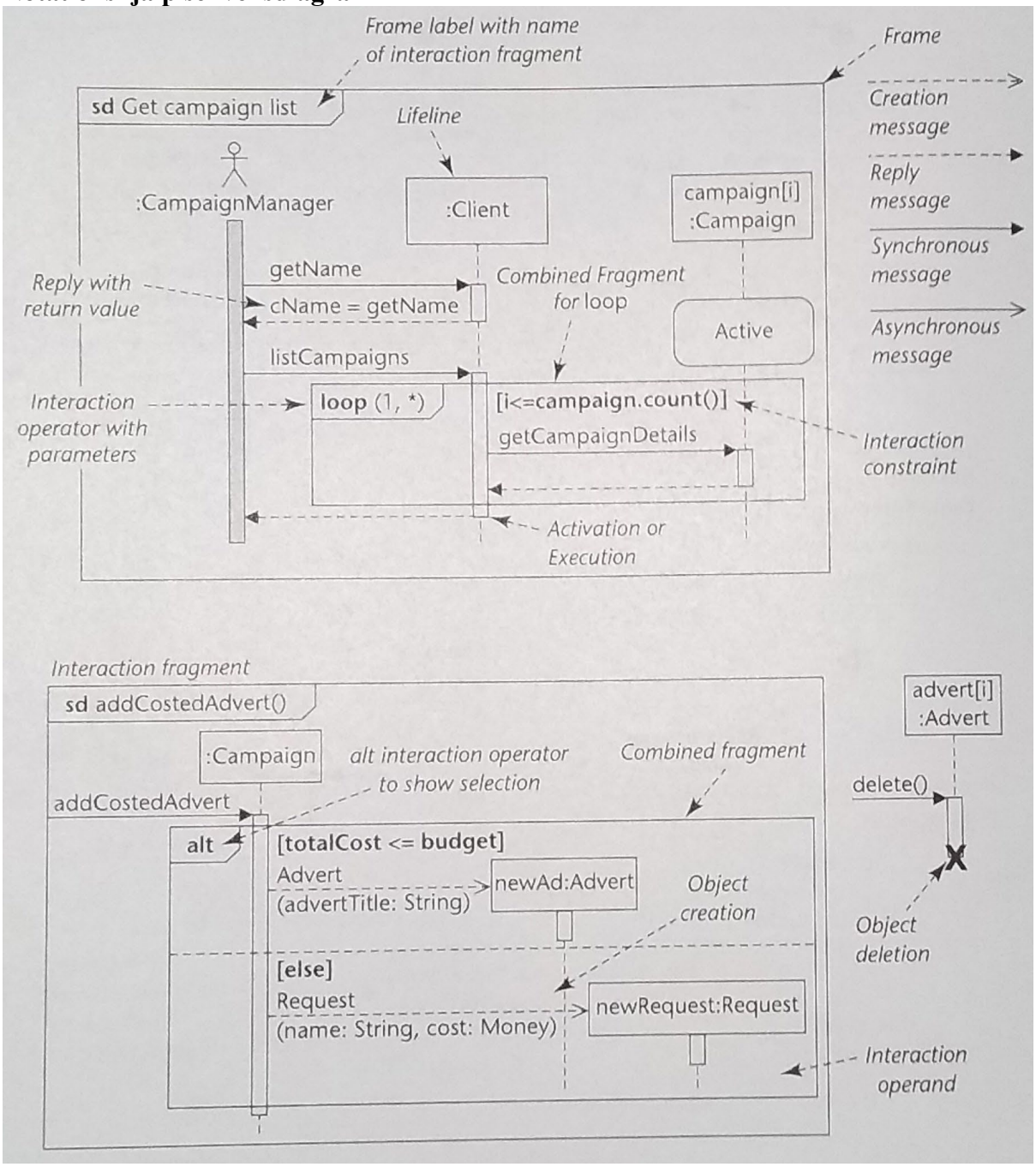
Lösningsförslag



För ett någorlunda rimligt flöde som ger ett slut ges 4 poäng. Bedömningen justeras uppåt beroende på hur väl slutfört flödet är och att detaljer som nödvändiga loopar och if-satser finns och hur rimlig notationen är (viss hänsyn tas till svårigheter att rita i verktyg men alla problem kan inte skyllas på detta om ingen extra förklaring till avsteg från notation finns).

Saker som ger betydligt lägre poäng är om det helt saknas loopar och if-satser där detta behövs för att lösa något, om anrop sker från eller till felaktig klass (det vill säga metoden finns inte i den klass som anropas med den eller ett objekt som inte finns tillgängligt i en klass anropas) eller om delar av flödet helt saknas. Helt orimlig notation ger också större avdrag.

Notationshjälp sekvensdiagram



Uppgifter för betyg VG

Uppgift 10a-d

Eumerationen **ContainerShapeType** är given nedan. Du ska skriva en klass **Container** som använder enumerationen, dess metoder och implementerar interfacet **IContainer**. Senare i uppgiften skall du också skriva ett par konkreta subklasser till Container. Klasserna är inte stora; du ska skriva så mycket kod som frågorna kräver. Figuren nedan visar de källfiler som skall kompletteras i uppgiften.

Observera: Glöm inte att deklarera de variabler du använder, strukturera metoderna på rätt sätt och använda rätt syntax. Alla instansvariabler ska vara deklarerade med modifieraren private.

```
public enum ContainerShapeType {

    Cube(5, "Bronze"),
    Triangle(10, "Silver"),
    Sphere(30, "Gold");

    private final double price;
    private final String material;

    ContainerShapeType(double price, String material){
        this.price = price;
        this.material = material;
    }

    public double getPrice(){
        return price;
    }

    public String getMaterial(){
        return material;
    }
}
```

Uppgift 10a: Interface

Komplettera interfacet vid markeringarna 1 till 7.

```
public interface IContainer {

  //1. En låda (Container) måste ha en typ av enum-typen
ContainerShapeType.
  //2. Komplettera med en getter- och en setter-metod som subklasserna
  //   (i detta fall klassen Container) ska implementera.

  //3. En låda (Container) skall räkna ut och returnera sin volym (volume)
  //   utifrån sin längd, höjd, bredd samt typ av låda.
  //    - Om det är av typen Cube: Volym = bredd*höjd*längd.
  //    - Om det är av typen Triangle: Volym = (bredd*höjd*längd)/2.
  //    - Om det är av typen Sphere: Volym = (4*PI*r^3)/3. Kan använda
  //      höjd/2 som ditt r. (r^3 = r*r*r).
  //4. Komplettera med metod som returnerar volymen som en double.

  //5. En låda (Container) skall räkna ut och returnera sitt totala pris
  //   (totalPrice) utifrån sin volym samt vilken ContainerShapeType lådan
  //   är med sitt unika pris.
  //6. Komplettera med metod som returnerar totala priset som en double
```

```
  //   eller int.

  //7. Skriv en metod getBuildingBlockInfo som skall returnera en String
  //   (mer om denna metod i uppgift 10b).

}
```

Uppgift 10b: Abstrakt klass och abstrakt metod

Skriv en klass **Container** som implementerar interfacet ovan. Klassen skall innehålla all kod som implementation av interfacet kräver, med undantag av metoden **getContainerInfo** som skall definieras som en abstrakt metod i klassen. Du får deklarera variabler och skriva flera metoder (metoderna får dock inte ersätta givna metoder så att de inte behöver användas) ifall du behöver dem för din lösning (instansvariabler ska vara deklarerade med modifieraren private).

Observera: det är **inte** obligatoriskt att skapa och använda konstruktorer i denna och de andra klasser som du skriver i denna uppgift. Setter- och getter-metoder skriver du efter behov för lösningen. Du behöver använda konstruktorer eller get/set-metoder för att lösa uppgiften.

Definiera **getContainerInfo** i klassen Container som skall returnera en string.

I denna del, skriv bara de ändringar som du behöver göra i klassen Container.

Uppgift 10c: Konkreta klasser

Skriv tre klasser, **Cube**, **Triangle** och **Sphere** som subklasser till Container. Deklarera en valfri instansvariabel i varje klass. Ifall du inte kommer på något bra attribut, använd följande:

Klassen Cube: numOfCubeSides (int)

Klassen Triangle: numOfTriangleSides (int)

Klassen Sphere: numOfSphereSides (int)

Alla klasserna skall överskugga (override) den abstrakta metoden **getContainerInfo**. Metoden skall returnera en String; en text, värdet på den valfria instansvariabeln, namnet och priset på materialet som används (beroende på vilken ContainerShapeType), volymen på lådan samt det totala priset. Exempel på output finns sist i uppgiften.

Klasserna behöver inte vara stora utan de skall bara vara så pass kompletta att de tillsammans med Container och IContainer kan kompileras. Små syntax-fel överses vid rättning om de inte påverkar arkitekturen för lösningen.

Uppgift 10d: Dynamisk bindning och Type Casting

Skriv färdigt metoden createContainer (se nedan) så main-metodens anrop fungerar för följande testvärden:

Cube: num of cube sides = 6
Triangle: num of triangle sides = 5
Sphere: num of sphere sides = 1

För Cube: b x h x l = 2 * 4 * 8 (bredd x höjd x längd)
För Triangle: (b x h x l) / 2 = (2 * 4 * 6) / 2 ((bredd x höjd x längd) / 2)
För Sphere: (4 * PI * (r*r*r)) / 3 = 4 * PI * (2*2*2)) / 3

```
public class MainProgram {
    public static void main(String[] args) {
        createContainer(ContainerShapeType.Cube);
        createContainer(ContainerShapeType.Triangle);
        createContainer(ContainerShapeType.Sphere);
    }

    public static void createContainer(ContainerShapeType csType) {
        //Komplettera
        //Krav:
        // - dynamisk bindning (och type casting, beroende på din lösning).
        // - metoderna (alla tre metoderna måste anropas):
        //   getContainerInfo(),
        //   calculateContainerVolume(ContainerShapeType csType) och
        //   calculateContainerPrice(ContainerShapeType csType)
        //   (två sista metoderna som räknar volymen och total pris, och
får in
        //   vilken typ av låda det är) måste anropas.
    }
}
```

Utdata från main-metoden:

Nice triangle
triangle has 5 sides
triangle material: Bronze
price on material: 5.0kr
Container has a volume of 48cm3
Total price for container: 240.0kr

Fantastic cube
cube has 6 sides
cube material: Silver
price on material: 10.0kr
Container has a volume of 24cm3
Total price for container: 240.0kr

Perfect sphere
sphere has 1 sides
sphere material: Gold
price on material: 30.0kr
Container has a volume of 33cm3
Total price for container: 990.0kr

Kom ihåg att dela upp koden så att det är bra strukturerar och lätt läst. Om du vill skriva kommentarer använd dig av // som vanligt när du skriver kod.