Tentamen på
Kurs:
DA339A, Objektorienterad programmering

Provkod: 2002 Tentamen 2, 2,5 hp

210416 kl 14.15 – 19.15

Tillåtna hjälpmedel:

- **Inga tillåtna hjälpmedel utöver det som finns i den trycka tentamen och eventuell medtagen engelsk-svensk ordbok eller annan språkkombination (anteckningar får inte förekomma i medtagna ordböcker)**

Betygsgränser

- Del med uppgifter för G: 40 poäng
- Del med uppgifter för VG: 1 uppgift med deluppgifter, bedöms i sin helhet som godkänd eller inte godkänd

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

För betyg VG Väl godkänd krävs betyg G på del med uppgifter för betyg G och att uppgiften på del med uppgifter för betyg VG bedöms som godkänd i sin helhet.

Övriga tentamensanvisningar

- 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.
- Skriv läsligt och använd gärna stödlinjerna i tentamensomslaget. Svar som inte kan läsas på grund av otydlig handstil erhåller inte poäng.
- Röd penna får ej användas för svar (reserverad för rättning).
- 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 så krävs att denna är lämpligt formaterad så man tydligt kan se vilka blocka av kod som hänger samman. Indentera väl!
- Häfte med frågor och anteckningspapper/kladdpapper får tas med från tentamenssalen när tentan lämnats in.

Uppgifter för betyg G 40 p

Uppgift 1 10p 0,5/påstående

Vilka påståenden A-T nedan är korrekta/sanna och vilka är felaktiga/falska? Alla påståenden kan förutsättas vara i förhållande till Java. Svara genom att lista alla bokstäver som gäller för frågorna med ett kommatecken emellan, enligt modellen:

Korrekta/sanna: X, Y, Z (obs. X,Y,Z skall ersättas av delfrågans bokstav)
Felaktiga/falska: Å, Ä (obs. Å,Ä skall ersättas av delfrågans bokstav)

Skriv texten ”Korrekta/sanna” och ”felaktiga/falska” innan listan med bokstäver. Om det bara är två listor med bokstäver vet vi inte vilka svar som du anser är vad.

Fråga	**Påstående**
A	Om klasserna A och B är associerade med en komposition, där A är helheten, så gäller att om ett objekt av A raderas så ska objekt av klassen B som har en länk till objektet av A också raderas.
B	Ett klassdiagram visar inte vilka objekt som finns.
C	En klass kan inte ha en association till sig själv.
D	Ett klassdiagram är ett statiskt diagram.
E	Om två klasser har flera associationer mellan sig så får dessa inte vara av samma typ.
F	Om navigeringsriktning inte är utsatt på en association så finns inget beroende mellan klasserna.
G	Vid överskuggning har man samma metod med olika implementation i subklasser i en klasshierarki.
H	En subklass ärver alla publika medlemmar av sin super klass inklusive konstruktor(er).
I	I ett `try-catch`, `finally` sats, kan man bara ha ett `try`-block och bara ett `finally` block men ett eller flera `catch` block.
J	Nyckelordet `throw` används för att kasta ett undantag inne i en metod medan nyckelordet `throws` används i metodens signatur för att prata om att ett undantag kan inträffa i metoden.
K	En klass kan ha endast en superklass och kan implementera endast ett interface.
L	Ett interface kan inte implementera ett annat interface.
M	Om en subklass **inte** skriver kod till alla abstrakta metoder i superklassen, skall subklassen definieras också som `abstract` och låta subklassen en nivå lägre att skriva kod till abstrakta metoden.
N	När typen av objekt är bestämt vid kompilering, sker en dynamisk bindning.
O	En superklass konstruktor anropas och exekveras alltid först och sen exekveras subklassens egen konstruktor.

P	Överskuggning (overriding) innebär att en metod i en subklass ska användas i stället för den metod som subklassen har ärvt från en överordnad klass. En överskuggande metod har samma namn och samma argument som metoden som den överskuggar.
Q	Överlagring (overloading) är ett annat namn för överskuggning (overriding), dvs. båda används vid arv.
R	Alla metoder i ett interface är implicit `public` och `abstract`.
S	Instansvariabler är inte tillåtna i ett interface men det går bra att använda konstanter. Konstanterna är `public, static, final`.
T	När en array eller ett annat objekt skickas som argument till en metod, tillämpas en mekanism som heter **pass-by-value** vilket betyder att en kopia av objektets data skickas till metoden.

Svar:
Sant: A, B, D, G, I, J, M, P, O, R, S,
Falskt: C, E, F, H, K, L,N, Q, T

Uppgift 2 2p

Du har följande förslag på en superklass och tre subklasser till denna i ett system som hanterar studieresultat på kurser:

Super-klass	Sub-klasser
Resultat	Poäng, Betyg, Namn

Motivera varför detta är en lämplig generalisering **eller inte** är en lämplig generalisering. Ditt svar ska motivera för respektive sub-klass huruvida det är en lämplig generalisering eller inte i förhållande till den givna super-klassen.

Svar: Det är inte lämpliga subklasser utan alla tre förslagen till subklasser är snarare attribut i klassen resultat. Hade förslaget Namn kan möjligen vara en referens till en annan klass som representerar en person på något sätt men detta är då inte en subklass till resultat.

Uppgift 3 2p

Förklara kortfattat konceptet MVC Model-View-Controller och varför detta är relevant vid programutveckling (med kortfattat avses att du inte borde behöva mer än runt 6-8 meningar för ditt svar – en mening för varje del i konceptet och några meningar till för att förklara relevansen).

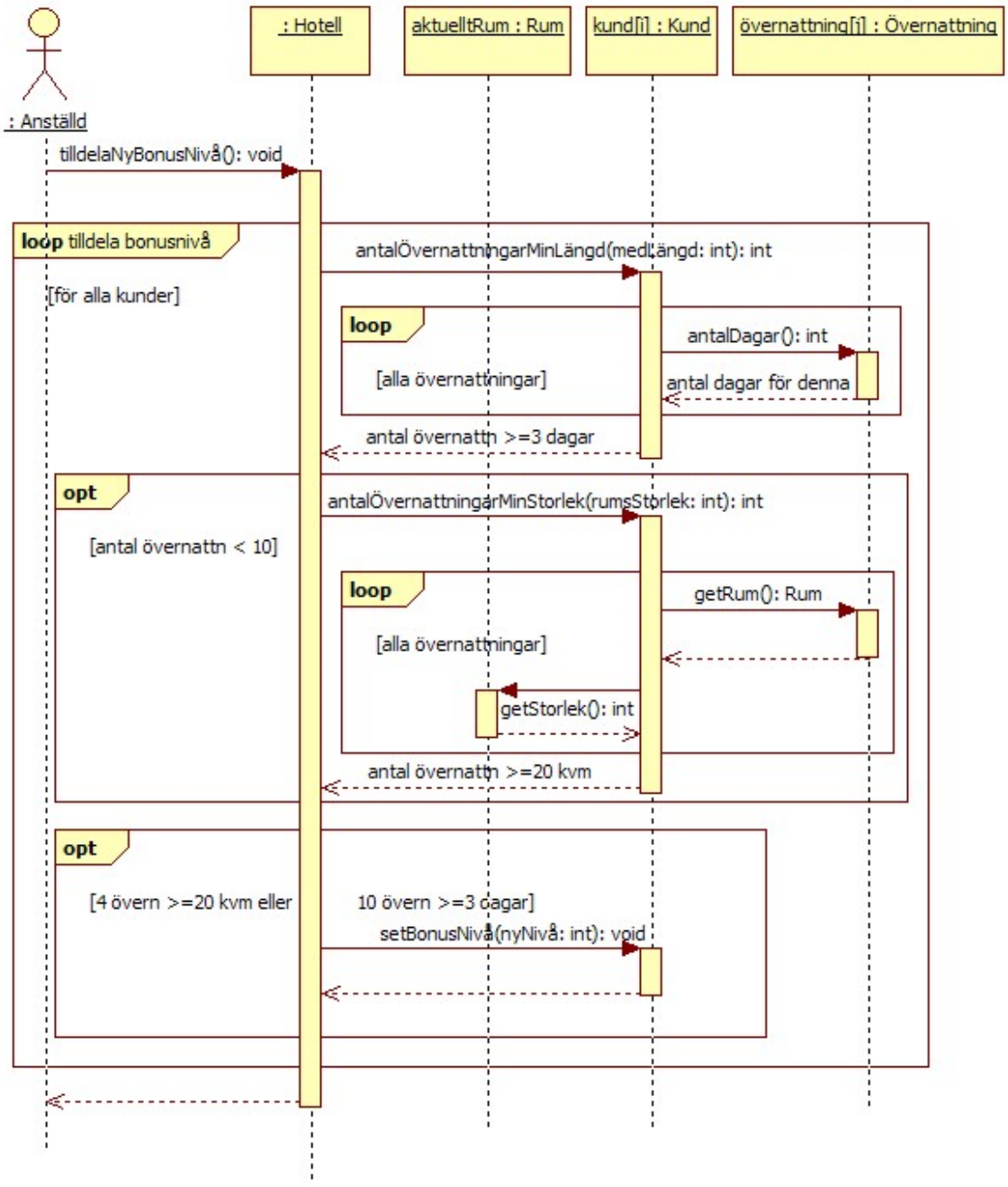
Svar: Model är de delar som på något sätt ansvarar för hur data och information är lagrat och organiserat och ansvarar för de operationer som direkt kan göras gentemot den här datan. View är de delar som ansvarar för att skapa ett gränssnitt mot användaren eller andra system och hur input och output förmedlas från och till dessa. Controller-delar ansvarar för logiken i programmet och flödet som sker utifrån given input och tillstånd i data. MVC används för att skapa mer struktur i programkod och en struktur som gör ett system lättare att underhålla och flexiblare där delar kan förändras lättare , exemepvlsi förändra View utan att behöva förändra Model.

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. Koden skall visa vilka metoder de två klasserna har samt den kod i metoderna som går att utläsa från diagrammet. Vissa antaganden kan göras om 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) {…}
}
```

Uppgift 5a 2p

Klasserna A, B och C som är givna enligt nedan.

```
class A {
    void print(){
        System.out.println("A");
    }
}

class B extends A {
    void print() {System.out.println("B"); }
}

class C extends A {
    void print(){
        System.out.println("C");
    }
}
```

Metoden Test i klassen TestOverriding använder sig av klasserna A, B och C:

```
class TestOverriding {

   void Test()
   {
      A obj = new C();
      obj.print();  //1
      obj = new A();
      obj.print(); //2
    }
}
```

Fråga: Vilken utskrift erhålls när metoden i klassen Test körs?

Svar:
C
A

Uppgift 5b 2p

Fråga: Vilken utskrift erhålls när metoden sumNumbers exekveras?

```
class TestOverloading {
    public double sum (int a, int b)
    {
        double sum = a+b;
        System.out.println("#1: sum = " + sum);
        return sum;
    }
    public double sum(double a, double b)
    {
        double sum = a+b;
        System.out.println("#2: sum = " + sum);
        return sum;
    }

    public void sumNumbers()
    {
        double a = 5;
        double b = 4;
        sum(a, b);
    }
}
```

Svar

#2: sum = 9.0

Uppgift 5c 4p

Klasserna Person, Player och Address innehåller de instansvariabler som visas i koden nedan. Du ska komplettera de konstruktorer som skall finnas enligt anvisningar som är markerade i koden.

```
class Person
{
    private String name;
    private int age;
    private Address adress;

   //To do 1 1p
   //Constructor with name, age and city as parameters
   //Use the arguments to initialize the related objects
    public Person (String name, int age, String city)
    {

    }
}


public class Player extends Person
{
    private int number;
    private int goals;

     //To do 2, 2p
     //Constructor with name, and city, age, number, goals as
     //parameters. Use the arguments to initialize the related
     //objects
    public Player(String name, String city, int age, int number, int goals)
    {

    }

}


class Address
{
    private String street;
    private String city;
    private String zipCode;

     //To Do 3, 1p
     //Write a constructor with city as parameter
     //Use the argument to initialize the object of the class
}
```

Metoden main i klassen nedan visar hur ett objekt av **Player** kan skapas

```
//Test program
class Program
{
    public static void main(String[] args)
    {
        int age = 39;
        int number = 10;
        int goals = 0;

        Player player = new Player("Apu", "Lund", 39, 10, 0);
    }
}
```

Lösningsförslag

```
class Person
{
    private String name;
    private int age;
    private Address adress;

    //To do 1 1p
    //Constructor with name, age and city as parameters
    //Use the arguments to initialize the related objects
    public Person (String name, int age, String city)
    {
        this.namne = name;
        this.age = age;
        address = new Address(city):
    }
}


public class Player extends Person
{
    private int number;
    private int goals;

     //To do 2, 2p
     //Constructor with name, and city, age, number, goals as
     //parameters. Use the arguments to initialize the related
     //objects
    public Player(String name, String city, int age, int number, int goals)
    {
        super(name, age, city);
        this.number = number;
        this.goals = goals;
    }

}
```

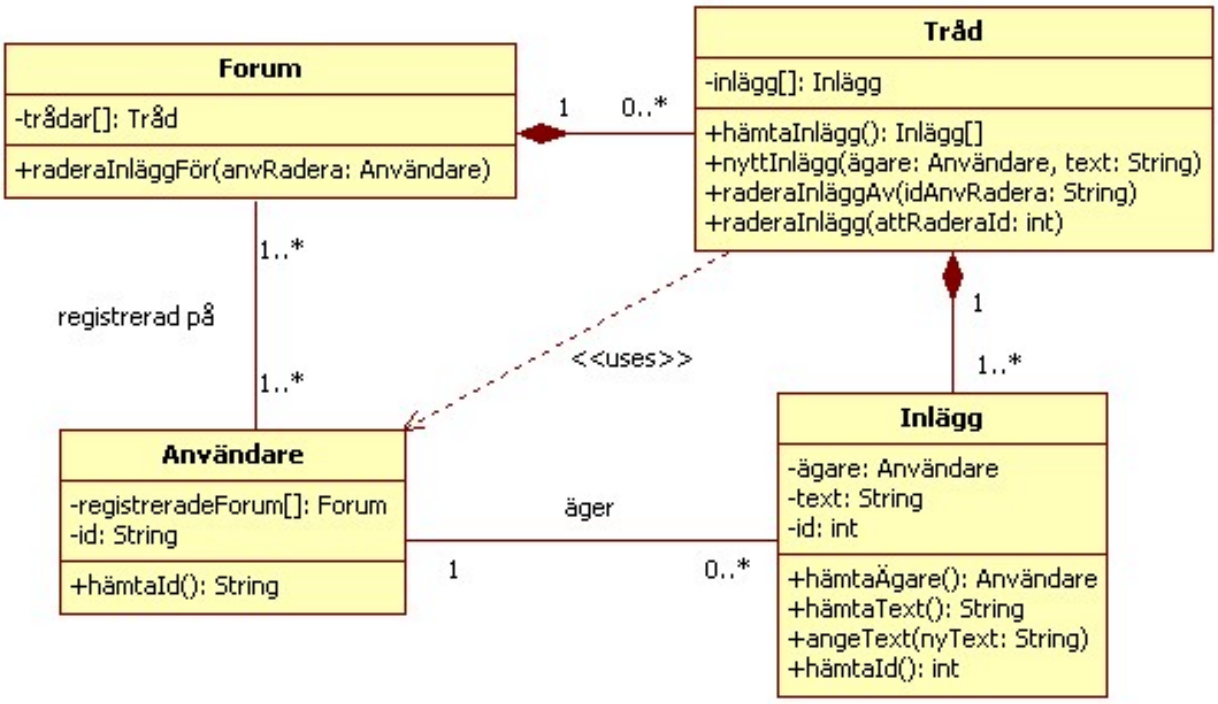
```
class Address
{
    private String street;
    private String city;
    private String zipCode;

    //To Do 3, 1p
    //Write a constructor with city as parameter
    //Use the argument to initialize the object of the class
       public Adress(String city)
       {
              this.city = city;
              /*de två övriga instansvariablerna kan hanteras på
olika sätt och lämnas till att få defaultvärden eller explicit
sättas till något så länge*/
       }
}
```

Uppgift 6 7p

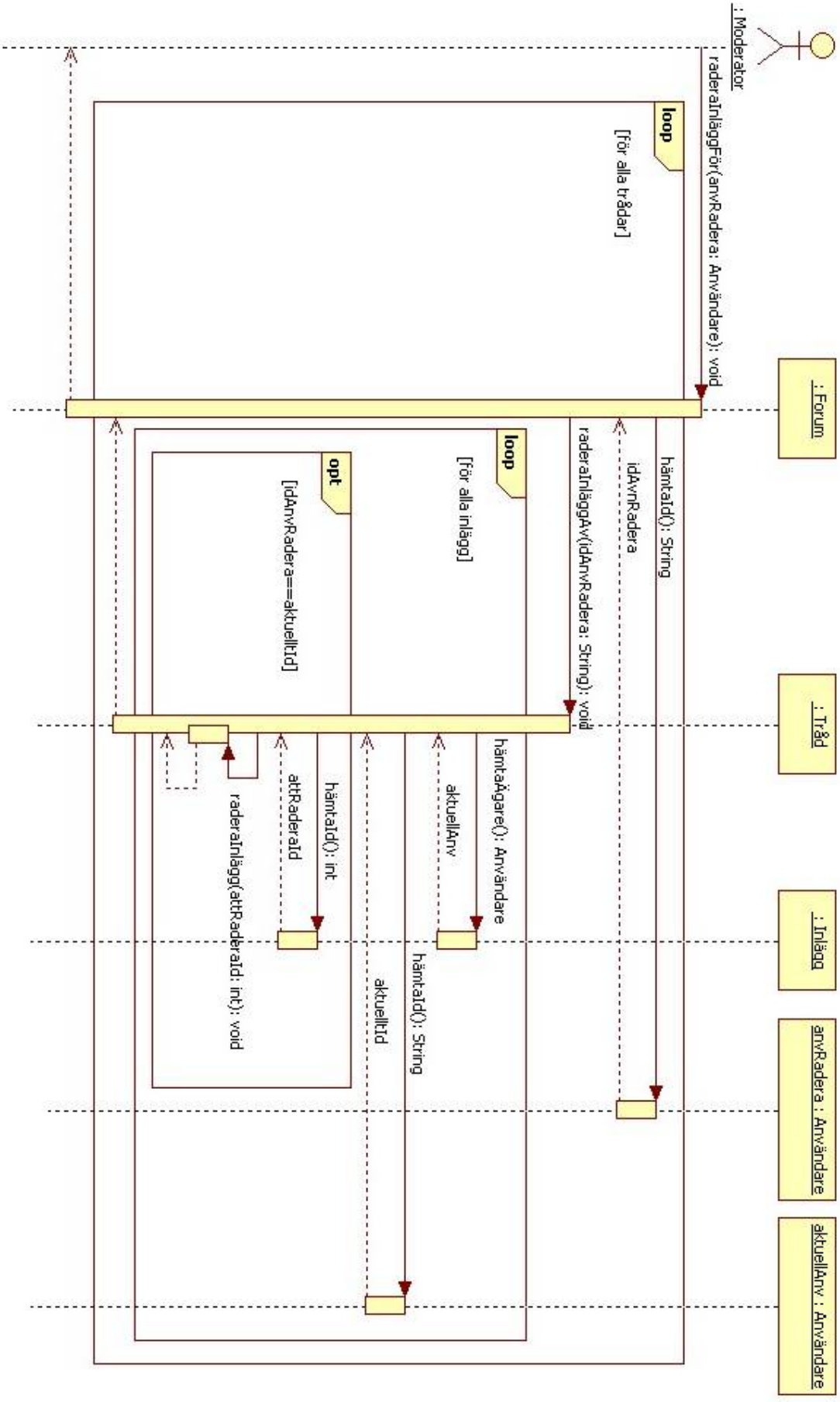
Klassdiagrammet nedan beskriver ett system som hanterar ett diskussionsforum.

Givet klassdiagrammet nedan skriv ett sekvensdiagram som visar vad som sker när en moderator får i uppdrag att radera alla inlägg gjorda av en viss given användare på ett forum.

Du får inte lägga till attribut eller operationer till klasserna. Du får däremot skapa en aktör som representerar moderatorn som ska radera inlägg och du kan förutsätta att anropet från aktören sker till metoden raderaInläggFör(anvRadera: Användare) i klassen Forum där parametern är den givna användaren som inlägg ska raderas för.



Lösningsförslag



Uppgift 7 7p

Konstruera ett lämpligt klassdiagram utifrån systembeskrivningen nedan. Identifiera 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 möjligt och 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 eventuella lämpliga antaganden där detta kan vara rimligt. Gör inga tillägg med information som 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 Bilaga 1).

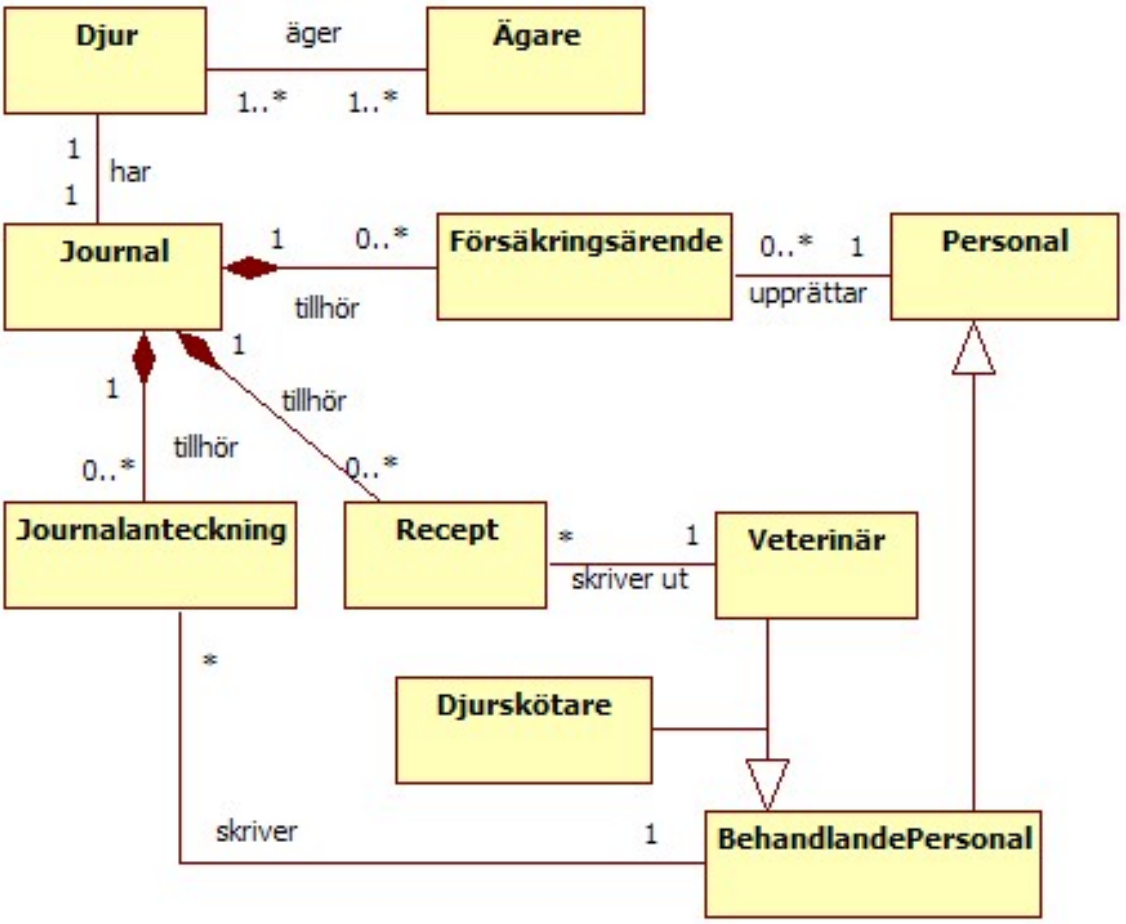
Systembeskrivning
På en veterinärklinik så måste man föra journaler över de djur man behandlar. För varje djur som behandlas så finns en journal som består av flera journalanteckningar. Varje djur har minst en ägare angiven. Personer kan givetvis äga flera djur som alla behandlats på veterinärkliniken någon gång. När ett nytt djur kommer med sin ägare till veterinärkliniken så registreras uppgifter om djurets art, namn, ålder och andra generella uppgifter samt att det upprättas en journal. Denna journal innehåller då ännu inga anteckningar. Om inte ägaren finns registrerad sedan tidigare på veterinärkliniken så görs detta genom att man registrerar namn och kontaktuppgifter.

För att underlätta för ägare som har sina djur försäkrade hos något försäkringsbolag så sköter veterinärkliniken viss kontakt med försäkringsbolagen genom att skicka in uppgifter som krävs för att ägaren ska få ersättning på sin försäkring. Till journalen kan då även finans kopplat ett antal försäkringsärenden.

På veterinärkliniken så arbetar olika typer av personal som kan behöva arbeta med registret.
Veterinärer kan skriva journalanteckningar gällande all form av undersökning och behandling och får lov att skriva ut recept på läkemedel och förskriva läkemedel som ges på kliniken. När en veterinär skriver ut ett recept så ska detta recept registreras som en del av journalen. Det måste registreras vem som utfärdade receptet. Läkemedel som ges på kliniken skrivs som en journalanteckning. Djurskötare kan skriva journalanteckningar som gäller enklare undersökningar och ge vissa läkemedel som ordinerats av veterinär. Det är viktigt att det registreras vem som skrivit en journalanteckning om där skulle uppstå problem.

I receptionen arbetar personal som inte utför någon form av behandling eller undersökning och dessa behöver inte heller vara utbildade djurskötare eller veterinärer. Denna form av personal arbetar exempelvis med att registrera uppgifter, kontakta försäkringsbolag och upprätta försäkringsärenden och göra tidbokningar. Djurskötare och veterinärer kan givetvis utföra dessa sysslor även om de inte vanligen gör detta.

Lösningsförslag



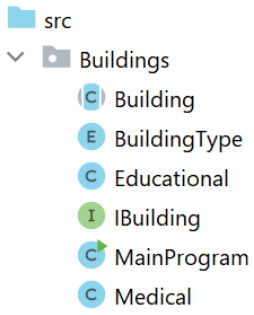
Uppgifter för betyg VG

Uppgift 8

Eumerationen **BuildingType** är given. Du ska skriva en klass **Building** som använder enumerationen och också implementerar interfacet **IBuilding**. Senare i uppgiften skall du också skriva ett par konkreta klasser. Klasserna är inte stora; du ska skriva så mycket kod som frågorna kräver.

Observera: Glöm inte att deklarera de variabler du använder, strukturera metoderna på rätt sätt och använda rätt syntax. Figuren till höger visar de källfiler som skall kompletteras i uppgiften.

```
public enum BuildingType
{
    Medical,        //sjukvård
    Residential,    //bostad
    Educational,    //utbildning
    Recreational    //fritid
}
```



Uppgift 8a: interface

Komplettera interfacet vid markeringarna 1 till 3.

```
public interface IBuilding
{
  //En byggnad måste ha en typ enligt enum-typen BuildingType
  //1. Komplettera med en getter och en setter-metod som
  //sumklasserna (i detta fall klassen Building) ska implemetera.

  //En byggnad skall räkna och returnera sin yta utifrån sin
  //längd och bredd.  area = bredd * längd
  //2.Komplettera med metod som returnerar arean som ett double tal.

  //3.Skriv en metod getBuildingInfo som skall returnera en string

}
```

Uppgift 8b: abstract klass och abstract metod

Skriv en klass **Building** som implementerar interfacet. Klassen skall innehålla all kod som implementation av interfacet kräver, med undantag av metoden **getBuildingInfo** som skall definieras som en abstract metod i klassen. Du får gärna deklarera variabler och skriva flera metoder i fall du behöver dem.

Obs. 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.

Definiera **getBuildingInfo** i klassen Building som skall returnera en string. I denna del, skriv bar de ändringar som du behöver göra i klassen Building.

Uppgift 8c: konkreta klasser

Skriv två klasser, Medical och Educational som subklasser till Building. Deklarera en valfri instansvariabel i varje klass. I fall du inte kommer på något bra attribut, använd följande:
Klassen Educational: numOfClassRooms (int)
Klassen Medical: numOfBeds (int)

Båda klasser skall överskugga (override) den abstrakta metoden **getBuildingInfo**. Metoden kan returnera en test string såsom: "Nice University" resp. "Fantastic hospital".

Klasserna behöver inte vara stora utan de skall vara så pass komplett så de tillsammans med Building och IBuilding kan kompileras. Små syntax-fel är tillåtna.

Uppgift 8d: Dynamisk bindning

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

Educational: num of class rooms 2000
Medical: num of beds =300
For båda: h x b = 200 * 150

```
public class MainProgram
{
    public static void main(String[] args)
    {
        createBuilding(BuildingType.Educational);
        createBuilding(BuildingType.Medical);
    }

    public static void createBuilding(BuildingType bldgType)
    {
       //Komplettera
       //Krav:
       //  dynamisk bindning
      //  metoderna getBildingInfo och calculateArea()
      //  måste anropas.

    }
}
```

Utdata från main-metoden:

```
Nice University
Building has an area 30000.0 m2
Fantastic hospital
Building has an area 30000.0 m2
```

Svar:

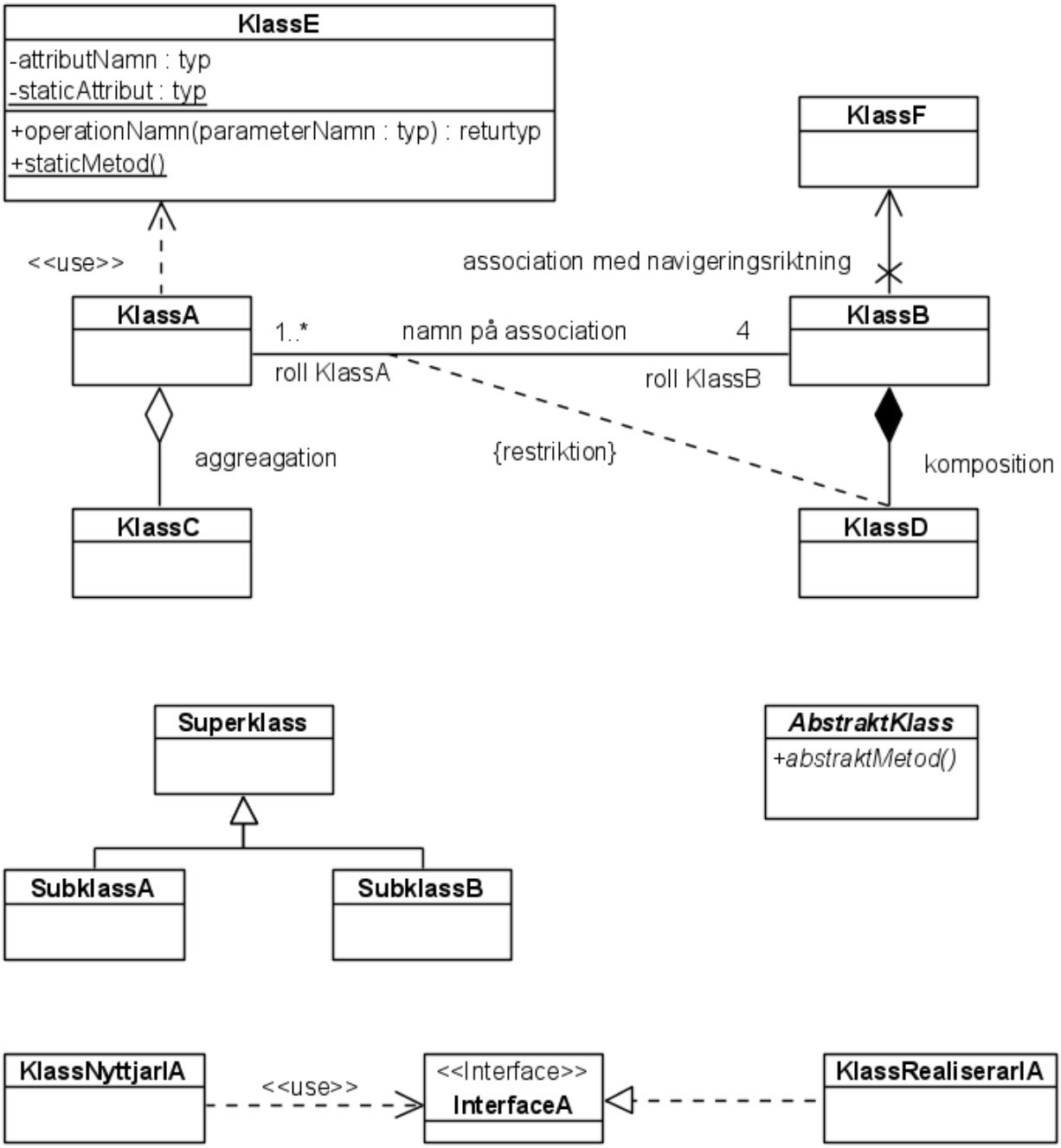
I källkoden som visas härnedan, är id en extra instansvariabel i Building men den ingår inte i uppgiften.

```
public interface IBuilding
{
    void setID(String id);
    void setBuildingType(BuildingType type);
    BuildingType getBuildingType();
    String getBuildingInfo();
    double calculateArea();

}


public abstract class Building implements IBuilding
{
    private String id;
    private BuildingType buildingType;
    private double width;
    private double height;

    public String getId()
    {
        return id;
    }

    public void setId(String id)
    {
        this.id = id;
    }

    public double getWidth()
    {
        return width;
    }

    public void setWidth(double width)
    {
        this.width = width;
    }

    public double getHeight()
    {
        return height;
    }

    public void setLength(double height)
    {
        this.height = height;
    }


    public BuildingType getBuildingType()
    {
        return buildingType;
    }

    public void setBuildingType(BuildingType buildingType)
    {
        this.buildingType = buildingType;
    }
```

```
    public void setID(String id)
    {
        this.id = id;
    }
    public double calculateArea()
    {
        return height*width;
    }
    public abstract String getBuildingInfo();
}


public class Educational extends Building
{
    public int getNumOfClassRooms()
    {
        return numOfClassRooms;
    }

    public void setNumOfClassRooms(int numOfClassRooms)
    {
        this.numOfClassRooms = numOfClassRooms;
    }

    private int numOfClassRooms;

      @Override
    public String getBuildingInfo()
    {
        return "Nice University";
    }

}
public class Medical extends Building
{
    public int getNumOfBeds()
    {
        return numOfBeds;
    }

    public void setNumOfBeds(int numOfBeds)
    {
        this.numOfBeds = numOfBeds;
    }

    private int numOfBeds;

    public Medical()
    {

    }

    @Override
    public String getBuildingInfo()
    {
        return "Fantastic hospital";
    }
```

```
public class MainProgram
{
    public static void main(String[] args)
    {
        createBuilding(BuildingType.Educational);
        createBuilding(BuildingType.Medical);
    }

    public static void createBuilding(BuildingType bldgType)
    {
        Building building = null;
        switch (bldgType)
        {
            case Medical:
                building = new Medical();
                ((Medical)building).setNumOfBeds(300);
                break;
            case Educational:
                building = new Educational();
                ((Educational)building).setNumOfClassRooms(2000);
                break;
                default:
                    System.out.println("Not implemented!");


        }
        if (building != null)
        {
            building.setBuildingType(bldgType);
            building.setLength(200);
            building.setWidth(150);
            System.out.println(building.getBuildingInfo());
            System.out.println("Building has an area " +
                        building.calculateArea() + "m2");
        }

    }
}
```

Bilaga 1 UML-notation klassdiagram



Bilaga 1 UML-notation sekvensdiagram